Hybrid Statistical Estimation of Mutual Information and its Application to Information Flow

Fabrizio Biondi, Yusuke Kawamoto, Axel Legay, Louis-Marie Traonouez

Under consideration for publication in Formal Aspects of Computing

Hybrid Statistical Estimation of Mutual Information and its Application to Information Flow

Fabrizio Biondi[1], Yusuke Kawamoto[2], Axel Legay[3], Louis-Marie Traonouez[3]

[1]CentraleSupélec Rennes, France. Email: fabrizio.biondi@inria.fr
[2]AIST, Japan. Email: yusuke.kawamoto@aist.go.jp
[3]Inria, France. Email: axel.legay@inria.fr, louis-marie.traonouez@inria.fr

Abstract. Analysis of a probabilistic system often requires to learn the joint probability distribution of its random variables. The computation of the exact distribution is usually an exhaustive *precise analysis* on all executions of the system. To avoid the high computational cost of such an exhaustive search, *statistical analysis* has been studied to efficiently obtain approximate estimates by analyzing only a small but representative subset of the system's behavior. In this paper we propose a *hybrid statistical estimation method* that combines precise and statistical analyses to estimate mutual information, Shannon entropy, and conditional entropy, together with their confidence intervals. We show how to combine the analyses on different components of the system with different accuracy to obtain an estimate for the whole system. The new method performs weighted statistical analysis with different sample sizes over different components and dynamically finds their optimal sample sizes. Moreover it can reduce sample sizes by using prior knowledge about systems and a new *abstraction-then-sampling* technique based on qualitative analysis. To apply the method to the source code of a system, we show how to decompose the code into components and to determine the analysis method for each component by overviewing the implementation of those techniques in HyLeak tool. We demonstrate with case studies that the new method outperforms the state of the art in quantifying information leakage.

Keywords: Mutual information; Statistical estimation; Hybrid method; Confidence interval; Statistical model checking

1. Introduction

In modeling and analyzing software and hardware systems, the statistical approach is often useful to evaluate quantitative aspects of the behaviors of the systems. In particular, probabilistic systems with complicated internal structures can be approximately and efficiently modeled and analyzed. For instance, statistical model checking has widely been used to verify quantitative properties of many kinds of probabilistic systems [LDB10].

The *statistical analysis* of a probabilistic system is usually considered as a black-box testing approach in which the

Correspondence and offprint requests to: Yusuke Kawamoto, AIST Tsukuba Central 1, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8560 JAPAN. Fabrizio Biondi, Inria Rennes, Campus de Beaulieu, 263 Avenue Général Leclerc, 35000 Rennes, France.

analyst does not require prior knowledge of the internal structure of the system. The analyst runs the system many times and records the execution traces to construct an approximate model of the system. Even when the formal specification or precise model of the system is not provided to the analyst, statistical analysis can be directly applied to the system if the analyst can execute the black-box implementation. Due to this random sampling of the systems, statistical analysis provides only approximate estimates. However, it can evaluate the precision and accuracy of the analysis for instance by providing the confidence intervals of the estimated values.

One of the important challenges in statistical analysis is to estimate entropy-based properties in probabilistic systems. For example, statistical methods [CCG10, CKN13, CKNP13, CKN14, BP14] have been studied for *quantitative information flow analysis* [CHM01, KB07, Mal07, CPP08], which estimates an entropy-based property to quantify the leakage of confidential information in a system. More specifically, the analysis estimates *mutual information* or other properties between two random variables on the secrets and on the observable outputs in the system to measure the amount of information that is inferable about the secret by observing the output. The main technical difficulties in the estimation of entropy-based properties are

1. to efficiently compute large matrices that represent probability distributions, and
2. to provide a statistical method for correcting the bias of the estimate and computing a confidence interval to evaluate the accuracy of the estimation.

To overcome these difficulties we propose a method for statistically estimating mutual information, one of the most popular entropy-based properties. The new method, called *hybrid statistical estimation method*, integrates black-box statistical analysis and white-box *precise analysis*, exploiting the advantages of both. More specifically, this method employs some prior knowledge on the system and performs precise analysis (e.g., static analysis of the source code or specification) on some components of the system. Since precise analysis computes the exact sub-probability distributions of the components, the hybrid method using precise analysis is more accurate than statistical analysis alone.

Moreover, the new method can combine multiple statistical analyses on different components of the system to improve the accuracy and efficiency of the estimation. This is based on our new theoretical results that extend and generalize previous work [Mod89, Bri04, CCG10] on purely statistical estimation. As far as we know this is the first work on a hybrid method for estimating entropy-based properties and their confidence intervals.

To illustrate the method we propose, Fig. 1 presents an example of a joint probability distribution P_{XY} between two random variables X and Y, built up from 3 overlapping components S_1, S_2 and T. To estimate the full joint distribution P_{XY}, the analyst separately computes the joint sub-distribution for the component T by precise analysis, estimates those for S_1 and S_2 by statistical analysis, and then combines these sub-distributions. Since the statistical analysis is based on the random sampling of execution traces, the empirical sub-distributions for S_1 and S_2 are different from the true ones, while the sub-distribution for T is exact. From these approximate and precise sub-distributions, the proposed method can estimate the mutual information for the entire system and evaluate its accuracy by providing a confidence interval. Owing to the combination of different kinds of analyses (with possibly different parameters such as sample sizes), the computation of the bias and confidence interval of the estimate is more complicated than the previous work on statistical analysis.

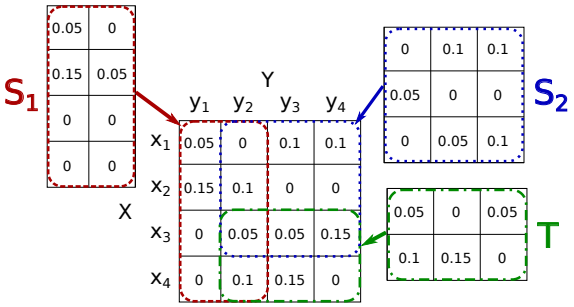


Fig. 1: Joint distribution P_{XY} composed of 3 components

1.1. Contributions

The contributions of this paper are as follows:

- We propose a new method, called hybrid statistical estimation, that combines statistical and precise analysis on the estimation of mutual information (which can also be applied to Shannon entropy and conditional Shannon entropy). Specifically, we show theoretical results on compositionally computing the bias and confidence interval of the estimate from multiple results obtained by statistical and precise analysis;
- We present a weighted statistical analysis method with different sample sizes over different components and a method for adaptively optimizing the sample sizes by evaluating the accuracy and cost of the analysis;

- We show how to reduce the sample sizes by using prior knowledge about systems, including an abstraction-then-sampling technique based on qualitative analysis;
- We show that the proposed method can be applied not only to composed systems but also to the source codes of a single system by decomposing it into components and determine the analysis method for each component;
- We provide a practical implementation in the HyLeak tool [BKLT], and show how the techniques in this paper can be applied to multiple benchmarks;
- We evaluate the quality of the estimation in this method, showing that the estimates are more accurate than statistical analysis alone for the same sample size, and that the new method outperforms the state-of-the-art statistical analysis tool LeakWatch [CKN14];
- We demonstrate the effectiveness of the hybrid method in case studies on the quantification of information leakage;

A preliminary version of this paper, without proofs, appeared in [KBL16]. Also a preliminary version of the implementation description (Sections 7 and 8.2.1), without details, appeared in the tool paper describing HyLeak [BKLT17]. In this paper we add the estimation of Shannon entropy (Propositions 4.3, 4.4 and 6.2) in Section 4.3 and that of conditional entropy (Propositions 5.4 and 5.5) in Section 5.1.3. We also show the formulas for the adaptive analysis using knowledge of prior distributions (Proposition 6.3) in Section 5.1.2 and using the abstraction-then-sampling technique (Theorem 6.4) in Section 5.2. Furthermore, we provide detailed explanation on the implementation in HyLeak tool in Section 7, including how to decompose the source code of a system into components. We also present more experimental results with details in Section 8.2. Finally, we add Appendix A to present the detailed proofs.

The rest of the paper is structured as follows. Section 2 introduces background in quantification of information and compares precise analysis with statistical analysis for the estimation of mutual information. Section 3 overviews our new method for estimating mutual information. Section 4 describes the main results of this paper: the statistical estimation of mutual information for the hybrid method, including the method for optimizing sample sizes for different components. Section 5 presents how to reduce sample sizes by using prior knowledge about systems, including the abstraction-then-sampling technique with qualitative analysis. Section 6 derives the optimal assignment of samples to components to be samples statistically to improve the accuracy of the estimate. Section 7 overviews how the implementation of the techniques in the HyLeak tool, including how to decompose the source code of a system into components and to determine the analysis method for each component. Section 8 evaluates the proposed method and illustrates its effectiveness against the state of the art and Section 9 concludes the paper. All proofs can be found in Appendix A.

1.2. Related Work

The information-theoretical approach to program security dates back to the work of Denning [Den76] and Gray [III91]. Clark et al. [CHM01, CHM07] presented techniques to automatically compute mutual information of an imperative language with loops. For a deterministic program, leakage can be computed from the equivalence relations on the secret induced by the possible outputs, and such relations can be automatically quantified [BKR09]. Under- and over-approximation of leakage based on the observation of some traces have been studied for deterministic programs [ME08, NMS09]. As an approach without replying on information theory McCamant et al. [KMPS11] developed tools implementing dynamic quantitative taint analysis techniques for security.

