Accounting for Missing Events in Statistical Information Leakage Analysis

Seongmin Lee
MPI-SP
Bochum, Germany
seongmin.lee@mpi-sp.org

Shreyas Minocha[†]
Georgia Institute of Technology
Atlanta, GA, USA
shreyas@shreyasminocha.me

Marcel Böhme
MPI-SP
Bochum, Germany
marcel.boehme@acm.org

***Abstract*—The leakage of secret information via a public channel is a critical privacy flaw in software systems. The more information is leaked per observation, the less time an attacker needs to learn the secret. Due to the size and complexity of the modern software, and because some empirical facts are not available for a formal analysis of the source code, researchers started investigating statistical methods using program executions as samples. However, current statistical methods require a high sample coverage. Ideally, the sample is large enough to contain every possible combination of secret × observable value to accurately reflect the joint distribution of ⟨secret, observable⟩. Otherwise, the information leakage is severely underestimated, which is problematic as it can lead to overconfidence in the security of an otherwise vulnerable program.**

In this paper, we introduce an improved estimator for information leakage and propose to use methods from applied statistics to improve our estimate of the joint distribution when sample coverage is low. The key idea is to reconstruct the joint distribution by casting our problem as a multinomial estimation problem in the absence of samples for all classes. We suggest two approaches and demonstrate the effectiveness of each approach on a set of benchmark subjects. We also propose novel refinement heuristics, which help to adjust the joint distribution and gain better estimation accuracy. Compared to existing statistical methods for information leakage estimation, our method can safely overestimate the mutual information and provide a more accurate estimate from a limited number of program executions.

***Index Terms*—Information Leakage, Mutual Information, Statistical Estimation**

I. Introduction

Adversaries should not be able to infer secret data by observing public information from a system. In fact, developers need to minimize the information an adversary can infer about secret values to protect a data-sensitive program from an adversary who can observe the program's behavior. If information about the secret is *leaked*, i.e., an adversary can infer it from the observed behavior of the program more than the developer intended, it becomes a critical privacy bug [1]. Therefore, *information leakage* needs to be identified/tested in the development process. But how can we even measure how much an adversary would learn about a secret?

Information leakage can be quantified using *mutual information* (MI) between secret values and observable values. MI is an information-theoretic measure that quantifies the correlation between two random variables related as a joint probability distribution. The MI between the secret and the observable values is the difference between the uncertainty of the secret value before observing the observable value and the uncertainty of the secret value after observing the observable value. Therefore, the MI represents the remaining uncertainty about the secret value after observing the observable value.

There is a stream of works that propose to measure the MI based on a white-box approach, which calculates the *number of inputs* corresponding to every possible ⟨secret, observable⟩ pair using symbolic execution and model counting, and computes the MI via the the joint probability distribution [2]–[5]. Nevertheless, even if we assume white-box access to the system, analytical methods are often intractable due to the size or complexity of the system, or since empirical facts, such as typical sensor data or average server load, are not available during source code analysis [6], [7]. These challenges have prompted exploration into techniques for estimating MI in a black-box fashion.

A more recent stream of works proposes to measure the MI based on a scale-oblivious black-box approach using statistical estimation methodologies [8]–[10]. Given a random sample of program executions containing ⟨secret, observable⟩ pairs, they apply a statistical model to infer the joint probability distribution and estimate the MI from it. However, the current state-of-the-art statistical methods suffer from the underestimation problem in the small-sample regime, where low-probability events, i.e., ⟨secret, observable⟩ pairs, are missing in the sample [11]. Underestimating the information leakage is especially harmful and *unsafe* in the sense of security evaluation as it can lead to overconfidence in the security of the vulnerable program. They require a sufficient number of samples for every possible ⟨secret, observable⟩ pair to estimate the MI accurately, which is a strong restriction in practice.

In this paper, we propose a method to recover the probabilities of the *undetected* events in the joint probability distribution concealed by the small sample size by adapting a methodology from biostatistics and then directly compute the MI from this recovered distribution. This methodology is Chao et al.'s *species-rank abundance distribution estimation* [12]. From a limited number of samples, it seeks to recover an unknown multinomial distribution (MD) by estimating the

[†]The author conducted this research while doing an internship at MPI-SP.

```
int16_t prog(int secret) {
  int16_t observable = secret & 0xff00;
  return observable;
}
```

Fig. 1: Example program. `int16_t` is a 16-bit integer type.

probabilities of the observed events *and* the set of potential probabilities of the unobserved events. As Chao's estimation methodology is designed for the MD of a *single* random variable, we suggest two approaches to extend the methodology for the joint distribution of *two* random variables: The *Flatten* approach considers a ⟨secret, observable⟩ pair as a single random variable. The *By-Secret* approach considers, for every secret, a conditional distribution of the observable given the secret as a single random variable MD. Moreover, we propose several heuristics for a more realistic reconstruction of the joint distribution by adjusting the probabilities assigned to the unobserved events.

We evaluate our proposed method on a set of benchmark subjects against the state-of-the-art statistical MI estimation methods; the benchmark consists of programs used in previous literature as well as data from the real-world software. We investigate the effectiveness of the proposed method with respect to the accuracy, i.e., the mean squared error (MSE) of the estimated MI, and the safety, i.e., the probability of underestimating the MI. We also investigate the effectiveness of each component of the proposed method and the effect of the refinement heuristics. The result shows that our estimator accurately estimates the MI as close to the state-of-the-art statistical methods with a small number of samples. Yet, it produces a safe overestimation of the MI even in the presence of missing events, unlike the state-of-the-art statistical methods that suffer from the underestimation problem. Especially in the real-world software, where the size of the observable space is vast, our method turns out to be much more accurate than the previous state-of-the-art statistical methods.

The main contributions of this paper are as follows:

- We propose a novel integration of the statistical MI and MD estimations to overcome the "missing low-probability event" problem in the statistical MI estimation.
- We suggest two approaches to reconstruct the joint distribution of the ⟨secret, observable⟩ pairs from the estimated probabilities of the observed and unobserved events.
- We propose novel refinement heuristics, which adjust the joint distribution to have a better estimation accuracy.
- We evaluate our methodology on a set of benchmarks and compare its accuracy and safety to state-of-the-art statistical methods. We make experimental data and analysis available at: https://github.com/niMgnoeSeeL/ChaoMI

II. Background

A. Quantifying Information Leakage

We first explain how the information leakage of a program is quantified with mutual information. The simple example program in Figure 1 takes a "secret" value as an input and returns the first 8 bits of the secret as the "observable" value. The correlation between the secret and observable values can be represented as a joint probability distribution between the two. Assuming that each secret value has the same probability than any other secret value to be an input (i.e., a uniform distribution over the secret domain), the joint distribution p_{XY} between the secret and the observable is as follows:

$$p(x, y)^{1} = \Pr(X = x, Y = y) = \begin{cases} 2^{-16} & \text{if } x_{0..7} = y_{0..7} \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where X and Y are the random variables for the secret and the observable, x and y are the bit-vector values of the secret and the observable, respectively, and $x_{0..7}$ is the first 8 bits of x. The initial uncertainty of the adversary about the secret value according to the Shannon entropy

$$H(X) = -\sum_{x \in \mathcal{X}} p_X(x) \log p_X(x) \tag{2}$$

is 16 bits, where $\mathcal{X}$ is the domain of the secret value, and $p_X(x) = \Pr(X = x)$ is the marginal probability of the secret value. When the adversary observes the observable value, the uncertainty of the adversary about the secret value regarding the conditional Shannon entropy

$$H(X \mid Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p_Y(y)} \tag{3}$$

is 8 bits, where $\mathcal{Y}$ is the domain of the observable value, and $p_Y(y) = \Pr(Y = y)$ is the marginal probability of the observable value. The reduction in the uncertainty about the secret value is the *mutual information* (MI) between the secret value and the observable values

$$I(X; Y) = H(X) - H(X \mid Y), \tag{4}$$

which is 8 bits. The MI represents the amount of information leakage from the program.

If the source code is available, the analytical approach can be considered to compute the MI. The program is transformed to the constraints with symbolic execution, and the number of inputs corresponding to every possible (secret, observable) pair is computed with the model counter to produce the MI [13]–[16]. Nonetheless, analytical methods are often intractable due to the path explosion problem for symbolic execution, due to restrictions on the type of constraints solved by the model counter, or due to the lack of empirical facts unavailable to a static analysis [6], [7].