Fremont and Seshia [FS14] present a polynomial time algorithm to approximate the weight of traces of deterministic programs with possible application to quantitative information leakage. Progress in randomized program analysis includes a scalable algorithm for uniform generation of sample from a distribution defined as constraints [CFM+15, CMV13], with applications to constrained-random program verification.

The statistical approach to quantifying information leakage has been studied since the seminal work by Chatzikokolakis et al. [CCG10]. Chothia et al. have developed this approach in tools leakiEst [CKN13, CKNa] and LeakWatch [CKN14, CKNb]. The hybrid statistical method in this paper can be considered as their extension with the inclusion of component weighting and adaptive priors inspired by the importance sampling in statistical model checking [BHP12, CZ11]. To the best of our knowledge, no prior work has applied weighted statistical analysis to the estimation of mutual information or any other leakage measures.

The idea on combining static and randomized approaches to quantitative information flow was first proposed by Köpf and Rybalchenko [KR10] while our approach takes a different approach relying on statistical estimation to have better precision and accuracy and is general enough to deal with probabilistic systems under various prior information conditions. In related fields the hybrid approach combining precise and statistical analysis have been proven

to be effective, for instance in concolic analysis [MS07, LCFS14], where it is shown that input generated by hybrid techniques leads to greater code coverage than input generated by both fully random and concolic generation.

Our tool HyLeak processes a simple imperative language that is an extension of the language used in the QUAIL tool version 2.0 [BLQ15]. The algorithms for precise computation of information leakage used in this paper are based on trace analysis [BLMW15], implemented in the QUAIL tool [BLTW, BLTW13, BLQ15]. As remarked above, the QUAIL tool implements only a precise calculation of leakage that examines all executions of programs. Hence the performance of QUAIL does not scale, especially when the program performs complicated computations that yield a large number of execution traces. The performance of QUAIL as compared to HyLeak is represented by the "precise" analysis approach in Section 8. Since QUAIL does not support the statistical approach or the hybrid approach, it cannot handle large problems that HyLeak can analyze.

As remarked above, the stochastic simulation techniques implemented in HyLeak have also been developed in the tools LeakiEst [CKN13] (with its extension [KCP14]) and LeakWatch [CKNP13, CKN14]. The performance of these tools as compared to HyLeak is represented by the "statistical" analysis approach in Section 8.

The tool Moped-QLeak [CMS14] computes the precise information leakage of a program by transforming it into an algebraic decision diagram (ADD). As noted in [BLQ15], this technique is efficient when the program under analysis is simple enough to be converted into an ADD, and fails otherwise even when other tools including HyLeak can handle it. In particular, there are simple examples [BLQ15] where Moped-QLeak fails to produce any result but that can be examined by QUAIL and LeakWatch, hence by HyLeak.

Many information leakage analysis tools restricted to deterministic input programs have been released, including TEMU [NMS09], squifc [PM14], jpf-qif [PMTP12], QILURA [PMPd14], nsqflow [VEB+16], and SHARPPI [Wei16]. Some of these tools have been proven to scale to programs of thousands of lines written in common languages like C and Java. Such tools are not able to compute the Shannon leakage for the scenario of adaptive attacks but only compute the min-capacity of a deterministic program for the scenario of one-try guessing attacks, which give only a coarse upper bound on the Shannon leakage. More specifically, they compute the logarithm of the number of possible outputs of the deterministic program, usually by using model counting on a SMT-constraint-based representation of the possible outputs, obtained by analyzing the program. Contrary to these tools, HyLeak can analyze randomized programs[1] and provides a quite precise estimation of the Shannon leakage of the program, not just a coarse upper bound. As far as we know, HyLeak is the most efficient tool that has this greater scope and higher accuracy.

2. Background

In this section we introduce the basic concepts used in the paper. We first introduce some notions in information theory to quantify the amount of some information in probabilistic systems. Then we compare two previous analysis approaches to quantifying information: precise analysis and statistical analysis.

2.1. Quantification of Information

In this section we introduce some background on information theory, which we use to quantify the amount of information in a probabilistic system. Hereafter we write X and Y to denote two random variables, and $\mathcal{X}$ and $\mathcal{Y}$ to denote the sets of all possible values of X and Y, respectively. We denote the number of elements of a set $\mathcal{A}$ by $\#\mathcal{A}$. Given a random variable A we denote by $\mathbb{E}[A]$ or by $\overline{A}$ the expected value of A, and by $\mathbb{V}[A]$ the variance of A, i.e., $\mathbb{V}[A] = \mathbb{E}\big[(A - \mathbb{E}[A])^2\big]$.

2.1.1. Channels

In information theory, a *channel* models the input-output relation of a system as a conditional probability distribution of outputs given inputs. This model has also been used to formalize information leakage in a system that processes confidential data: *inputs* and *outputs* of a channel are respectively regarded as *secrets* and *observables* in the system and the channel represents relationships between the secrets and observables.

A *discrete channel* is a triple $(\mathcal{X}, \mathcal{Y}, C)$ where $\mathcal{X}$ and $\mathcal{Y}$ are two finite sets of discrete input and output values

[1] Some of these tools, like jpf-qif and nsqflow, present case studies on randomized protocols. However, the randomness of the programs is assumed to have the most leaking behavior. E.g., in the Dining Cryptographers this means assuming all coins produce head with probability 1.

respectively and C is an $\#\mathcal{X} \times \#\mathcal{Y}$ matrix where each element $C[x, y]$ represents the conditional probability of an output y given an input x; i.e., for each $x \in \mathcal{X}$, $\sum_{y\in\mathcal{Y}} C[x, y] = 1$ and $0 \leq C[x, y] \leq 1$ for all $y \in \mathcal{Y}$.

A *prior* is a probability distribution on input values $\mathcal{X}$. Given a prior P_X over $\mathcal{X}$ and a channel C from $\mathcal{X}$ to $\mathcal{Y}$, the *joint probability distribution* P_{XY} of X and Y is defined by: $P_{XY}[x, y] = P_X[x]C[x, y]$ for each $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

2.1.2. Shannon Entropy

We recall some information-theoretic measures as follows. Given a prior P_X on input X, the *prior uncertainty* (before observing the system's output Y) is defined as

$$H(X) = -\sum_{x\in\mathcal{X}} P_X[x] \log_2 P_X[x]$$

while the *posterior uncertainty* (after observing the system's output Y) is defined as

$$H(X|Y) = -\sum_{y\in\mathcal{Y}^+} P_Y[y] \sum_{x\in\mathcal{X}} P_{X|Y}[x|y] \log_2 P_{X|Y}[x|y],$$

where P_Y is the probability distribution on the output Y, $\mathcal{Y}^+$ is the set of outputs in $\mathcal{Y}$ with non-zero probabilities, and $P_{X|Y}$ is the conditional probability distribution of X given Y:

$$P_Y[y] = \sum_{x'\in\mathcal{X}} P_{XY}[x', y] \qquad P_{X|Y}[x|y] = \frac{P_{XY}[x, y]}{P_Y[y]} \quad \text{if } P_Y[y] \neq 0.$$

$H(X|Y)$ is also called the *conditional entropy* of X given Y.

2.1.3. Mutual Information

The amount of information gained about a random variable X by knowing a random variable Y is defined as the difference between the uncertainty about X before and after observing Y. The *mutual information* $I(X;Y)$ between X and Y is one of the most popular measures to quantify the amount of information on X gained by Y:

$$I(X;Y) = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} P_{XY}[x, y] \log_2 \left(\frac{P_{XY}[x, y]}{P_X[x]P_Y[y]} \right)$$

where P_Y is the marginal probability distribution defined as $P_Y[y] = \sum_{x\in\mathcal{X}} P_{XY}[x, y]$.

In the security scenario, information-theoretical measures quantify the amount of secret information leaked against some particular attacker: the mutual information between two random variables X on the secrets and Y on the observables in a system measures the information that is inferable about the secret by knowing the observable. In this scenario mutual information, or Shannon leakage, assumes an attacker that can ask binary questions on the secret's value after observing the system while min-entropy leakage [Smi09] considers an attacker that has only one attempt to guess the secret's value.

Mutual information has been employed in many other applications including Bayesian networks [Jen96], telecommunications [Gal68], pattern recognition [ESB09], machine learning [Mac02], quantum physics [Wil13], and biology [Ada04]. In this work we focus on mutual information and its application to the above security scenario.

2.2. Computing Mutual Information in Probabilistic Systems

In this section we present two previous approaches to computing mutual information in probabilisitic systems in the context of quantitative information flow. Then we compare the two approaches to discuss their advantages and disadvantages.

In the rest of the paper a *probabilistic system* $\mathcal{S}$ is defined as a finite set of *execution traces* such that each trace tr records the values of all variables in $\mathcal{S}$ and is associated with a probability $\mathbf{P}_\mathcal{S}[tr]$. Note that $\mathcal{S}$ does not have non-deterministic transitions. For the sake of generality we do not assume any specific constraints at this moment.

The main computational difficulties in calculating the mutual information $I(X;Y)$ between input X and output Y lies in the computation of the joint probability distribution P_{XY} of X and Y especially when the system consists of a large number of execution traces and when the distribution P_{XY} is represented as a large data structure. In previous

work this computation has been performed either by the *precise* approach using program analysis techniques or by the *statistical* approach using random sampling and statistics.

2.2.1. Precise Analysis

Precise analysis consists of analyzing all the execution traces of a system and determining for each trace tr, the input x, output y, and probability $\mathbf{P}_{\mathcal{S}}[tr]$ by concretely or symbolically executing the system. The precise analysis approach in this paper follows the depth-first trace exploration technique presented by Biondi et al. [BLQ15].

To obtain the exact joint probability $P_{XY}[x, y]$ for each $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ in a system $\mathcal{S}$, we sum the probabilities of all execution traces of $\mathcal{S}$ that have input x and output y, i.e.,

$$P_{XY}[x, y] = \sum \left\{ \mathbf{P}_{\mathcal{S}}[tr] \;\middle|\; tr \in \mathcal{S} \text{ has input } x \text{ and output } y \right\}$$

where $\mathbf{P}_{\mathcal{S}}$ is the probability distribution over the set of all traces in $\mathcal{S}$. This means the computation time depends on the number of traces in the system. If the system has a very large number of traces, it is intractable for the analyst to precisely compute the joint distribution and consequently the mutual information.

In [YT14] the calculation of mutual information is shown to be computationally expensive. This computational difficulty comes from the fact that entropy-based properties are hyperproperties [CS10] that are defined using all execution traces of the system and therefore cannot be verified on each single trace. For example, when we investigate the information leakage in a system, it is insufficient to check the leakage separately for each component of the system, because the attacker may derive sensitive information by combining the outputs of different components. More generally, the computation of entropy-based properties (such as the amount of leaked information) is not compositional, in the sense that an entropy-based property of a system is not the (weighted) sum of those of the components.

For this reason it is inherently difficult to naïvely combine analyses of different components of a system to compute entropy-based properties. In fact, previous studies on the compositional approach in quantitative information flow analysis have faced certain difficulties in obtaining useful bounds on information leakage [BK11, ES13, KG15, KCP17].

2.2.2. Statistical Analysis

Due to the complexity of precise analysis, some previous studies have focused on computing approximate values of entropy-based measures. One of the common approaches is *statistical analysis* based on Monte Carlo methods, in which approximate values are computed from repeated random sampling and their accuracy is evaluated using statistics. Previous work on quantitative information flow has used statistical analysis to estimate mutual information [CCG10, Mod89, Bri04], channel capacity [CCG10, BP14] and min-entropy leakage [CKN14, CK14].

In the statistical estimation of mutual information between two random variables X and Y in a probabilistic system, the analyst executes the system many times and collects the execution traces, each of which has a pair of values $(x, y) \in \mathcal{X} \times \mathcal{Y}$ corresponding to the input and output of the trace. This set of execution traces is used to estimate the empirical joint distribution $\hat{P}_{XY}$ of X and Y and then to estimate the mutual information $\hat{I}(X;Y)$.

Note that the empirical distribution $\hat{P}_{XY}$ is different from the true distribution P_{XY} and thus the estimated mutual information $\hat{I}(X;Y)$ is different from the true value $I(X;Y)$. In fact, it is known that entropy-based measures such as mutual information and min-entropy leakage have some bias and variance that depends on the number of collected traces, the matrix size and other factors. However, results on statistics allow us to correct the bias of the estimate and to compute the variance (and the 95% confidence interval). This way we can guarantee the quality of the estimation, which differentiates the statistical approach from the testing approach.

2.2.3. Comparing the Two Analysis Methods

The cost of the statistical analysis is proportional to the size $\#\mathcal{X} \times \#\mathcal{Y}$ of the joint distribution matrix (strictly speaking, to the number of non-zero elements in the matrix). Therefore, this method is significantly more efficient than precise analysis if the matrix is relatively small and the number of all traces is very large (for instance because the system's internal variables have a large range).

On the other hand, if the matrix is very large, the number of executions needs to be very large to obtain a reliable and small confidence interval. In particular, for a small sample size, statistical analysis does not detect rare events, i.e., traces with a low probability that affect the result. Therefore the precise analysis is significantly more efficient than statistical analysis if the number of all traces is relatively small and the matrix is relatively large (for instance because the system's internal variables have a small range).

	Precise analysis	**Statistical analysis**
Type	White box	Black/gray box
Analyzes	Source code	Implementation
Produces	Exact value	Estimate & accuracy evaluation
Reduces costs by	Random sampling	Knowledge of code & abstraction
Impractical for	Large number of traces	Large channel matrices

Table 1. Comparison of the precise and statistical analysis methods.

The main differences between precise analysis and statistical analysis are summarized in Table 1.

3. Overview of the Hybrid Statistical Estimation Method

In this section we overview a new method for estimating the mutual information between two random variables X (over the inputs $\mathcal{X}$) and Y (over the outputs $\mathcal{Y}$) in a system. The method, we call *hybrid statistical estimation*, integrates both precise and statistical analyses to overcome the limitations on those previous approaches (explained in Section 2.2).

In our hybrid analysis method we first decompose a given probabilistic system $\mathcal{S}$ into distinct *components*, which we will define below, and then apply different types of analysis (with possibly different parameters) on different components of the system. More specifically, for each component, our hybrid method chooses the faster analysis between the precise and statistical analyses. Hence the hybrid analysis of the whole system is faster than the precise analysis alone and than the statistical analysis alone, while it gives more accurate estimates than the statistical analysis alone.

To introduce the notion of components we recall that in Section 2.2 a probabilistic system $\mathcal{S}$ is defined as the set of all execution traces such that each trace tr is associated with probability $\mathbf{P}[tr]$[2]. Then a *decomposition* α of $\mathcal{S}$ is defined as a partition of the set $\mathcal{S}$; i.e., α is a collection of subsets of $\mathcal{S}$ such that: $\emptyset \notin \alpha$, $\mathcal{S} = \bigcup_{S_i \in \alpha} S_i$, and for any $S_i, S_j \in \alpha$, $S_i \neq S_j$ implies $S_i \cap S_j \neq \emptyset$. Then each element of α is called a *component*. The probability that an execution of $\mathcal{S}$ yields a component $S_i \in \alpha$ is given by $\mathbf{P}[S_i]$.

In decomposing a system we roughly investigate the characteristics of each component's behaviour to choose a faster analysis method for each component. Note that information about a component like its number of traces and the size of its joint sub-distribution matrix can be estimated heuristically before computing the matrix itself. This will be explained in Section 7; before that section this information is assumed to be available. The choice of the analysis method is as follows:

- If a component's behaviour is deterministic, we perform a precise analysis on it.
- If a component's behaviour is described as a joint sub-distribution matrix over *small*[3] subsets of $\mathcal{X}$ and $\mathcal{Y}$, then we perform a statistical analysis on the component.
- If a component's behaviour is described as a matrix over *large*[2] subsets of $\mathcal{X}$ and $\mathcal{Y}$, then we perform a precise analysis on the component.
- By combining the analysis results on all components we compute the estimated value of mutual information and its variance (and confidence interval). See Section 4 for details.
- By incorporating information form *qualitative* information flow analysis, the analyst may obtain partial knowledge on components and be able to reduce the sample sizes. See Section 5 for details.

See Section 7 for the details on how to decompose a system.

One of the main advantages of hybrid statistical estimation is that we guarantee the quality of the outcome by removing its bias and providing its variance (and confidence interval) even though different kinds of analysis with different parameters (such as sample sizes) are combined together.

Another advantage is the compositionality in estimating bias and variance. Since the sampling of execution traces is performed independently for each component, we obtain that the bias and variance of mutual information can be

[2] Note that this work considers only probabilistic systems without non-deterministic transitions.
[3] Relatively to the number of all execution traces of the component.

		Bias correction	Variance computation	Adaptive sampling
No knowledge on the system	Mutual information	Theorem 4.1	Theorem 4.2	Theorem 6.1
	Shannon entropy	Proposition 4.3	Proposition 4.4	Proposition 6.2
Knowledge on the prior	Mutual information	Proposition 5.2	Proposition 5.3	Proposition 6.3
	Conditional entropy	Proposition 5.4	Proposition 5.5	—
Abstraction-then-sampling	Mutual information	Theorem 5.6	Theorem 5.7	Theorem 6.4

Table 2. Our results on the hybrid method.

computed in a compositional way, i.e., the bias/variance for the entire system is the sum of those for the components. This compositionality enables us to find optimal sample sizes for the different components that maximize the accuracy of the estimation (i.e., minimize the variance) given a fixed total sample size for the entire system. On the other hand, the computation of mutual information itself is not compositional [KCP17]: it requires calculating the *full* joint probability distribution of the system by summing the joint sub-distributions of all components of the system.

Finally, note that these results can be applied to the estimation of Shannon entropy (Section 4.3) and conditional Shannon entropy (Section 5.1.3) as special cases. The overview of all results is summarized in Table 2.

4. Hybrid Method for Statistical Estimation of Mutual Information

In this section we present a method for estimating the mutual information between two random variables X (over the inputs $\mathcal{X}$) and Y (over the outputs $\mathcal{Y}$) in a system, and for evaluating the precision and accuracy of the estimation.