On the other hand, the statistical approach treats the program as a black-box and estimates the MI from a limited number of samples of the program executions [8]–[10]. Given samples of (secret, observable) pairs, it applies the statistical model to estimate the MI either directly or from the estimated joint probability distribution. As it is agnostic to the complexity of the program and can analyze non-deterministic or probabilistic software without any additional effort, it has been

[1] We use the notation p instead of p_{XY} if the random variables of the distribution are clear from the context.

shown that statistical methods are often more scalable than the analytical methods [3], [11]. However, the current state-of-the-art statistical methods [9] require a sufficient number of samples for every possible $\langle$secret, observable$\rangle$ pair to estimate the MI accurately, which is a strong restriction in practice. We further explore the consequences of the missing event problem that arise from a small sample size in the following section.

B. Entropy Estimation in the Presence of Missing Events

Given a set of samples from an unknown multinomial distribution (MD), one can estimate the probability of each event from the empirical probability, i.e., X_i/n, where n is the total number of samples, and X_i is the frequency of the i-th event. However, in the presence of low-probability events in the MD, collecting a small number of samples may miss rare events. The probabilities of the undetected events are underestimated to zero, and the probabilities of the detected events are instead overestimated to compensate for the underestimation, which the following equation can explain:

$$\mathbb{E}\left[\hat{p}_i \mid X_i > 0\right] = \mathbb{E}\left[\frac{X_i}{n}\middle| x_i > 0\right] = \frac{p_i}{1-(1-p_i)^n} \tag{5}$$

where p_i and $\hat{p}_i$ are the true probability and empirical probability of the i-th event, respectively. Due to the bias, using the *empirical* Shannon entropy estimate $\hat{H}_{\text{emp}}$ underestimates, which is why the empirical MI estimate $\hat{I}_{\text{emp}}$ overestimates.

Miller and Madow [17] suggested a bias correction method for Shannon entropy estimation, which is known as the Miller estimator:

$$\hat{H}_{\text{Miller}} = \hat{H}_{\text{emp}} + \frac{m-1}{2n}, \tag{6}$$

where m is the number of the observed events, and $\hat{H}_{\text{emp}}$ is the empirical entropy. In the case of the MI between the random variable X and Y, the Miller estimator is applied as follows:

$$\hat{I}_{\text{Miller}} = \hat{I}_{\text{emp}} - \frac{(m_{\mathcal{X}}-1)(m_{\mathcal{Y}}-1)}{2n}, \tag{7}$$

where $m_{\mathcal{X}}$ and $m_{\mathcal{Y}}$ are the number of the observed events of the random variables X and Y, respectively, and $\hat{I}_{\text{emp}}$ is the empirical MI. The Miller estimator is used in the state-of-the-art statistical information leakage analysis methods [8], [9]. However, it is shown that if the number of samples is insufficient to observe the events, the bias correction of the Miller estimator overcompensates the underestimation of the entropy, which leads to the underestimation of the MI [11]. Regarding the information leakage analysis, the underestimation of the MI is especially *harmful*; estimating the MI as a smaller value than the actual MI can lead to overconfidence in the privacy of the vulnerable program.

C. Single-Variable MD Estimation

The *absence* of species (here, events) from the sample is a long-standing challenge in ecological biostatistics. Chao et al. [12] proposed a method that estimates the species distribution in the target assemblage from a limited number of samples from that assemblage. This unvieled species distribution represents our hidden multinomial distribution (MD).

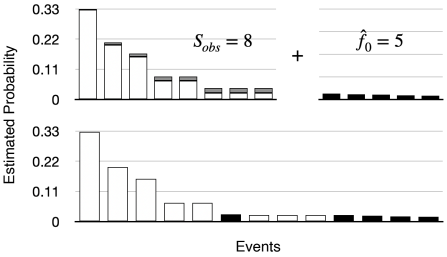


Fig. 2: Illustration of MD estimation. The upper left plot shows the empirical probabilities (white + grey) and the estimated probabilities (white) of the detected events. The upper right plot shows the estimated probabilities of the undetected events (black). The lower plot shows the merged distribution.

Given a set of samples from the hidden MD, Chao's MD estimation first computes the true probabilities of S_{obs} detected events, which is shown in the upper left plot of Figure 2. To get rid of the overapproximation (grey bars) of the empirical probabilities of the detected species (white + grey bars), it models the true probabilities of the detected events with two parameters, $\lambda > 0$ and $0 < \theta \leq 1$: for $X_i > 0$, $p_i \approx (X_i/n)(1 - \lambda e^{-\theta X_i})$. Then, it solves the system of two non-linear equations for the 1st and 2nd order sample coverage ${}^1C$ and ${}^2C$ to estimate the parameters $\hat{\lambda}$ and $\hat{\theta}$.

$${}^1C = \sum_{i \in \text{detected}} p_i, \quad {}^2C = \sum_{i \in \text{detected}} p_i^2. \tag{8}$$

The result gives the estimated probabilities $\bar{p}_i^{\text{det}}$ for each detected events (white bars).

To estimate the probabilities of the undetected events (upper right plot of Figure 2), it first estimates the number of undetected events $\hat{f}_0$ from the number of singleton events f_1 and the number of doubleton events f_2. Then, similar to the detected events, it models the true probabilities of the detected events with two parameters α and β: for $1 \leq i \leq \hat{f}_0$, $p_i^{\text{und}} = \alpha\beta^i$. It solves another system of two non-linear equations for the complement of 1st and 2nd order sample coverage $1 -^1 C$ and $1 -^2 C$ to estimate the parameters $\hat{\alpha}$ and $\hat{\beta}$. The result gives the set of estimated probabilities $\{\bar{p}_i^{\text{und}}\}_i$ of potential undetected events (black bars).

The probabilities of detected and undetected events combined and normalized by their sum, as shown in the lower plot of Figure 2, represents the estimated multinomial distribution (MD). The result of the MD estimation lessens the overestimation of the probabilities of the observed events and distributes those to the probabilities of the unobserved events of the unknown MD. Therefore, computing Shannon entropy from the estimated MD results in a larger entropy than the underestimated entropy of the empirical distribution. Note that our method only estimates the set of probabilities of the undetected events; it does not provide which undetected class has which probability. Also note that the running time of our method is dominated by the running time of the non-linear

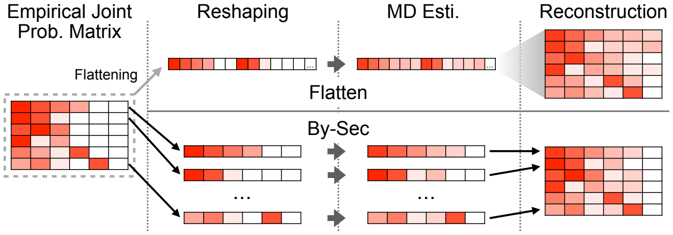


Fig. 3: Two approaches to estimate the joint distribution

optimization which is known to depend on the number of equations (here, 2) and the number of variables (here, 2).

III. Methodology

In this section, we propose a novel integration of the multinomial distribution (MD) estimation and mutual information (MI) estimation methods to reconstruct the joint distribution of the ⟨secret, observable⟩ pair. We first suggest two approaches to reconstruct the joint distribution: *Flatten* and *By-Secret*. Then, we propose novel refinement heuristics to estimate more rational joint distribution, which can improve the estimation accuracy. Finally, we design a new adjusted Miller-Madow bias correction method for the reconstructed joint distribution.

A. MD Estimation for MI

Chao's MD estimation, which may complement the bias of the Shannon entropy from the empirical distribution, considers the distribution of a single random variable. However, the MI is defined over the joint distribution of two random variables of the secret value X and the observable value Y. Thus, it is nontrivial to be applied to estimate the joint distribution $\hat{p}_{XY}$.

The key idea to employ the MD estimation for the MI estimation is to consider the joint distribution p_{XY} as one or more MDs of a single random variable(s). Figure 3 illustrates the two approaches, *Flatten* and *By-Secret*.