We consider a probabilistic system $\mathcal{S}$ that consists of $(m+k)$ components $S_1, S_2, \ldots, S_m$ and $T_1, T_2, \ldots, T_k$ each executed with probabilities $\theta_1, \theta_2, \ldots, \theta_m$ and $\xi_1, \xi_2, \ldots, \xi_k$, i.e., when $\mathcal{S}$ is executed, it yields S_i with the probability θ_i and T_j with the probability ξ_j. Let $\mathcal{I} = \{1, 2, \ldots, m\}$ and $\mathcal{J} = \{1, 2, \ldots, k\}$, one of which can be empty. Then the probabilities of all components sum up to 1, i.e., $\sum_{i\in\mathcal{I}} \theta_i + \sum_{j\in\mathcal{J}} \xi_j = 1$. We assume that the analyst is able to compute these probabilities by precise analysis.

Once the system is decomposed into components each component is analyzed either by precise analysis or by statistical analysis. We assume that the analyst can run the component S_i for each $i \in \mathcal{I}$ to record a certain number of S_i's execution traces, and precisely analyze the components T_j for $j \in \mathcal{J}$ to record a certain symbolic representation of T_j's all execution traces, e.g., by static analysis of the source code or specification.

In the rest of this section we present a method for computing the joint probability distribution $\hat{P}_{XY}$ (Section 3), for estimating the mutual information $\hat{I}(X;Y)$ (Section 4.1), and for evaluating the accuracy of the estimation (Section 4.2). Then we show the application of our hybrid method to Shannon entropy estimation (Section 4.3).

In the estimation of mutual information between the two random variables X and Y in the system $\mathcal{S}$, we need to estimate the joint probability distribution P_{XY} of X and Y.

In our approach this is obtained by combining the joint *sub-probability distributions* of X and Y for all the components S_i's and T_j's. More specifically, let R_i and Q_j be the joint sub-distributions of X and Y for the components S_i's and T_j's respectively. Then the joint (full) distribution P_{XY} for the whole system $\mathcal{S}$ is defined by:

$$P_{XY}[x,y] \stackrel{\text{def}}{=} \sum_{i\in\mathcal{I}} R_i[x,y] + \sum_{j\in\mathcal{J}} Q_j[x,y]$$

for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Note that for each $i \in \mathcal{I}$ and $j \in \mathcal{J}$, the sums of all probabilities in the sub-distribution R_i and in Q_j respectively equal the probabilities θ_i (of executing S_i) and ξ_j (of executing T_j).

To estimate the joint distribution P_{XY} the analyst computes

- for each $j \in \mathcal{J}$, the *exact* sub-distribution Q_j for the component T_j by precise analysis on T_j, and
- for each $i \in \mathcal{I}$, the *empirical* sub-distribution $\hat{R}_i$ for S_i from a set of traces obtained by executing S_i a certain number n_i of times.

More specifically, the empirical sub-distribution $\hat{R}_i$ is constructed as follows. When the component S_i is executed n_i times, let K_{ixy} be the number of traces that have input $x \in \mathcal{X}$ and output $y \in \mathcal{Y}$. Then $n_i = \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} K_{ixy}$. From

these numbers K_{ixy} of traces we compute the empirical joint (full) distribution $\hat{D}_i$ of X and Y by:

$$\hat{D}_i[x,y] \stackrel{\text{def}}{=} \frac{K_{ixy}}{n_i} \ .$$

Since S_i is executed with probability θ_i, the sub-distribution $\hat{R}_i$ is given by $\hat{R}_i[x,y] \stackrel{\text{def}}{=} \theta_i \hat{D}_i[x,y] = \frac{\theta_i K_{ixy}}{n_i}$.

Then the analyst sums up these sub-distributions to obtain the joint distribution $\hat{P}_{XY}$ for the whole system $\mathcal{S}$:

$$\hat{P}_{XY}[x,y] \stackrel{\text{def}}{=} \sum_{i\in\mathcal{I}} \hat{R}_i[x,y] + \sum_{j\in\mathcal{J}} Q_j[x,y] = \sum_{i\in\mathcal{I}} \frac{\theta_i K_{ixy}}{n_i} + \sum_{j\in\mathcal{J}} Q_j[x,y]\,.$$

Note that R_i and Q_j may have different matrix sizes and cover different parts of the joint distribution matrix $\hat{P}_{XY}$, so they may have to be padded with zeroes for the summation.

4.1. Estimation of Mutual Information and Correction of its Bias

In this section we present our new method for estimating mutual information and for correcting its bias. For each component S_i let D_i be the joint (full) distribution of X and Y obtained by normalizing R_i: $D_i[x,y] = \frac{R_i[x,y]}{\theta_i}$. Let $D_{Xi}[x] = \sum_{y\in\mathcal{Y}} D_i[x,y]$, $D_{Yi}[y] = \sum_{x\in\mathcal{X}} D_i[x,y]$ and $\mathcal{D} = \{(x,y) \in \mathcal{X}\times\mathcal{Y} : P_{XY}[x,y] \neq 0\}$.

Using the estimated joint distribution $\hat{P}_{XY}$ we can compute the mutual information estimate $\hat{I}(X;Y)$. Note that the mutual information for the whole system is smaller than (or equals) the weighted sum of those for the components, because of its convexity w.r.t. the channel matrix. Therefore it cannot be computed compositionally from those of the components, i.e., it is necessary to compute the joint distribution matrix $\hat{P}_{XY}$ for the whole system.

Since $\hat{I}(X;Y)$ is obtained from a limited number of traces, it has bias, i.e., its expected value $\mathbb{E}\Big[\hat{I}(X;Y)\Big]$ is different from the true value $I(X;Y)$. The bias $\mathbb{E}\Big[\hat{I}(X;Y)\Big] - I(X;Y)$ in the estimation is quantified as follows.

Theorem 4.1 (Mean of estimated mutual information). The expected value $\mathbb{E}\Big[\hat{I}(X;Y)\Big]$ of the estimated mutual information is given by:

$$\mathbb{E}\Big[\hat{I}(X;Y)\Big] = I(X;Y) + \sum_{i\in\mathcal{I}} \frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}} \varphi_{ixy} - \sum_{x\in\mathcal{X}^+} \varphi_{ix} - \sum_{y\in\mathcal{Y}^+} \varphi_{iy}\Big) + \mathcal{O}(n_i^{-2})$$

where $\varphi_{ixy} = \frac{D_i[x,y]-D_i[x,y]^2}{P_{XY}[x,y]}$, $\varphi_{ix} = \frac{D_{Xi}[x]-D_{Xi}[x]^2}{P_X[x]}$ and $\varphi_{iy} = \frac{D_{Yi}[y]-D_{Yi}[y]^2}{P_Y[y]}$.

Proof sketch. Here we present only the basic idea. Appendices A.1 and A.2 present a proof of this theorem by proving a more general claim, i.e., Theorem 5.6 in Section 5.2.

By properties of mutual information and Shannon entropy, we have:

$$\begin{aligned}\mathbb{E}\Big[\hat{I}(X;Y)\Big] - I(X;Y) &= \mathbb{E}\Big[\hat{H}(X) + \hat{H}(Y) - \hat{H}(X,Y)\Big] - \Big(H(X) + H(Y) - H(X,Y)\Big)\\ &= \Big(\mathbb{E}\Big[\hat{H}(X)\Big] - H(X)\Big) + \Big(\mathbb{E}\Big[\hat{H}(Y)\Big] - H(Y)\Big) - \Big(\mathbb{E}\Big[\hat{H}(X,Y)\Big] - H(X,Y)\Big)\,.\end{aligned}$$

Hence it is sufficient to calculate the bias in $\hat{H}(X)$, $\hat{H}(Y)$, and $\hat{H}(X,Y)$, respectively.

We calculate the bias in $\hat{H}(X,Y)$ as follows. Let $f_{xy}(K_{1xy}, K_{2xy}, \ldots, K_{mxy})$ be the m-ary function defined by:

$$f_{xy}(K_{1xy}, K_{2xy}, \ldots, K_{mxy}) = \Big(\sum_{i\in\mathcal{I}} \frac{\theta_i K_{ixy}}{n_i} + \sum_{j\in\mathcal{J}} Q_j[x,y]\Big) \log\Big(\sum_{i\in\mathcal{I}} \frac{\theta_i K_{ixy}}{n_i} + \sum_{j\in\mathcal{J}} Q_j[x,y]\Big),$$

which equals $\hat{P}_{XY}[x,y] \log \hat{P}_{XY}[x,y]$. Let $\boldsymbol{K}_{xy} = (K_{1xy}, K_{2xy}, \ldots, K_{mxy})$. Then the empirical joint entropy is:

$$\hat{H}(X,Y) = -\sum_{(x,y)\in\mathcal{D}} \hat{P}_{XY}[x,y] \log \hat{P}_{XY}[x,y] = -\sum_{(x,y)\in\mathcal{D}} f_{xy}(\boldsymbol{K}_{xy}).$$

Let $\overline{K_{ixy}} = \mathbb{E}[K_{ixy}]$ for each $i \in \mathcal{I}$ and $\overline{\boldsymbol{K}_{xy}} = \mathbb{E}[\boldsymbol{K}_{xy}]$. By the Taylor expansion of $f_{xy}(\boldsymbol{K}_{xy})$ (w.r.t. the multiple dependent variables $\boldsymbol{K}_{xy}$) at $\overline{\boldsymbol{K}_{xy}}$, we have:

$$f_{xy}(\boldsymbol{K}_{xy}) = f_{xy}(\overline{\boldsymbol{K}_{xy}}) + \sum_{i\in\mathcal{I}} \frac{\partial f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{ixy}}(K_{ixy} - \overline{K_{ixy}}) + \frac{1}{2}\sum_{i,j\in\mathcal{I}} \frac{\partial^2 f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{ixy}\partial K_{jxy}}(K_{ixy} - \overline{K_{ixy}})(K_{jxy} - \overline{K_{jxy}}) + \sum_{i\in\mathcal{I}} \mathcal{O}(K_{ixy}^3).$$

We use the following properties:

- $\mathbb{E}\big[K_{ixy} - \overline{K_{ixy}}\big] = 0$, which is immediate from $\overline{K_{ixy}} = \mathbb{E}[K_{ixy}]$.
- $\mathbb{E}\big[(K_{ixy} - \overline{K_{ixy}})(K_{jxy} - \overline{K_{jxy}})\big] = 0$ if $i \neq j$, because K_{ixy} and K_{jxy} are independent.
- $\mathbb{E}\big[(K_{ixy} - \overline{K_{ixy}})^2\big] = \mathbb{V}[K_{ixy}] = n_i D_i[x,y](1 - D_i[x,y])$.

Then

$$\begin{aligned}
\mathbb{E}\Big[\hat{H}(X,Y)\Big] &= -\sum_{(x,y)\in\mathcal{D}} \mathbb{E}[\, f_{xy}(\boldsymbol{K}_{xy})\,] \\
&= -\sum_{(x,y)\in\mathcal{D}} \Big(f_{xy}(\overline{\boldsymbol{K}_{xy}}) + \frac{1}{2}\sum_{i\in\mathcal{I}} \frac{\partial^2 f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{ixy}\partial K_{ixy}}\, \mathbb{E}\big[(K_{ixy} - \overline{K_{ixy}})^2\big] + \mathcal{O}(K_{ixy}^3)\Big) \\
&= -\sum_{(x,y)\in\mathcal{D}} \Big(f_{xy}(\overline{\boldsymbol{K}_{xy}}) + \frac{1}{2}\sum_{i\in\mathcal{I}} \frac{\theta_i^2}{n_i^2 P_{XY}[x,y]}\, n_i D_i[x,y](1 - D_i[x,y]) + \mathcal{O}(n_i^{-2})\Big) \\
&= H(X,Y) - \sum_{i\in\mathcal{I}} \frac{\theta_i^2}{2n_i} \sum_{(x,y)\in\mathcal{D}} \varphi_{ixy} + \mathcal{O}(n_i^{-2}),
\end{aligned}$$

where the derivation of the equalities is detailed in Appendix A. Hence the bias in estimating $H(X,Y)$ is given by:

$$\mathbb{E}\Big[\hat{H}(X,Y)\Big] - H(X,Y) = -\sum_{i\in\mathcal{I}} \frac{\theta_i^2}{2n_i} \sum_{(x,y)\in\mathcal{D}} \varphi_{ixy} + \mathcal{O}(n_i^{-2})\,.$$

Analogously, we can calculate the bias in $\hat{H}(X)$ and $\hat{H}(Y)$ to derive the theorem. See Appendices A.1 and A.2 for the details. □

Since the higher-order terms in the formula are negligible when the sample sizes n_i are large enough, we use the following as the *point estimate* of the mutual information:

$$pe = \hat{I}(X;Y) - \sum_{i\in\mathcal{I}} \frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}} \widehat{\varphi}_{ixy} - \sum_{x\in\mathcal{X}^+} \widehat{\varphi}_{ix} - \sum_{y\in\mathcal{Y}^+} \widehat{\varphi}_{iy}\Big)$$

where $\widehat{\varphi}_{ixy}$, $\widehat{\varphi}_{ix}$ and $\widehat{\varphi}_{iy}$ are respectively empirical values of φ_{ixy}, φ_{ix} and φ_{iy} that are computed from traces; i.e., $\widehat{\varphi}_{ixy} = \frac{\hat{D}_i[x,y] - \hat{D}_i[x,y]^2}{\hat{P}_{XY}[x,y]}$, $\widehat{\varphi}_{ix} = \frac{\hat{D}_i[x] - \hat{D}_i[x]^2}{\hat{P}_{XY}[x]}$, and $\widehat{\varphi}_{iy} = \frac{\hat{D}_i[y] - \hat{D}_i[y]^2}{\hat{P}_{XY}[y]}$. Then the bias is closer to 0 when the sample sizes n_i are larger.

4.2. Evaluation of the Accuracy of Estimation

In this section we present a way of evaluating the accuracy of mutual information estimation.

The quality of the estimate depends on the sample sizes n_i and other factors. The sampling distribution of the estimate $\hat{I}(X;Y)$ tends to follow the normal distribution when n_i's are large enough. The following gives the variance of the distribution.

Theorem 4.2 (Variance of estimated mutual information). The variance $\mathbb{V}\Big[\hat{I}(X;Y)\Big]$ of the estimated mutual information is given by:

$$\mathbb{V}\Big[\hat{I}(X;Y)\Big] = \sum_{i\in\mathcal{I}} \frac{\theta_i^2}{n_i}\left(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1 + \log\tfrac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)^2 - \left(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1 + \log\tfrac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)\right)^2\right) + O(n_i^{-2}).$$

Proof sketch. The variance is calculated using the following:

$$\begin{aligned}\mathbb{V}\Big[\hat{I}(X;Y)\Big] &= \mathbb{V}\Big[\hat{H}(X)+\hat{H}(Y)-\hat{H}(X,Y)\Big]\\ &= \mathbb{V}\Big[\hat{H}(X)\Big]+\mathbb{V}\Big[\hat{H}(Y)\Big]+\mathbb{V}\Big[\hat{H}(X,Y)\Big]\\ &\quad + 2Cov\Big[\hat{H}(X),\hat{H}(Y)\Big]-2Cov\Big[\hat{H}(X),\hat{H}(X,Y)\Big]-2Cov\Big[\hat{H}(Y),\hat{H}(X,Y)\Big]\,.\end{aligned}$$

The calculation of these variances and covariances and the whole proof are shown in Appendices A.3 and A.4. (We will present a proof of this theorem by showing a more general claim, i.e., Theorem 5.7 in Section 5.2). □

The confidence interval of the estimate of mutual information is useful to know how accurate the estimate is. A small confidence interval corresponds to a reliable estimate. The confidence interval is calculated using the variance v obtained by Theorem 4.2. Given a significance level α, we denote by $z_{\alpha/2}$ the *z-score* for the $100(1-\frac{\alpha}{2})$ percentile point. Then *the* $(1-\alpha)$ *confidence interval* of the estimate is given by:

$$[\,\max(0, pe - z_{\alpha/2}\sqrt{v}),\;\; pe + z_{\alpha/2}\sqrt{v}\,]\,.$$

For example, we use the z-score $z_{0.0025} = 1.96$ to compute the 95% confidence interval. To ignore the higher order terms the sample size $\sum_{i\in\mathcal{I}} n_i$ needs to be at least $4\cdot\#\mathcal{X}\cdot\#\mathcal{Y}$.

By Theorems 4.1 and 4.2, the bias and variance for the whole system can be computed compositionally from those for the components, unlike the mutual information itself. This allows us to adaptively optimize the sample sizes for the components as we will see in Section 6.

4.3. Application to Estimation of Shannon Entropy

Hybrid statistical estimation can also be used to estimate the Shannon entropy $H(X)$ of a random variable X in a probabilistic system. Although the results for Shannon entropy are straightforward from those for mutual information, we present the formulas here for completeness. For each $i \in \mathcal{I}$ let D_{Xi} be the sub-distribution of X for the component S_i. Then the mean and variance of the estimate are obtained in the same way as in the Sections 4.1 and 4.2.

Proposition 4.3 (Mean of estimated Shannon entropy). The expected value $\mathbb{E}\Big[\hat{H}(X)\Big]$ of the estimated Shannon entropy is given by:

$$\mathbb{E}\Big[\hat{H}(X)\Big] = H(X) - \sum_{i\in\mathcal{I}} \frac{\theta_i^2}{2n_i} \sum_{x\in\mathcal{X}^+} \frac{D_{Xi}[x]\,(1-D_{Xi}[x])}{P_X[x]} + \mathcal{O}(n_i^{-2})\,.$$

See Appendix A.2 for the proof. From this we obtain the bias of the Shannon entropy estimates.

Proposition 4.4 (Variance of estimated Shannon entropy). The variance $\mathbb{V}\Big[\hat{H}(X)\Big]$ of the estimated Shannon entropy is given by:

$$\mathbb{V}\Big[\hat{H}(X)\Big] = \sum_{i\in\mathcal{I}} \frac{\theta_i^2}{n_i}\left(\sum_{x\in\mathcal{X}^+} D_{Xi}[x]\Big(1+\log P_X[x]\Big)^2 - \Big(\sum_{x\in\mathcal{X}^+} D_{Xi}[x]\Big(1+\log P_X[x]\Big)\Big)^2\right) + \mathcal{O}(n_i^{-2})\,.$$

See Appendix A.4 for the proof. From this we obtain the confidence interval of the Shannon entropy estimates.