Algorithm 1: Flatten (**FL**) approach

```
   Input: $O = \{(x_i, y_i)\}_i$: An ordered set of samples
          $\mathcal{X}, \mathcal{Y}$: Sample spaces of the secret and the observable
          $isRefine$: A flag for refinement heuristics
   Output: $\hat{p}_{XY}$: The estimated joint distribution matrix
 1 $f_0^{\max} \leftarrow |\mathcal{X}| \times |\mathcal{Y}| - |\{(x, y) \mid (x, y) \in O\}|$
 2 $O_Z \leftarrow \{z_i\}_i$ s.t. $z_i = (x_i, y_i)$                    // Notation change
 3 $\bar{P}_{\text{det}}, \bar{P}_{\text{und}} \leftarrow \text{MD}(O_Z, f_0^{\max})$                    // MD estimation
 4 if $isRefine$ then
 5     $\bar{P}_{\text{det}}, \bar{P}_{\text{und}} \leftarrow \text{Reorder}(\bar{P}_{\text{det}}, \bar{P}_{\text{und}})$   // Refinement (Sec. III-B)
 6 $\hat{p}_{XY} \leftarrow \text{Reshape}(\bar{P}_{\text{det}}, |\mathcal{X}|, |\mathcal{Y}|)$      // Reconstruct the joint dist.
 7 for $j \leftarrow 1$ to $|\bar{P}_{und}|$ do
 8     $i_x, i_y \leftarrow \text{RndZeroIdx}(p_{XY})$
 9     $\hat{p}_{XY}[i_x, i_y] \leftarrow \bar{P}_{\text{und}}[j]$    // Assign the undetected probability
10 return $\hat{p}_{XY}$
```

Approach 1. Flatten (FL): The first approach is to consider the pair of the secret and the observable as a single random variable, which we call a flattening of the joint distribution. Given two sample spaces $\mathcal{X}$ and $\mathcal{Y}$ of the secret and the observable, respectively, we define the joint sample space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, i.e, $z = (x, y) \in \mathcal{Z}$, $x \in \mathcal{X}$, and $y \in \mathcal{Y}$. Then the probability of the event z is defined as $p(z) = \Pr(Z = z) = \Pr(X = x, Y = y) = p(x, y)$, and the probability distribution of a random variable Z over the sample space $\mathcal{Z}$ is a MD.

Algorithm 1 shows the pseudocode of the **FL** approach. A set of samples $O = \{(x_i, y_i)\}_i$ from the joint distribution p_{XY} is equivalent to a set of samples $\{z_i\}_i$ from the flattened distribution p_Z, where $z_i = (x_i, y_i)$ (line 2). Thus, the flattened distribution p_Z can be estimated by applying the MD estimation to $\{z_i\}_i$ (line 3). The result of the MD is adjusted probabilities $\bar{P}_{\text{det}}$ for each observed $\bar{p}_i^{\text{det}} = \hat{p}(z_i) = \hat{p}(x_i, y_i)$ ($(x_i, y_i) \in$ detected) and a set of estimated probabilities $\bar{P}_{\text{und}}$ of the unobserved events $\{\bar{p}_1^{\text{und}}, \cdots \bar{p}_{\hat{f}_0}^{\text{und}}\}$. During the MD estimation, $\hat{f}_0$ is bounded by the number of undetected joint events $f_0^{\max} = |\mathcal{X}| \times |\mathcal{Y}| -$ [# of detected (x,y)] (line 1).

To reconstruct the joint distribution $\hat{p}_{XY}$ from the estimated probabilities, the probabilities of the detected events $\bar{p}_i^{\text{det}}$ are assigned to the observed (x_i, y_i) pairs (line 6), and the probabilities of the undetected events $\bar{p}_j^{\text{und}}$ are assigned to the unobserved (x_j, y_j) pairs. The function Reshape (line 6) reshapes the ordered set of the probabilities of the detected events $\bar{P}_{\text{det}}$ to the joint distribution matrix $\hat{p}_{XY}$ with the size of $|\mathcal{X}| \times |\mathcal{Y}|$. For the undetected probability assignment, we randomly choose unobserved (x_j, y_j) and assign $\bar{p}_j^{\text{und}}$ for each probabilities in $\bar{P}_{\text{und}}$ (line 7-9). The function RndZeroIdx (line 8) returns a random index of the zero element in the given vector. Lines 4 and 5 describe the refinement heuristics, which will be explained in Section III-B.

Approach 2. By-Secret (BS): The second approach considers the conditional probability distribution of the observable given the secret value as an individual MD. It applies the MD estimation for each of the conditional probability distributions. Given one secret value $x \in \mathcal{X}$, the conditional probability distribution of the observable, $p(y \mid x) = \Pr(Y = y \mid X = x) = p(x, y)/p(x)$, becomes a MD with a single random variable Y over the sample space $\mathcal{Y}$.

Algorithm 2 shows the pseudocode of the **BS** approach. For each secret value $x' \in \{x \in \mathcal{X} \mid (x, y) \in O\}$, the MD estimation is applied to the sub-sample of the observed observable values $\{y_j\}_j$ where $(x_j, y_j) \in O \wedge x_j = x'$ (line 4-6, 9). The result of the MD is adjusted conditional probabilities $\bar{P}_{\text{det}}$ of $\bar{p}_i^{\text{det}} = \hat{p}(y_i \mid x')$ for each observed y_i with x' ($(x', y_i) \in O$) and a set of estimated conditional probabilities $\bar{P}_{\text{und}}$ of the unobserved events $\{\bar{p}_1^{\text{und}}, \cdots \bar{p}_{\hat{f}_0}^{\text{und}}\}$. Here, when estimating MD, we use the minimum value between the usual estimated f_0 and the number of detected observable but not with x', i.e., $|\{y \in \mathcal{Y} \mid (x', y) \notin O \wedge \exists x \in \mathcal{X} \text{ s.t., } (x, y) \in O\}|$, as $\hat{f}_0$ for the MD estimation (line 8).

The probabilities from MD estimation for each secret value x' are used only for the joint probability of the same secret value x' when reconstruction. $\bar{p}_i^{\text{det}} \times \hat{p}(x')$ is assigned to the observed (x', y_i) pairs (line 11), and $\bar{p}^{\text{und}} \times \hat{p}(x')$ is assigned to the unobserved (x', y) pairs; for the assignment, we randomly choose y such that $(x', y) \notin O \wedge \exists x \in \mathcal{X}$ s.t., $(x, y) \in O$

Algorithm 2: By-Secret (**BS**) approach

```
Input: $O = \{(x_i, y_i)\}_i$: An ordered set of samples
       $\mathcal{X}, \mathcal{Y}$: Sample spaces of the secret and the observable
       $isRefine$: A flag for refinement heuristics
Output: $\hat{p}_{XY}$: The estimated joint distribution matrix
1  $\mathcal{X}_O \leftarrow \{x \in \mathcal{X} \mid (x, y) \in O\}$                // Detected secrets
2  $\mathcal{Y}_O \leftarrow \{y \in \mathcal{Y} \mid (x, y) \in O\}$              // Detected observables
3  $\hat{p}_{XY} \leftarrow$ Zeros$(|\mathcal{X}|, |\mathcal{Y}|)$                             // Zero matrix
4  foreach $x' \in \mathcal{X}_O$ do
5      $i_x \leftarrow$ Idx$(x', \mathcal{X})$      // For each $x' \in \mathcal{X}$, get $y' \in Y$ observed
6      $O_{x'} \leftarrow \{y_j \mid (x_j, y_j) \in O \wedge x_j = x'\}_j$                // with $x'$
7      $\hat{p}(x') \leftarrow \frac{|O_{x'}|}{|O|}$                    // Empirical probability of $x'$
8      $f_0^{\max} \leftarrow |\mathcal{Y}_O| - |\{y \in \mathcal{Y}_O \mid (x', y) \notin O_{x'}\}|$
9      $\bar{P}_{\text{det}}, \bar{P}_{\text{und}} \leftarrow$ MD$(O_{x'}, f_0^{\max})$            // MD estimation for $x'$
10     if $\neg isRefine$ then
11         $\hat{p}_{XY}[i_x, :] \leftarrow \bar{P}_{\text{det}} \times \hat{p}(x')$   // Elementwise multiplication
12         for $j \leftarrow 1$ to $|\bar{P}_{und}|$ do
13             $i_y \leftarrow$ RndZeroIdxFrom$(p_{XY}[i_x, :], \mathcal{Y}_O)$
14             $\hat{p}_{XY}[i_x, i_y] \leftarrow \bar{P}_{\text{und}}[j] \times \hat{p}(x')$   // Assign und. prob.
15     else
           // Refinement heuristic (Sec. III-B)
16         $\bar{P}'_{\text{det}}, \bar{P}'_{\text{und}} \leftarrow$ Reorder$(\bar{P}_{\text{det}}, \bar{P}_{\text{und}})$
17         $\hat{p}_{XY}[i_x, :] \leftarrow \bar{P}'_{\text{det}} \times \hat{p}(x')$
18         for $j \leftarrow 1$ to $|\bar{P}'_{und}|$ do
19             $i_y \leftarrow$ PropZeroIdxFrom$(\hat{p}_{XY}[i_x, :], \mathcal{Y}_O, O)$
20             $\hat{p}_{XY}[i_x, i_y] \leftarrow \bar{P}'_{\text{und}}[j] \times \hat{p}(x')$   // Assign und. prob.
21 return $\hat{p}_{XY}$
```

(line 12-14). The function RndZeroIdxFrom (line 13) returns a random index of the observation from $\mathcal{Y}_O = \{y \in \mathcal{Y} \mid (x, y) \in O\}$, with zero probability in the given vector. Lines 15-19 describe the refinement heuristics presented in Section III-B.

After the estimation of the joint distribution, the MI is computed from the estimated joint distribution $\hat{p}_{XY}$ as Eq. (4).

Comparison between FL vs. BS: The advantage of the **BS** approach against the **FL** approach is that it assigns the estimated probability of the undetected events more precisely regarding the frequency of the observable values given the secret value. The intuition of Chao's MD estimation is that the number of undetected events is close to the number of detected rare events, and the frequencies of the detected rare events determine the probabilities of the undetected events. If there is no rare observable value regarding a secret value in the samples, MD estimation will predict that there is no undetected observable value regarding the secret value. In contrast, if many rare observable values are detected within a secret value, MD estimation will predict that there still remain many undetected observable values regarding the secret value. However, as the **BS** approach applies MD estimation for each secret value x_i, it may need more samples than the **FL** approach to apply MD estimation for each secret value.

Conversely, the **FL** approach applies MD estimation regarding all the samples, which may make MD estimation more accurate than the **BS** approach. Also, the **FL** approach can assign the estimated probability of the undetected events to the unobserved secret value, unlike the **BS** approach.

B. Refinement Heuristics

As the MD estimation does not provide which undetected class would have which estimated probability, the estimated probability of the undetected events has been randomly assigned to the unobserved (x, y) pairs without any guidance in the suggested approaches. Also, the MD estimation uses different models for the detected and undetected events, which may cause some of the estimated probabilities of the undetected events to be larger than the minimum estimated probability of the detected events, which is less likely to be true.

We propose a heuristic refinement of the adjustment to address those abnormalities of probability assignment. The refinement is based on the following hypothesis: 1) the probability of the detected secret-observable pair should not be smaller than the probability of the undetected secret-observable pair, and 2) the probability of assigning the estimated probability to the undetected secret-observable pair should be proportional to the marginal probability of the observable; in other words, the more probable observable regarding the general program execution should be more likely to be observed than the less probable observable. Notice that the second heuristic is only applicable to the **BS** approach as the **FL** approach flattens the secret-observable pair and does not provide the marginal probability of the observable.

Function Reorder (line 5 in Algorithm 1 and line 16 in Algorithm 2) implements the heuristic regarding the first hypothesis. It receives $\bar{P}_{\text{det}} = \{\bar{p}_i^{\text{det}}\}_i$, an ordered set of estimated probabilities for the detected events, and $\bar{P}_{\text{und}} = \{\bar{p}_j^{\text{und}}\}$, a multiset of estimated probabilities for the undetected events. Until the minimum probability of the detected events is larger than the maximum probability of the undetected events, it swaps the probability of the detected event and the probability of the undetected event; during the swapping, it preserves the order of the probabilities of the detected events.

Algorithm 3: Function PropZeroIdxFrom

```
Input: $p$: The current estimated probability vector
       $\mathcal{Y}_O$: A set of detected observable values
       $O$: A set of samples
Output: $j$: The index of the next assignment for the
        undetected event
$\mathcal{Y}_{zero} \leftarrow \{y \in \mathcal{Y} \mid p_y = 0\}$       // Get zero probability observables
// Construct a map from $y$ to marginal probability
$ProbMap \leftarrow \{y \in \mathcal{Y}_{zero} : \frac{|\{y_j \mid (x_j, y_j) \in O \wedge y_j = y\}|}{|O|}\}$
$j \leftarrow$ select from $\mathcal{Y}_{zero}$ with prob. $ProbMap$
return $j$
```

Algorithm 3 shows the pseudocode Function PropZeroIdxFrom (line 18 in Algorithm 2): the heuristic regarding the second hypothesis. It first identifies the candidate observable values $\mathcal{Y}_{zero}$ that have zero probability in the current estimated probability vector p. Then, it computes the marginal empirical probability of each candidate observable value $y \in \mathcal{Y}_{zero}$. Finally, it randomly selects an observable value $y \in \mathcal{Y}_{zero}$ with the probability proportional to the marginal empirical probability of y. The function RndZeroIdxFrom (line 13 in

Algorithm 2) is a special case of Function PropZeroIdxFrom, where the index is selected uniformly from $\mathcal{Y}_{zero}$.

C. Miller-Madow Bias Correction for Our Estimator

In addition to the refinement, we also consider applying the Miller-Madow bias correction to the estimated entropy. In Equation (7), the Miller-Madow bias correction term $\frac{(m_{\mathcal{X}}-1)(m_{\mathcal{Y}}-1)}{2n}$ depends on the total number of samples n. After applying the MD estimation, there may exist a non-zero probability smaller than $1/n$, the minimum non-zero empirical probability with n samples, in the estimated joint distribution. The reciprocal of the minimum non-zero probability $n' = \lceil 1/\min p(x, y)\rceil$ in the estimated joint distribution represents the resolution of the distribution; the distribution can exhibit what would have been the expected frequency of the event that occurs with n' samples. Regarding this, we use $\max(n, n')$ instead of n in the Miller-Madow bias correction in our estimator. The bias correction value is subtracted from the MI estimated from the reconstructed joint distribution $\hat{p}_{XY}$.

IV. Experimental Setup

A. Research Questions

We design four research questions to investigate our main hypothesis: whether accounting for missing events improves the information leakage estimate.

RQ1: How accurate and safe is the proposed method compared to existing statistical methods?

We measure *accuracy* using the mean-squared error (MSE), a standard measure of estimator performance, and *safety* as the frequency of underestimating the MI. Underestimating the MI can lead to undue confidence in the confidentiality of the data processed by the program and renders the estimator unreliable.

RQ2: How does each component of our proposed method affect the performance of the estimation?

We investigate the effect of each component on the performance of our proposed method. The components include: (a) the selected approach (Flatten [**FI**] or By-Secret [**BS**]), (b) with or without refinement heuristics, and (c) with or without the Miller-Madow bias correction.

RQ3: What is the timing cost of the proposed method?

Compared to the empirical MI estimation or Miller-Madow bias correction, which have negligible time overhead, the proposed method requires additional computation to estimate the MD. We investigate the timing cost of the proposed method and discuss the feasibility of the proposed method in practice.

RQ4: How does the proposed method perform on real-world applications for information leakage quantification?

We evaluate the proposed method in the practical context of information leakage quantification. We consider two real-world applications: location privacy and e-passport privacy.

B. Baseline and State-of-the-Art Estimators

We consider two baseline estimators: the empirical estimator and the Miller estimator, i.e., the empirical MI with Miller-Madow bias correction used in Leakwatch [8], [9].

We consider one state-of-the-art estimator, HyLeak, a *hybrid* estimation method that combines statistical and precise analysis [11]. Similar to our work, Biondi et al. [11] aimed to overcome the limitations of large sample sizes required for statistical methods. Their HyLeak approach relies on precise analysis for components that are impractical to analyze statistically. They suggest an automatic program decomposition and a set of heuristics to select components that have a large joint sub-distribution matrix. For such components, they use precise analysis, and for the others, they use statistical analysis. Then, the results are combined to estimate the MI of the program. The authors claimed that HyLeak is more scalable than the Miller estimator and the purely precise method.

C. Subjects and Design of Experiment

Benchmark Programs (Table I). For the first three research questions, we use the benchmarks from HyLeak [11] for our evaluation. The subjects cover a wide variety of programs, some of which operate probabilistically and exhibit non-trivial secret-observable mappings. More details can be found in the supplementary material. For every program, we choose between four to five variants (choices of N) for a total of 22 variants. For all variants, we choose four sample sizes for a total of 88 configurations. We consider the uniform distribution over the domain of secret values as in previous work [9], [11].

To determine the ground truth, at this scale, there exists no method to compute the MI precisely. Hence, we follow the approach typically followed in applied statistics. Given a program, we compute the ground truth MI between (secret, observable) from the empirical MI given a very large number (1M) of executions. This is more than one to five orders of magnitude larger than the number of samples for the estimation.

We consider the performance of the estimators for different numbers of sample program executions. Our evaluation considers $\times 0.5$, $\times 1$, $\times 2$, and $\times 5$ of the size of the observable domain ($\mathcal{Y}$) for each secret value ($x \in \mathcal{X}$) as the number of program executions. We call this relative number of executions as a *sample ratio*. The range covers starting from the natural situation when it is simply not possible to observe all the observable values for each secret value ($\times 0.5$) and up to a case when each secret value may have had enough opportunity to be observed with all the observable values ($\times 2$, $\times 5$). For each $N \in \{0.5, 1, 2, 5\} \cdot |\mathcal{X}| \cdot |\mathcal{Y}|$, we randomly sample N program executions and estimate the MI using the estimators.