5. Estimation Using Prior Knowledge about Systems

In this section we show how to use prior knowledge about systems to improve the accuracy of the estimation, i.e., to make the variance (and the confidence interval size) smaller and reduce the required sample sizes.

5.1. Approximate Estimation Using Knowledge of Prior Distributions

Our hybrid statistical estimation method integrates both precise and statistical analysis, and it can be seen as a generalization and extension of previous work [CCG10, Mod89, Bri04].

Due to an incorrect computation of the bias, the LeakWatch [CKN14, CKNb] tool based on this work does not correctly estimate mutual information. We explain this problem in Section 5.1.1 and show how to fix it in Section 5.1.2. Section 5.1.3 extends this result to the estimation of conditional entropy.

5.1.1. State of the Art

For example, Chatzikokolakis et.al. [CCG10] present a method for estimating mutual information between two random variables X (over secret input values $\mathcal{X}$) and Y (over observable output values $\mathcal{Y}$) when the analyst knows the (prior) distribution P_X of X. In the estimation they collect execution traces by running a system for each secret value $x \in \mathcal{X}$. Thanks to the precise knowledge of P_X, they have more precise and accurate estimates than the other previous work [Mod89, Bri04] that also estimates P_X from execution traces.

Estimation using the precise knowledge of P_X is an instance of our result if a system is partitioned into the component S_x for each secret $x \in \mathcal{X} = \mathcal{I}$. If we assume all joint probabilities are non-zero, the following approximate result in [CCG10] follows from Theorem 4.1.

Corollary 5.1. The expected value $\mathbb{E}\left[\hat{I}(X;Y)\right]$ of the estimated mutual information is given by:

$$\mathbb{E}\left[\hat{I}(X;Y)\right] = I(X;Y) + \frac{(\#\mathcal{X}-1)(\#\mathcal{Y}-1)}{2n} + \mathcal{O}(n^{-2}),$$

where $\#\mathcal{X}$ and $\#\mathcal{Y}$ denote the numbers of possible secrets and observables respectively.

Using this result the bias $\mathbb{E}\left[\hat{I}(X;Y)\right] - I(X;Y)$ is calculated as $\frac{(\#\mathcal{X}-1)(\#\mathcal{Y}-1)}{2n}$ in [CCG10], which depends only on the size of the joint distribution matrix. However, the bias can be strongly influenced by probability values close or equivalent to zero in the distribution, therefore their approximate results can be correct only when all joint probabilities are non-zero and large enough, which is a strong restriction in practice. We show in Section 8.2.3 that the tool LeakWatch [CKN14] uses Corollary 5.1, and consequently miscalculates bias and gives an estimate far from the true value in the presence of very small probability values.

5.1.2. Our Estimation Using Knowledge of Prior Distributions

To overcome these issues we present more general results in the case in which the analyst knows the prior distribution P_X. We assume that a system $\mathcal{S}$ is partitioned into the disjoint component S_{ix} for each index $i \in \mathcal{I}$ and input $x \in \mathcal{X}$, and that each S_{ix} is executed with probability θ_{ix} in the system $\mathcal{S}$. Let $\Theta = \{\theta_{ix} : i \in \mathcal{I},\, x \in \mathcal{X}\}$.

Estimation of Mutual Information In the estimation of mutual information we separately execute each component S_{ix} multiple times to collect execution traces. Unlike the previous work the analyst may change the number of executions $n_i P_X[x]$ to $n_i \lambda_i[x]$ where $\lambda_i[x]$ is an *importance prior* that the analyst chooses to determine how the sample size n_i is allocated for each component S_{ix}. Let $\Lambda = \{\lambda_i : i \in \mathcal{I}\}$.

Given the number K_{ixy} of S_{ix}'s traces with output y, we define the conditional distribution D_i of output given input: $D_i[y|x] \stackrel{\text{def}}{=} \frac{K_{ixy}}{n_i \lambda_i[x]}$. Let $M_{ixy} = \frac{\theta_{ix}^2}{\lambda_i[x]} D_i[y|x]\,(1 - D_i[y|x])$. Then we can calculate the mean and variance of the mutual information $\hat{I}_{\Theta,\Lambda}(X;Y)$ using $\hat{D}_i$, Θ, Λ as follows.

Proposition 5.2 (Mean of mutual information estimated using the knowledge of the prior). The expected value $\mathbb{E}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right]$ of the estimated mutual information is given by:

$$\mathbb{E}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right] = I(X;Y) + \sum_{i\in\mathcal{I}} \frac{1}{2n_i} \sum_{y\in\mathcal{Y}^+} \left(\sum_{x\in\mathcal{D}_y} \frac{M_{ixy}}{P_{XY}[x,y]} - \frac{\sum_{x\in\mathcal{D}_y} M_{ixy}}{P_Y[y]} \right) + \mathcal{O}(n_i^{-2})\,.$$

Proposition 5.3 (Variance of mutual information estimated using the knowledge of the prior). The variance $\mathbb{V}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right]$ of the estimated mutual information is given by:

$$\mathbb{V}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right] = \sum_{i\in\mathcal{I}} \sum_{x\in\mathcal{X}^+} \frac{\theta_{ix}^2}{n_i \lambda_i[x]} \left(\sum_{y\in\mathcal{D}_x} D_i[y|x] \left(\log \frac{P_Y[y]}{P_{XY}[x,y]} \right)^2 - \left(\sum_{y\in\mathcal{D}_x} D_i[y|x] \left(\log \frac{P_Y[y]}{P_{XY}[x,y]} \right) \right)^2 \right) + \mathcal{O}(n_i^{-2})\,.$$

See Appendix A.6 for the proofs.

5.1.3. Estimation of Conditional Entropy

The new method can also estimate the conditional Shannon entropy $H(X|Y)$ of a random variable X given a random variable Y in a system. In the context of quantitative security, $H(X|Y)$ represents the uncertainty of a secret X after observing an output Y of the system. The mean and variance of the conditional entropy are obtained from those of the mutual information in the case where the analyst knows the prior.

Proposition 5.4 (Mean of estimated conditional entropy). The expected value $\mathbb{E}\left[\hat{H}_{\Theta,\Lambda}(X|Y)\right]$ of the estimated conditional Shannon entropy is given by $H(X) - \mathbb{E}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right]$ where $\mathbb{E}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right]$ is the expected value of the mutual information in the case where the analyst knows the prior (shown in Proposition 5.2).

Proof. By $H(X|Y) = H(X) - I(X;Y)$, we obtain $\mathbb{E}\left[\hat{H}_{\Theta,\Lambda}(X|Y)\right] = H(X) - \mathbb{E}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right]$. □

Proposition 5.5 (Variance of estimated conditional entropy). The variance $\mathbb{V}\left[\hat{H}_{\Theta,\Lambda}(X|Y)\right]$ of the estimated conditional Shannon entropy coincides with the variance $\mathbb{V}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right]$ of the mutual information in the case where the analyst knows the prior (shown in Proposition 5.3).

Proof. By $H(X|Y) = H(X) - I(X;Y)$, we obtain $\mathbb{V}\left[\hat{H}_{\Theta,\Lambda}(X|Y)\right] = \mathbb{V}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right]$. □

5.2. Abstraction-Then-Sampling Using Partial Knowledge of Components

In this section we extend the hybrid statistical estimation method to consider the case in which the analyst knows that the output of some of the components does not depend on the secret input (for instance by static code analysis). Such prior knowledge may help us abstract components into simpler ones and thus reduce the sample size for the statistical analysis.

We illustrate the basic idea of this "abstraction-then-sampling" technique as follows. Let us consider an analyst who knows two pairs (x, y) and (x', y') of inputs and outputs have the same probability in a component S_i: $D_i[x, y] = D_i[x', y']$. Then, when we construct the empirical distribution $\hat{D}_i$ from a set of traces, we can count the number $K_{i\{(x,y),(x',y')\}}$ of traces having either (x, y) or (x', y'), and divide it by two: $K_{ixy} = K_{ix'y'} = \frac{K_{i\{(x,y),(x',y')\}}}{2}$. Then the sample size required for a certain accuracy is smaller than when we do not use the prior knowledge on the equality $K_{ixy} = K_{ix'y'}$.

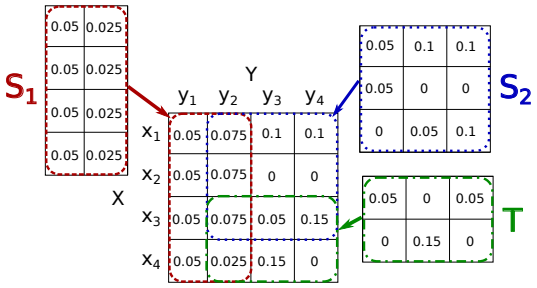


Fig. 2: Joint distribution composed of 3 components. All the rows of component S_1 are identical, hence abstraction-then-sampling can be used on it.

In the following we generalize this idea to deal with similar information that the analyst may possess about the components. Let us consider a (probabilistic) system in which for some components, observing the output provides no information on the input. Assume that the analyst is aware of this by *qualitative* information analysis (for verifying non-interference). Then such a component S_i has a sub-channel matrix where all non-zero rows have an identical conditional distribution of outputs given inputs [CT06]. Consequently, when we estimate the $\#\mathcal{X}_i \times \#\mathcal{Y}_i$ matrix of S_i it suffices to estimate one of the rows, hence the number of executions is proportional to $\#\mathcal{Y}_i$ instead of $\#\mathcal{X}_i \times \#\mathcal{Y}_i$.