Real-world Applications. To investigate how the proposed method performs in practice, we consider two real-world applications of information leakage quantification in **RQ4**.

Our first application is measuring information leakage from a *location privacy-preserving mechanism* (LPPMs) which is used to protect the user's location privacy (secret) by reporting

TABLE I: List of subject programs (from HyLeak [11])

Subject	$(\lvert\mathcal{X}\rvert, \lvert\mathcal{Y}\rvert)$	Variants (N)	Information Leakage
ProbTerm	$(N+1, 10\text{–}20)$	$\{5, 7, 9, 12\}$	A program implementing a loop that terminates after a certain number of iterations (observable). The termination condition for that loop is a probabilistic function on the secret value (secret).
RandomWalk	$(500, 24\text{–}40)$	$\{3, 5, 7, 14\}$	A robot control program leaks the final location of the controlled robot (observable). The adversary wants to guess the starting location (secret).
Reservoir	$(2^N, 2^{N/2})$	$\{4, 6, 8, 10, 12\}$	An implementation of reservoir sampling [18] that leaks the random sample of size $N/2$ (observable) chosen without replacement from a population of size N (secret).
SmartGrid	$(3^N, 12)$	$\{1, 2, 3, 4, 5\}$	A smart grid control program leaks whether the total consumption of all users exceed a cetain threshold (observable). The adversary can be one of the users and wants to guess the consumption of another user (secret).
Window	(N, N)	$\{20, 24, 28, 32\}$	A secret-avoiding random number generator which leaks the number chosen within a random range (observable). If the selected range includes the secret, a different range is chosen. The adversary wants to guess the secret.

the obfuscated location (observable) to the corresponding location-based service (LBS) [19]. Some vendors develop their own mechanisms [20], e.g., the Onion Router (Tor) network, while others use 3rd-party services [21]–[23]. It is important to quantify the information leakage of the LPPMs as there is a trade-off between the privacy and the utility of the location-based services. If the obfuscation is too strong, the utility of the LBS is lost, while if the obfuscation is too weak, the user's privacy is lost. Yet, it is difficult to analytically compute the leakage not only due to the accessability or complexity (Tor's LOC is ∼300K) but also due to the availability of additional information, like points of interest (e.g., shops, churches) or geographical characteristics of the area (e.g., roads, lakes).

For the evaluation, we implement two LPPMs: the planar Laplacian [24] and one of the optimal mechanisms proposed by Oya et al. [25]. We measure the information leakage of the LPPMs on the real location data from the Gowalla dataset [26], which contains users' checkins and their geographical location. The size of the secret ($\mathcal{X}$) and observable ($\mathcal{Y}$) domains are 400 and 115,600 locations, respectively. The distribution over the secret domain is based on Gowalla check-ins. Each LPPM defines the joint distribution of the secret and the observable, where we retrieve the ground-truth MI and the samples. As the joint space is vast, we consider the sample sizes of $2^i \cdot |\mathcal{X}|$ for $i \in \{0, 1, \cdots\}$ until the sample size reaches one million. More details can be found in the supplementary material.

Our second application is measuring information leakage from the *RFID chip on an e-passport*. The old e-passport protocol was vulnerable to a replay attack, where the processing time for the replayed message from the same passport was longer than the time for the message from a different passport [27], [28]. The observable is the time it takes to read and process the data from the e-passport, while the secret is a boolean value indicating whether the message is from the same passport or not. The MI between secret and observable quantifies the likelihood of an adversary successfully tracing a passport. By measuring the MI, researchers can quantify the information leakage of the e-passport protocol and develop a countermeasure to protect the e-passport from the attack [8].

We investigate how our proposed method performs to quantify the MI of the e-passport protocol. The dataset from the previous work [8] consists of the response time of the e-passport across three different countries both before and after the fix of the vulnerability. Given the data, we first reconstruct the underlying distribution by fitting the timing data to the normal distribution, and then we sample the timing data from the distribution. Similar to the benchmark programs, we consider the sample ratios of $\times 0.5$, $\times 1$, $\times 2$, and $\times 5$.

We evaluate our proposed method along with the baseline estimators, the empirical estimator, and the Miller estimator on the real-world applications. HyLeak cannot be applied to them since no source code is available which could be analyzed symbolically. For location privacy, the LPPM implementation is a probabilistic model obfuscating location information. For the passport protocol, we only have the binary outcome of passport verification and the time taken.

Infrastructure. Experiments were conducted in Ubuntu 20.04 docker containers running on an *AMD EPYC 7713P 64-Core* server with 256 GB of RAM. To mitigate the randomness, we repeat all the experiments 30 times for each (program, sample size) pair. For our estimator, we run 30 times the MD estimation and compute the average of the estimates. All data and scripts used in this paper are publicly available at

https://github.com/niMgnoeSeeL/ChaoMI.

D. Variables and Measures

To measure *estimator performance*, we use a standard metric, i.e., the *mean squared error (MSE)*. The MSE is computed as the average squared difference between the estimated MI and the ground truth MI. It measures the accuracy of the estimation methods in terms of both the mean bias and the variance of the produced estimates.

To measure *estimator safety*, we report the mean bias and the proportion of estimates larger than ground truth across the 88 configurations. We argue that a safe estimator should not systematically underestimate the true leakage; otherwise, we might wrongly believe that an attacker might need more resources to guess the secret than actually required. To facilitate a fair comparison of estimator performance (in terms of MSE and mean bias) across different subjects, we normalized the MI values by dividing them by the log of the secret domain size (i.e., the maximum MI). We used the Wilcoxon signed-rank test to check the statistical significance of the difference between the estimator performances.

To measure *cost effectiveness*, we calculate the time taken to estimate the MI for our estimator. As the analysis time for the empirical estimation and Miller-Madow bias correction

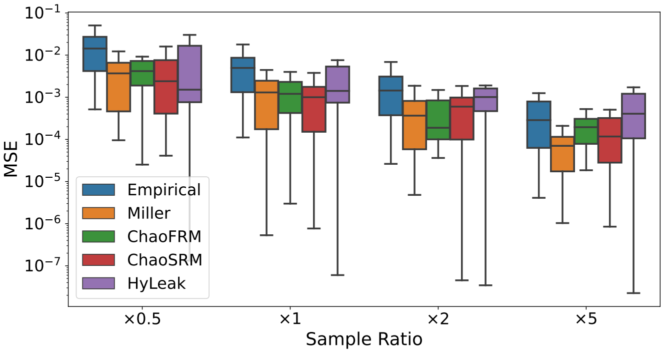


Fig. 4: MSE of the estimators across the configurations.

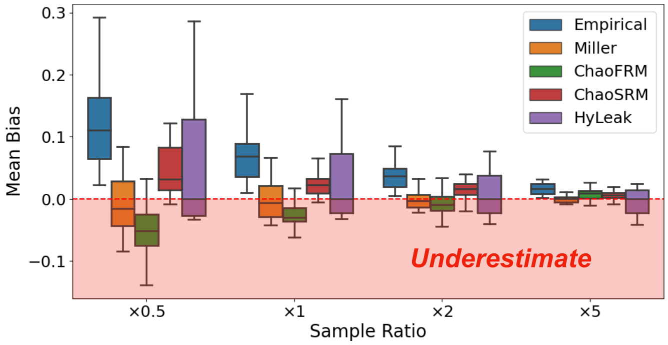


Fig. 5: Mean bias of the estimators across the configurations.

are negligible due to their simplicity, the measured time of our estimator can be considered as the extra analysis time required for the estimation. We investigate the affordability of the extra analysis time taken by our estimator. We exclude the time taken to sample the executions of the program from the efficiency evaluation as all the estimators use the same number of samples for the same sample size.

V. Experimental Results

RQ1: Estimator Performance and Safety

We named the eight variants of our mutual information (MI) estimator using Chao's multinomial distribution (MD) estimation as the following scheme: 'Chao' + 'F' (for the **FL** approach) or 'S' (for the **BS** approach) + 'O' (for the original MD estimates) or 'R' (for the refined MD estimates) + 'N' (for no bias correction) or 'M' (for Miller's bias correction).

Estimator performance. Figure 4 shows the distribution of the mean-squared error (MSE) for the five estimators: the empirical estimator (Empirical), Miller estimator (Miller), ChaoFRM, ChaoSRM, and HyLeak estimator. The result for all estimators is available in the replication package.

Our ChaoSRM and Miller estimators perform better than the empirical estimator. These differences are statistically significant with $p < 0.01$. Our ChaoFRM and HyLeak perform better than the empirical estimator, which is also supported by the statistical test, except for the sample ratio of $\times 5$.

Our ChaoSRM is the best performing estimator for the sample ratios $\times 1$ ($\mu_{\text{MSE}} = 1.44\text{e}-3$), while Miller is best performing for $\times 0.5$ ($\mu_{\text{MSE}} = 3.77\text{e}–3$), $\times 2$ ($\mu_{\text{MSE}} = 4.93\text{e}–4$) and $\times 5$ ($\mu_{\text{MSE}} = 9.52\text{e}–5$). However, the differences between ChaoSRM, ChaoFRM, and Miller are *not* significant for $\times 0.5$, $\times 1$, and $\times 2$ with p-$value > 0.05$. Miller performs better than our ChaoSRM and ChaoFRM for $\times 5$ *with* significance. HyLeak's performance depends significantly on the success of its heuristics decomposing the software and deciding which will be analyzed either with precise analysis or with sampling. It showed the smallest MSE for subjects ProbTerm and RandomWalk but worse performance than other estimators for the other subjects, which resulted in a bad performance overall. This implies that the HyLeak estimator is not robust to the target program and the sample size.

Estimator safety. While the performance of the three estimators, Miller, ChaoFRM, and ChaoSRM does not differ much, there are significant differences in the safety of the estimators. Figure 5 shows the distribution of the mean bias of the estimators across the configurations. The empirical estimator always overestimates the true MI as theoretically expected. ChaoSRM also produces safe overestimates in most of the configurations; it underestimates only 7 (8%) configurations, while it is always more accurate than the empirical estimator. In contrast, each Miller estimator and HyLeak estimator underestimates the true MI in 50 (57%) and 51 (58%) configurations, respectively. The ChaoFRM estimator does not help to avoid the underestimation of the MI; it underestimates 59 (67%) configurations.

We further analyze how the proportion of undetectable events affect the underestimation of the estimators. In case of the Miller estimator, it underestimates 75% and 100% of the configurations if the proportion of undetectable events is more than 45% and 75%, respectively. In contrast, the ChaoSRM estimator always safely overestimates the MI if the proportion of undetectable events is less than 75% except for a single configuration, and it safely overestimates 50% of the configurations even if more than 75% of the ⟨secret, observable⟩ pairs are undetectable.

In summary, our ChaoSRM estimator is the best estimator in terms of both safety and accuracy; its average MSE is the smallest when sample size is small, and it safely overestimates the MI for most of the configurations. The Miller estimator performs best when sample size is large. Yet, it often unsafely underestimates the MI if the program has undetectable ⟨secret, observable⟩ pairs, even with a small proportion.

RQ2: Component-wise In-depth Analysis

Empirical vs. Chao: The first three rows of Table II show the effect of estimating the probabilities of the missing events using the MD estimation. The result shows that, without the refinement and the bias correction, both the **FL** (ChaoFON) and the **BS** (ChaoSON) approach produce a smaller MI than the empirical MI for more than half of the configurations, which leads to a lower MSE. This implies that the proposed estimators successfully are more accurate than the overestimating empirical estimator using the MD estimation. Between the **FL** and the **BS** approach, there is no significant difference in

Esti-A	Esti-B	#(MI$_A \geq$ MI$_B$)	#(MSE$_A \geq$ MSE$_B$)	p-value
Empirical	ChaoFON	58	56	2e-4
Empirical	ChaoSON	69	69	<1e-4
ChaoFON	ChaoSON	43	46	0.60
Empirical	ChaoFRN	61	60	<1e-4
Empirical	ChaoSRN	88	88	<1e-4
ChaoFON	ChaoFRN	58	65	<1e-4
ChaoSON	ChaoSRN	65	65	<1e-4

TABLE II: The number of configurations matching the condition regarding the MI and the MSE of the estimators out of 88 configurations. The p-value is from the one-sided Wilcoxon signed-rank test comparing the MSE of the estimators.

	y_1	y_2	y_3	y_4
x_1	0.45	0.05	0.	0.
x_2	0.2	0.25	0.05	0.

(a) Empirical

	y_1	y_2	y_3	y_4
x_1	0.44	0.02	0.04	0.
x_2	0.2	0.25	0.05	0.

(b) Estimated MD 1

	y_1	y_2	y_3	y_4
x_1	0.445	0.04	0.	0.015
x_2	0.2	0.25	0.05	0.

(c) Estimated MD 2

(a) $MI(X;Y) = 0.226$
(b) $MI(X;Y) = 0.234$
(c) $MI(X;Y) = 0.256$

(d) MI

Fig. 6: Illustration: how MD estimation affects MI estimation

terms of the MI and the MSE. We can assume that the pros and cons of the **FL** and the **BS** approach discussed in Section III-A did not overwhelm each other in the experiment.

Effect of the refinement: The last four rows of Table II show the effect of the refinement. For both the **FL** and the **BS** approaches, the refinement reduces the MI and the MSE, helping to produce more accurate estimates for majority of the configurations. It is noticeable that the ChaoSRN estimator produces a smaller or equal MI/MSE than the empirical MI/MSE for all configurations. This implies that the refinement successfully improves the accuracy of the MI estimation.

We investigate how the refinement affects the MI estimation. Figure 6 shows two examples of how the MD estimation *without the refinement* could *increase* the MI estimate compared to the original empirical probability distribution. The original empirical probability distribution shown in Figure 6a has an MI of 0.226. If the estimated probability of the undetected event becomes larger than some estimated probability of the detected event ((x_1, y_3) vs. (x_1, y_2) in Figure 6b), the MI estimation can increase than before (0.234). Also, if the decreases of the discovery probabilities of the observed events are distributed to the undetected event (x_i, y_i) whose y_i is less observed in any x_i (y_4 in Figure 6c), the MI after the adjustment increases than before (0.256) as such an assignment increase the discriminability of Y given X.

Effect of the bias correction: Table III shows the effect of the Miller-Madow bias correction for empirical estimator and the modified bias correction for the proposed estimator. It shows that the amount of the decrease in the empirical MI is larger than the decrease in our estimators; the difference is similar for the **FL** approach but significant for the **BS** approach. As a result, while the bias correction makes the empirical estimator more accurate for 72/88 configurations, the huge decrease in the MI makes it underestimates the MI; thus, the underestimation happens for more than half (50/88) of the number of configurations. In contrast, the bias correction of our estimator with the **BS** approach makes the estimator not only more accurate but also keeps the safety; the estimator becomes more accurate for 80/88 configurations, while only seven configurations become underestimated. The bias correction of our estimator with the **FL** approach does not help significantly to improve the accuracy and the safety; it becomes more accurate for 52/88 configurations, while it creates 52 more underestimations.

TABLE III: Effect of Miller-Madow bias correction. Column Δ_{MI} and shows the average decrease in MI. $\Delta_{\text{u.e.}}$ shows the difference of the number of underestimated configurations (Esti-A$-$Esti-B). The fifth column shows the number of configurations where Esti-A has lower MSE than Esti-B, and the p-value is the one-sided Wilcoxon signed-rank test with MSEs.

Esti-A	Esti-B	Δ_{MI}	$\Delta_{\text{u.e.}}$	#(MSE$_A \geq$ MSE$_B$)	p-value
Empirical	Miller	0.35	50	72	<1e-4
ChaoFRN	ChaoFRM	0.24	52	52	0.0817
ChaoSRN	ChaoSRM	0.06	7	80	<1e-4

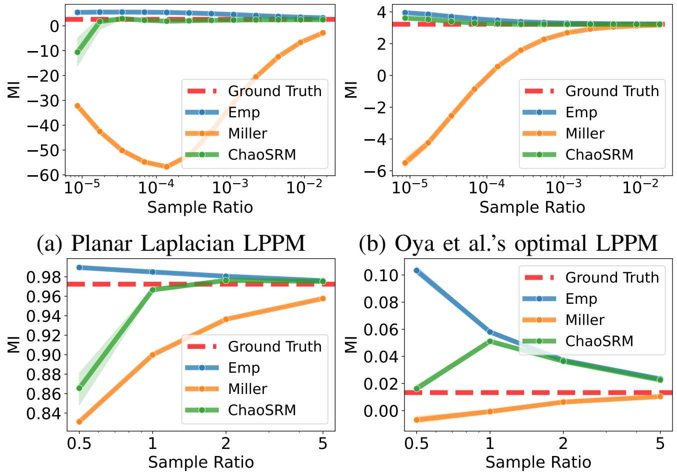
In summary, each component of the proposed estimator contributes to the safety and accuracy of the estimator. Both **BS** and **FL** approach are effective in terms of the accuracy, which is further improved by the refinement. The Miller-Madow bias correction is effective in terms of the safety and accuracy when it is applied to the **BS** approach.