The abstraction-then-sampling approach can be simply explained by referring to the joint distribution matrix in Fig. 2. Note that each row of the sub-distribution matrix for component S_1 is identical, even though the rows of the joint matrix are not, and assume that the analyst knows this by analyzing the code of the program and finding out that for component S_1 the output is independent from the input. Then the analyst would know that it is unnecessary to execute the component separately for each possible input value in S_1: it is sufficient to execute the component only for one value of the input, and to apply the results to each row in the sub-distribution matrix for component S_1. This allows the analyst to obtain more precise results and a smaller variance (and confidence interval) on the estimation given a fixed total sample size n_i for the component.

Note that even when some components leak no information, computing the mutual information for the whole system requires constructing the matrix of the system, hence the matrices of all components.

Let $\mathcal{I}^\star$ be the set of indexes of components that have channel matrices whose non-zero rows consist of the same conditional distribution. For each $i \in \mathcal{I}^\star$, we define $\pi_i[x]$ as the probability of having an input x in the component S_i. To estimate the mutual information for the whole system, we apply the abstraction-then-sampling technique to the components $\mathcal{I}^\star$ and the standard sampling technique (shown in Section 4) to the components $\mathcal{I} \setminus \mathcal{I}^\star$.

Then the mean and variance of the mutual information are as follows. The following results show that the bias and confidence interval are narrower than when not using the prior knowledge of components.

Theorem 5.6 (Mean of mutual information estimated using the abstraction-then-sampling). The expected value $\mathbb{E}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big]$ of the estimated mutual information is given by:

$$\mathbb{E}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big] = I(X;Y) + \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} \frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}} \varphi_{ixy} - \sum_{x\in\mathcal{X}^+} \varphi_{ix} - \sum_{y\in\mathcal{Y}^+} \varphi_{iy}\Big) + \sum_{i\in\mathcal{I}^\star} \frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}} \psi_{ixy} - \sum_{y\in\mathcal{Y}^+} \varphi_{iy}\Big) + \mathcal{O}(n_i^{-2})$$

where $\psi_{ixy} \stackrel{\text{def}}{=} \frac{D_i[x,y]\pi_i[x] - D_i[x,y]^2}{P_{XY}[x,y]}$.

See Appendix A.1 for the proof.

Theorem 5.7 (Variance of mutual information estimated using the abstraction-then-sampling). The variance $\mathbb{V}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big]$ of the estimated mutual information is given by:

$$\mathbb{V}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big] = \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} \frac{\theta_i^2}{n_i}\Bigg(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1+\log\frac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)^2 - \Big(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1+\log\frac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)\Big)^2\Bigg)$$
$$+ \sum_{i\in\mathcal{I}^\star} \frac{\theta_i^2}{n_i}\Bigg(\sum_{y\in\mathcal{Y}^+} D_{Yi}[y]\gamma_{ixy}^2 - \Big(\sum_{y\in\mathcal{Y}^+} D_{Yi}[y]\gamma_{ixy}\Big)^2\Bigg) + \mathcal{O}(n_i^{-2})$$

where $\gamma_{ixy} \stackrel{\text{def}}{=} \log P_Y[y] - \sum_{x\in\mathcal{X}} \pi_i[x] \log P_{XY}[x,y]$.

See Appendix A.3 for the proof.

6. Adaptive Optimization of Sample Sizes

In this section we present a method for deciding the sample size n_i of each component S_i to estimate mutual information with an optimal accuracy when using the hybrid estimation technique in Section 4 and its variants using prior information on the system in Section 5. The proof for all results in this section can br found in Appendix A.5.

Mutual Information To decide the sample sizes we take into account the trade-off between accuracy and cost of the statistical analysis: The computational cost increases proportionally to the sample size n_i (i.e., the number of S_i's execution traces), while a larger sample size n_i provides a smaller variance hence a more accurate estimate.

More specifically, given a budget of a total sample size n for the whole system, we obtain an optimal accuracy of the estimate by adjusting each component's sample size n_i [4] (under the constraint $n = \sum_{i\in\mathcal{I}} n_i$). To compute the optimal sample sizes, we first run each component to collect a small number (compared to n, for instance dozens) of execution traces. Then we calculate certain intermediate values in computing the variance and determine sample sizes for further executions. Formally, let v_i be the following intermediate value of the variance for the component S_i:

$$v_i = \theta_i^2\Bigg(\sum_{(x,y)\in\mathcal{D}} \hat{D}_i[x,y]\Big(1+\log\frac{\widehat{P}_X[x]\widehat{P}_Y[y]}{\widehat{P}_{XY}[x,y]}\Big)^2 - \Big(\sum_{(x,y)\in\mathcal{D}} \hat{D}_i[x,y]\Big(1+\log\frac{\widehat{P}_X[x]\widehat{P}_Y[y]}{\widehat{P}_{XY}[x,y]}\Big)\Big)^2\Bigg).$$

Then we find n_i's that minimize the variance $v = \sum_{i\in\mathcal{I}} \frac{v_i}{n_i}$ of the estimate by using the following theorem.

[4] This idea resembles the *importance sampling* in statistical model checking in that the sample size is adjusted to make the estimate more accurate.

Theorem 6.1 (Optimal sample sizes). Given the total sample size n and the above intermediate variance v_i of the component S_i for each $i \in \mathcal{I}$, the variance of the mutual information estimate is minimized if, for all $i \in \mathcal{I}$, the sample size n_i for S_i is given by: $n_i = \frac{\sqrt{v_i}n}{\sum_{j=1}^{m}\sqrt{v_j}}$.

Shannon Entropy Analogously to Theorem 6.1 we can adaptively optimize the sample sizes in the estimation of Shannon entropy in Section 4.3. To compute the optimal sample sizes we define v_i' by:

$$v_i' = \theta_i^2\left(\sum_{x\in\mathcal{X}^+} D_{Xi}[x]\Big(1+\log P_X[x]\Big)^2 - \Big(\sum_{x\in\mathcal{X}^+} D_{Xi}[x]\Big(1+\log P_X[x]\Big)\Big)^2\right) + \mathcal{O}(n_i^{-2})\,.$$

Then we can compute the optimal sample sizes by using the following proposition.

Proposition 6.2 (Optimal sample sizes for Shannon entropy estimation). Given the total sample size n and the above intermediate variance v_i' of the component S_i for each $i \in \mathcal{I}$, the variance of the Shannon entropy estimate is minimized if, for all $i \in \mathcal{I}$, the sample size n_i for S_i satisfies $n_i = \frac{\sqrt{v_i'}n}{\sum_{j=1}^{m}\sqrt{v_j'}}$.

Knowledge of the Prior Analogously to Theorem 6.1, the sample sizes n_i and the importance priors λ_i can be adaptively optimized in the case in which the prior distribution of the input is known presented in Section 5.1.

Proposition 6.3 (Optimal sample sizes when knowing the prior). For each $i \in \mathcal{I}$ and $x \in \mathcal{X}$, let v_{ix} be the following intermediate variance of the component S_{ix}.

$$v_{ix} = \theta_{ix}^2\left(\sum_{y\in\mathcal{D}_x} \hat{D}_i[y|x]\Big(\log\tfrac{\widehat{P}_Y[y]}{\widehat{P}_{XY}[x,y]}\Big)^2 - \Big(\sum_{y\in\mathcal{D}_x} \hat{D}_i[y|x]\Big(\log\tfrac{\widehat{P}_Y[y]}{\widehat{P}_{XY}[x,y]}\Big)\Big)^2\right).$$

Given the total sample size n, the variance of the estimated mutual information is minimized if, for all $i \in \mathcal{I}$ and $x \in \mathcal{X}$, the sample size n_i and the importance prior λ_i satisfy: $n_i\lambda_i[x] = \frac{\sqrt{v_{ix}}n}{\sum_{j=1}^{m}\sqrt{v_{jx}}}$.

Abstraction-then-sampling Finally, the sample sizes can be optimized for the abstraction-then-sampling approach in Section 5.2 by using the following theorem.

Theorem 6.4 (Optimal sample sizes using the abstraction-then-sampling). Let $v_i^\star$ be the following intermediate variance of the component S_i:

$$v_i^\star = \begin{cases} \theta_i^2\left(\sum_{(x,y)\in\mathcal{D}} \hat{D}_i[x,y]\Big(1+\log\frac{\hat{P}_X[x]\hat{P}_Y[y]}{\hat{P}_{XY}[x,y]}\Big)^2 - \Big(\sum_{(x,y)\in\mathcal{D}} \hat{D}_i[x,y]\Big(1+\log\frac{\hat{P}_X[x]\hat{P}_Y[y]}{\hat{P}_{XY}[x,y]}\Big)\Big)^2\right) & \text{if } i \in \mathcal{I}\setminus\mathcal{I}^\star \\ \theta_i^2\left(\sum_{y\in\mathcal{Y}^+} \hat{D}_{Yi}[y]\hat{\gamma}_{ixy}^2 - \Big(\sum_{y\in\mathcal{Y}^+} \hat{D}_{Yi}[y]\hat{\gamma}_{ixy}\Big)^2\right) & \text{if } i \in \mathcal{I}^\star \end{cases}$$

Given the total sample size n, the variance of the estimated mutual information is minimized if, for all $i \in \mathcal{I}$ and $x \in \mathcal{X}$, the sample size n_i is given by: $n_i = \frac{\sqrt{v_i^\star}n}{\sum_{j=1}^{m}\sqrt{v_j^\star}}$.