RQ3: Cost Effectiveness

We find that the time taken to compute our estimates negligibly affected (Pearson correlation coefficient (PCC) < 0.1) by the sample ratio compared to the size of the joint sample space (PCC ≈ 0.97). The time taken by the **FL** approach is generally higher than that taken by the **BS** approach. Without the refinement or the bias correction, the average time across the configuration taken by the **FL** approach is 326.2s, while the one with the **BS** approach is 94.7s for the sample ratio of $\times 1$. This is because the size of the non-linear equation gets larger by the size of the joint sample space for the **FL** approach, while, for the **BS** approach, the size of the linear equation gets larger by the size of the secret space and the number of equations to solve is relative to the size of the observable space. The regression analysis shows that the time taken by ChaoSRM is linearly proportional to the size of the joint sample space. The slope of the regression line is $6.94e$–03, with an $R^2 = 0.98$. The refinement process takes non-negligible time (roughly $\times 3.5$) for the **FL** approach but not for the **BS** approach (less than $\times 0.1$) for the same reason. The bias correction has negligible effect on the time taken.

RQ4: Evaluation on the Practical Scenarios

Location privacy. Figure 7a and 7b shows the information leakage estimates for the two LPPM implementations. Our



(a) Planar Laplacian LPPM

(b) Oya et al.'s optimal LPPM

(c) British passport, unfixed

(d) British passport, fixed

Fig. 7: MI estimation for the practical application.

ChaoSRM estimator consistently outperforms all other estimators and produces safer MI estimates than Miller which consistently underestimates. The maximum sample size considered is $2^{11} \cdot |\mathcal{X}| = 819,200$, which is $\sim$1.7% of the size of $|\mathcal{X}| \cdot |\mathcal{Y}|$. This means, the domain of the observables ($|\mathcal{Y}|$) is substantially larger than the previous benchmarks programs; 115,600 locations can be observed in the location privacy application. As we can see, Miller significantly underestimates (even an impossible negative mutual information) due to its large bias correction term, which overclaims the safety of LPPMs. In contrast, the ChaoSRM estimator consistently provides more accurate, i.e., less overestimating, estimates compared to the empirical estimator. Knowing the LPPM's MI with less overestimation can provide the better quality of the location-based services as it can avoid over-obfuscating the location for the privacy. Compared to the empirical estimator, ChaoSRM has 1.89-10.88× lower MSE for the optimal algorithm and 1.29-7.56× lower MSE for the planar algorithm, except for the smallest sample size.

Passport privacy. Figure 7c and 7d illustrates the estimation for the required computational resources for a successful timing attack against the British passport RFID protocol. Roughly 1,200-2,000 different timing values are observed in the timing attack application. While the MI of unfixed protocol is close to 1 (left), showing the high information leakage, it becomes close to 0, meaning that guessing for the passport trace is half-and-half, after the fix (right).

Just like for the location privacy application, the Miller estimator significantly underestimates the MI for the unfixed protocol, leading to the overclaim of the safety of the protocol, while the ChaoSRM estimator provides the most accurate estimates with sample ratio $\geq$ 1. The ChaoSRM estimator always overestimates less and is more accurate than the empirical estimator for the fixed protocol. Accurate estimation of the MI can avoid the cost of replacing the protocol and losing the trust from the users. The results for other passport protocols are similar and presented in the supplementary material. ChaoSRM is always better, up to 13×, than the empirical estimator for the fixed passport protocol. Only when $\langle$Unfix, sample ratio$=$ 0.5$\rangle$ configuration is the empirical estimator better than ChaoSRM. However, ChaoSRM still estimates significant MI, indicating leakage in the passport protocol.

To summarize, our proposed ChaoSRM estimator is especially more practical for the real-world applications, where the domain of observables can be very large.

VI. Threats to Validity

External validity concerns whether the results from the study can be generalized. To mitigate this concern, we use various programs used in the previous study [11] with multiple variants of different sizes and complexity. While it cannot guarantee the generalizability of our work to every program, it provides a fair comparison with the previous work and is a good starting point for future work. *Internal validity* concerns the degree of confidence of our study, having not been influenced by any factor beyond the scope of the study. To mitigate the randomness of the experiment, we repeat the estimation 30 times for each configuration and conduct the statistical test in the evaluation. To avoid missing any potential error in our evaluation and to facilitate the reproduction of our study, we make our scripts and data publicly available.

VII. Related Work

Quantifying Information Leakage. In recent decades, there have been many works on quantifying information leakage. A larger number of works use information theoretic approaches to measure information leakage. As in this paper, several works measure the information leakage by estimating the mutual information (MI) between the input and the output of the program [8]–[11], [29], [30]. Other works consider different measures to quantify the information leakage, such as the channel capacity [5], [9], [31], Shannon entropy [2], [3], [32], [33], or g-leakage [7].

Many of the works [2]–[5] employ model counting techniques [13]–[16] to compute the information leakage in a white-box manner. As model counting-based techniques inevitably face scalability issues due to limitations of the constraint extraction and the solver, existing literature computes a bound on the information leakage [3], uses approximate model counting techniques [5], or combines with sampling-based techniques [11]. In this work, we solely focus on the black-box approach to estimate the MI and compare the performance with the state-of-the-art black-box and hybrid approaches.

MI Estimation. There are several approaches to estimating the Shannon entropy and MI from the limited number of samples besides the empirical MI or Miller-Madow method. Grassberger [34] proposed a method by exchanging the logarithmic function for a scalar function represented with the digamma function to avoid the bias. Jackknife [35], [36], the resampling method to reduce the bias of the estimator, has also been used to estimate the MI. Bayesian reasoning for entropy [37], [38] is to specify a model relating the observed events to the unknown quantity, then compute the posterior distribution over the entropy. Our method is orthogonal to these previous methods; we approximate the joint probability distribution itself.

Combining Biostatistics Methods for Software Testing/Security. Recently, there have been several works that combine biostatistical estimation methods with software testing and security. The key property of such biostatistical methods is the treatment of the unobserved events. For instance, the missing mass estimation [39], [40] estimates the probability of finding new species (or events) in an assemblage (or sample space) from the observed data. These methods have been employed in various dynamic software analysis tasks to solve the fundamental problem of dynamic software analysis, i.e., the missing events. For instance, Böhme et al. [41] estimate the residual risk, i.e., the probability that the next generated input is the first bug-revealing input, in a greybox fuzzing campaign using the Good-Turing [39] estimator, which estimates the probability of finding new species in an assemblage. Lee et al. [42] further extend the Good-Turing estimator to consider the structure of the given program to estimate the probability of reaching a certain, currently *unreached* statement in the program. Liyanage et al. [43] approximated the number of coverable program elements by a fuzzer using various species richness estimators [44].

VIII. Discussion

In this paper, we propose a statistical program analysis to analyze the privacy of a software system. Only by making public observations about a software system, what can attacker tell about its secrets? More formally, what is the mutual information (MI) between the random variable representing the secret and that representing the observable? Given a small sample of $\langle$secret, observable$\rangle$-pairs, we propose an improved methodology to estimate the MI for a software system that is both more safe and more accurate than existing methods according to our experiments.

We focus particularly on the small sample regime because in practice we cannot assume that we can draw samples so large that they contain all $\langle$secret, observable$\rangle$-pairs. In practice, a large proportion of pairs will be missing from the sample due to time or resource constraints during sampling. For instance, the time between samples [45] or getting a sufficient number of samples may be prohibitive to cover the domain of secrets or observables with a certain degree. Hence, reducing the estimation error due to these missing pairs has been the main challenge we sought to address in this paper.

With our methodology, a developer can evaluate and attempt to reduce the leakage of private information from their program without any restrictions on the size or language of the program. In fact, as a statistical program analysis [42], [46], our methodology works for software systems of arbitrary size and composition. In fact, apart from the time spent getting the samples, the running time of our methdology depends only on the time complexity of the non-linear optimization which depends on the number of equations and variables (both fixed at two for our estimation problem). In our experiment, we used MINPACK's hybrd/hybrj algorithms from the scipy library and observed the time to be linear in the size of the sample space.

Acknowledgment

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence strategy - EXC2092 - Project#390781972 and by the Max Planck Institute for Security and Privacy internship program. We thank the anonymous reviewers for their valuable comments and suggestions.

References

[1] Y. Noller and S. Tizpaz-Niari, "Qfuzz: Quantitative fuzzing for side channels," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2021. New York, NY, USA: Association for Computing Machinery, 2021, pp. 257–269. [Online]. Available: https://doi.org/10.1145/3460319.3464817

[2] L. Bang, A. Aydin, Q.-S. Phan, C. S. Păsăreanu, and T. Bultan, "String analysis for side channels with segmented oracles," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016. New York, NY, USA: Association for Computing Machinery, 2016, pp. 193–204. [Online]. Available: https://doi.org/10.1145/2950290.2950362

[3] S. Saha, S. Ghentiyala, S. Lu, L. Bang, and T. Bultan, "Obtaining information leakage bounds via approximate model counting," *Proc. ACM Program. Lang.*, vol. 7, no. PLDI, jun 2023. [Online]. Available: https://doi.org/10.1145/3591281

[4] F. Biondi, A. Legay, and J. Quilbeuf, "Comparative analysis of leakage tools on scalable case studies," in *International SPIN Workshop on Model Checking of Software*. Springer, 2015, pp. 263–281.

[5] F. Biondi, M. A. Enescu, A. Heuser, A. Legay, K. S. Meel, and J. Quilbeuf, "Scalable approximation of quantitative information flow in programs," in *Verification, Model Checking, and Abstract Interpretation: 19th International Conference, VMCAI 2018, Los Angeles, CA, USA, January 7-9, 2018, Proceedings 19*. Springer, 2018, pp. 71–93.

[6] G. Cherubin, K. Chatzikokolakis, and C. Palamidessi, "F-bleau: Fast black-box leakage estimation," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 835–852.

[7] M. Romanelli, K. Chatzikokolakis, C. Palamidessi, and P. Piantanida, "Estimating g-leakage via machine learning," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 697–716. [Online]. Available: https://doi.org/10.1145/3372297.3423363

[8] T. Chothia, Y. Kawamoto, and C. Novakovic, "A tool for estimating information leakage," in *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*. Springer, 2013, pp. 690–695.

[9] ——, "Leakwatch: Estimating information leakage from java programs," in *Computer Security-ESORICS 2014: 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II 19*. Springer, 2014, pp. 219–236.

[10] Y. Yuan, Z. Liu, and S. Wang, "Cacheql: Quantifying and localizing cache side-channel vulnerabilities in production software," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023.

[11] F. Biondi, Y. Kawamoto, A. Legay, and L.-M. Traonouez, "Hybrid statistical estimation of mutual information and its application to information flow," *Formal Aspects of Computing*, vol. 31, pp. 165–206, 2019.

[12] A. Chao, T. C. Hsieh, R. L. Chazdon, R. K. Colwell, and N. J. Gotelli, "Unveiling the species-rank abundance distribution by generalizing the good-turing sample coverage theory." *Ecology*, vol. 96 5, pp. 1189–201, 2015.

[13] A. Aydin, L. Bang, and T. Bultan, "Automata-based model counting for string constraints," in *International Conference on Computer Aided Verification*. Springer, 2015, pp. 255–272.

[14] A. Aydin, W. Eiers, L. Bang, T. Brennan, M. Gavrilov, T. Bultan, and F. Yu, "Parameterized model counting for string and numeric constraints," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 400–410.

[15] Y. Noller, "Model counting of string constraints for probabilistic symbolic execution," Master's thesis, University of Stuttgart, 2016.

[16] A. I. Barvinok, "A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed," *Mathematics of Operations Research*, vol. 19, no. 4, pp. 769–779, 1994.

[17] G. Miller, "Note on the bias of information estimates," *Information theory in psychology: Problems and methods*, 1955.

[18] J. S. Vitter, "Random sampling with a reservoir," *ACM Trans. Math. Softw.*, vol. 11, no. 1, pp. 37–57, mar 1985. [Online]. Available: https://doi.org/10.1145/3147.3165

[19] H. Jiang, J. Li, P. Zhao, F. Zeng, Z. Xiao, and A. Iyengar, "Location privacy-preserving mechanisms in location-based services: A comprehensive survey," *ACM Comput. Surv.*, vol. 54, no. 1, jan 2021. [Online]. Available: https://doi.org/10.1145/3423165

[20] D. P. Team, "Learning with privacy at scale," Apple Machine Learning Research, Tech. Rep., December 2017. [Online]. Available: https://machinelearning.apple.com/research/learning-with-privacy-at-scale

[21] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 1–18, 2008.

[22] K. Vu, R. Zheng, and J. Gao, "Efficient algorithms for k-anonymous location privacy in participatory sensing," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 2399–2407.

[23] S. Zhang, G. Wang, M. Z. A. Bhuiyan, and Q. Liu, "A dual privacy preserving scheme in continuous location-based services," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4191–4200, 2018.

[24] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: differential privacy for location-based systems," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 901–914. [Online]. Available: https://doi.org/10.1145/2508859.2516735

[25] S. Oya, C. Troncoso, and F. Pérez-González, "Back to the drawing board: Revisiting the design of optimal location privacy-preserving mechanisms," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 1959–1972. [Online]. Available: https://doi.org/10.1145/3133956.3134004

[26] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 1082–1090. [Online]. Available: https://doi.org/10.1145/2020408.2020579

[27] T. Chothia and V. Smirnov, "A traceability attack against e-passports," in *Financial Cryptography and Data Security*, R. Sion, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 20–34.

[28] T. Chothia and A. Guha, "A statistical test for information leaks using continuous mutual information," in *2011 IEEE 24th Computer Security Foundations Symposium*, 2011, pp. 177–190.

[29] J. Newsome, S. McCamant, and D. Song, "Measuring channel capacity to distinguish undue influence," in *Proceedings of the ACM SIGPLAN Fourth Workshop on Programming Languages and Analysis for Security*, 2009, pp. 73–85.

[30] F. Biondi, Y. Kawamoto, A. Legay, and L.-M. Traonouez, "Hyleak: hybrid analysis tool for information leakage," in *Automated Technology for Verification and Analysis: 15th International Symposium, ATVA 2017, Pune, India, October 3–6, 2017, Proceedings 15*. Springer, 2017, pp. 156–163.

[31] C. G. D. Val, M. A. Enescu, S. Bayless, W. Aiello, and A. J. Hu, "Precisely measuring quantitative information flow: 10k lines of code and beyond," *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 31–46, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:16644725

[32] B. Köpf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. New York, NY, USA: Association for Computing Machinery, 2007, pp. 286–296. [Online]. Available: https://doi.org/10.1145/1315245.1315282

[33] Q.-S. Phan, L. Bang, C. S. Pasareanu, P. Malacaria, and T. Bultan, "Synthesis of adaptive side-channel attacks," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, 2017, pp. 328–342.

[34] P. Grassberger, "Entropy estimates from insufficient samplings," *arXiv preprint physics/0307138*, 2003.

[35] S. Zahl, "Jackknifing an index of diversity," *Ecology*, vol. 58, no. 4, pp. 907–913, 1977. [Online]. Available: http://www.jstor.org/stable/1936227

[36] X. Zeng, Y. Xia, and H. Tong, "Jackknife approach to the estimation of mutual information," *Proceedings of the National Academy of Sciences*, vol. 115, no. 40, pp. 9956–9961, 2018. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1715593115

[37] D. H. Wolpert and D. R. Wolf, "Estimating functions of probability distributions from a finite set of samples," *Physical Review E*, vol. 52, no. 6, p. 6841, 1995.

[38] I. Nemenman, F. Shafee, and W. Bialek, "Entropy and inference, revisited," *Advances in neural information processing systems*, vol. 14, 2001.

[39] I. J. Good, "The population frequencies of species and the estimation of population parameters," *Biometrika*, vol. 40, no. 3/4, pp. 237–264, 1953. [Online]. Available: http://www.jstor.org/stable/2333344

[40] S. Lee and M. Böhme, "How much is unseen depends chiefly on information about the seen," 2024. [Online]. Available: https://arxiv.org/abs/2402.05835

[41] M. Böhme, D. Liyanage, and V. Wüstholz, "Estimating residual risk in greybox fuzzing," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, pp. 230–241. [Online]. Available: https://doi.org/10.1145/3468264.3468570

[42] S. Lee and M. Böhme, "Statistical reachability analysis," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ser. ESEC/FSE*, vol. 2023, 2023, p. 12.

[43] D. Liyanage, M. Böhme, C. Tantithamthavorn, and S. Lipp, "Reachable coverage: Estimating saturation in fuzzing," in *Proceedings of the 45th International Conference on Software Engineering*, ser. ICSE '23, 2023, pp. 1–13.

[44] A. Chao and R. K. Colwell, "Thirty years of progeny from chao's inequality: Estimating and comparing richness with incidence data and incomplete sampling," *SORT-Statistics and Operations Research Transactions*, pp. 3–54, 2017.

[45] A. G. Clark, M. Foster, B. Prifling, N. Walkinshaw, R. M. Hierons, V. Schmidt, and R. D. Turner, "Testing causality in scientific modelling software," 2022. [Online]. Available: https://arxiv.org/abs/2209.00357

[46] M. Böhme, "Statistical reasoning about programs," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 76–80. [Online]. Available: https://doi.org/10.1145/3510455.3512796