7. Implementation in the HyLeak Tool

We describe how HyLeak estimates the Shannon leakage of a given program, i.e., the mutual information between secret and output, implementing the hybrid statistical estimation procedure described above. The tool determines which components of the program to analyze with precise analysis and which with statistical analysis, and inserts appropriate annotations in the code. The components are analyzed with the chosen technique and the results are composed into a joint probability distribution of the secret and observable variables. Finally, the mutual information and its confidence interval are computed from the joint distribution.

The HyLeak tool, including user documentation and source code is freely available at https://project.inria.fr/hyleak/. Multiple examples and the scripts to generate the results are also provided.

HyLeak is very simple to use. The code of the system to analyze is written in a file e.g. `system.hyleak`. We invoke the tool with the command:

```
./hyleak system.hyleak
```

The tool generates various `.pp` text files with analysis information and the control flow graph of the program. Finally, it outputs the prior and posterior Shannon entropy estimates for the secret, the estimated Shannon leakage of the program before and after bias correction, and its confidence interval. HyLeak can also print the channel matrix and additional information; the full list of arguments is printed by `./hyleak -h`.

7.1. Illustrating Example: Random Walk

Consider the following random walk problem (modeled in Fig. 3).

The secret is the initial location of an agent, encoded by a single natural number representing an approximate distance from a given point, e.g. in meters. Then the agent takes a fixed number of steps. At each step the distance of the agent increases or decreases by 10 meters with the same probability. After this fixed number of random walk steps, the final location of the agent is revealed, and the attacker uses it to guess the initial location of the agent.

This problem is too complicated to analyze by precise analysis, because the analysis needs to explore every possible combination of random paths, amounting to an exponential number in the random walk steps. It is also intractable to analyze with a fully statistical approach, since there are hundreds of possible secret values and the program has to be simulated many times for each of them to sufficiently observe the agent's behavior.

As shown in Section 8, HyLeak's hybrid approach computes the leakage significantly faster than the fully precise analysis and more accurately than the fully statistical analysis.

7.2. Architecture

The HyLeak tool implementation consists of the following 4 steps. Steps 1 and 2 are implemented with different ANTLR parsers [Par07]. The implementation of Step 3 inherits some code from the QUAIL tool [BLTW, BLTW13, BLQ15] to employ QUAIL's optimization techniques for precise analysis, i.e., parallel analysis of execution traces and compact Markovian state representation.

Step 1: Preprocessing

Step 1a. Lexing, parsing and syntax checking. HyLeak starts by lexical analysis, macro substitution and syntax analysis. In macro substitution the constants defined in the input program are replaced with their declared values, and simple operations are resolved immediately. In the example in Fig. 3, this replaces the value of constant *MAX* on Line 24 with its declared value from Line 1. The tool checks whether the input program correctly satisfies the language syntax. In case of syntax errors, an error message describing the problem is produced and execution is terminated.

Step 1b. Loop unrolling and array expansion. `for` loops ranging over fixed intervals are unrolled to optimize the computation of variable ranges and thus program decomposition in Step 2. In the example in Fig. 3, the `for` loop in Line 24 gets replaced by a fixed number of repetitions of its code with increasing values of the variable *time*. Similarly, arrays are replaced with multiple variables indexed by their position number in the array. Note that these techniques are used only to optimize program decomposition and not required to compute the leakage in programs with arbitrary loops.

Step 2: Program Decomposition and Internal Code Generation

If a `simulate` or `simulate-abs` statement is present in the code, the program decomposition step is skipped and such statements are used to determine program decomposition.

The code may be decomposed only at conditional branching. Moreover, each component must be a terminal in the control flow graph, hence no component is executed afterwards. This is because the estimation method requires that the channel matrix for the system is the weighted sum of those for its components, and that the weight of a component is the probability of executing it, as explained in Sections 3.

The analysis method and its parameters for each component S_i are decided by estimating the computational cost of analyzing S_i. Let $\mathcal{Z}_i$ be the set of all *internal randomness* (i.e., the non-secret variables whose values are assigned according to probability distributions) in S_i. Then the cost of the statistical analysis is proportional to S_i's sub-channel

```
1  const MAX:=14;
2  secret int32 sec := [201,800];
3  observable int32 obs := 0;
4  public int32 loc := 0;
5  public int32 seed := 0;
6  public int32 ran := 0;
7  if sec ≤ 250// sec:[201,800]; TOT_OBS = 1 TOT_INT = 1; SEC DEP
8   then
9   |  loc := 200;
10 else if sec ≤ 350 then
11  |  loc := 300;
12 else if sec ≤ 450 then
13  |  loc := 400;
14 else if sec ≤ 550 then
15  |  loc := 500;
16 else if sec ≤ 650 then
17  |  loc := 600;
18 else if sec ≤ 750 then
19  |  loc := 700;
20 else
21  |  loc := 800;
22 end
23 simulate-abs;// loc: [200,800]; sec: [201,800]; TOT_OBS = 1; TOT_INT = 601
24 for time in [0,MAX] do
25  |  ran := random(0,9);
26  |  if ran ≤ 5 then
27  |  |  loc := loc + 10;
28  |  else
29  |  |  loc := loc - 10;
30  |  end
    |  // loc: [50,950]; ran: [0,9], TOT_OBS = 1; TOT_INT = 9010
31 end
32 obs := loc; // obs: [50,950]; TOT_OBS = 901; TOT_INT = 9010
33 return;
```

Fig. 3. Source code for the Random Walk illustrative example explained in Section 7.1. The comments show the estimates for the value ranges of some variables, as computed by HyLeak following Step 2 of Section 7.2, where `TOT_OBS` represent the estimate of the possible combinations of values of all observable variables and `TOT_INT` the estimate of the possible combinations of values of all internal variables. The red `simulate-abs` statement in Line 23 shows where the statement will be automatically added to implement the division in components, as explained in Step 2 of Section 7.2 and in more details in Section 7.3.

matrix size $\#\mathcal{X}_i \times \#\mathcal{Y}_i$, while the cost of the precise analysis is proportional to the number of all traces in S_i (in the worst case proportional to $\#\mathcal{X}_i \times \#\mathcal{Z}_i$). Hence the cost estimation is reduced to counting $\#\mathcal{Y}_i$ and $\#\mathcal{Z}_i$.

To obtain this, for each variable and each code line, an estimation of the number of possible values of the variable at the specific code line is computed. This is used to evaluate at each point in the input program whether it would be more expensive to use precise or statistical analysis. These estimations are shown as comments for different lines of the source code in Fig. 3. To reduce the computational cost of the estimation of variable ranges, we apply ad-hoc heuristics to obtain approximate estimates.

After determining the decomposition of the program, HyLeak automatically adds `simulate` and/or `simulate-abs` statements in the code to signal which parts of the input program should be analyzed with standard random sampling and with abstraction-then-sampling. For instance, since no annotations originally exist in the example source code in Fig. 3, HyLeak adds the `simulate-abs` statement (written in red) on Line 23. The procedure for decomposition is shown in Fig. 5 and is illustrated in Section 7.3 using the Random Walk example of Fig. 3. While the decomposition procedure is automated, it is a heuristic that does not guarantee to produce an optimal decomposition. Hence for usability the choice of analysis can be controlled by user's annotations on the code.

At the end, the input program is translated into a simplified internal language. Conditional statements and loops (`if`, `for`, and `while`) are rewritten into `if-goto` statements.

Step 3: Program Analysis

In this step the tool analyzes the executions of the program using the two approaches.

Step 3a. Precise analysis. The tool performs a depth-first symbolic execution of all possible execution traces of the input program, until it finds a `return`, `simulate`, or `simulate-abs` statement. When reaching a `return` statement the tool recognizes the symbolic path as terminated and stores its secret and output values. In the cases of `simulate` and `simulate-abs` statements it halts the symbolic path, saves the resulting program state, and schedules it for standard random sampling or for abstraction-then-sampling, respectively, starting from the saved program state. In the example in Fig. 3, the tool analyzes the code precisely and generates one symbolic path for each of the possible `if-elseif-else-end` statements from Line 7 to Line 22, so 7 symbolic paths in total (as shown in the control flow graph of the code in Fig. 4). Then each of the 7 symbolic paths meets the `simulate-abs` statement in Line 23, so it gets removed from precise analysis and scheduled for abstaction-then-sampling.

Step 3b. Statistical analysis. The tool performs all the statistical analyses and abstraction-then-sampling analyses, using the saved program states from Step 3a as starting point of each component to analyze statistically. The sample size for each simulation is automatically decided by using heuristics to have better accuracy with less sample size, as explained in Sections 4 and 5. The results of each analysis is stored as an appropriate joint probability sub-distribution between secret and observable values. In the example in Fig. 3, each of the 7 symbolic paths scheduled for abstraction-then-sampling gets analyzed with the technique. For each of the symbolic paths HyLeak choses a value of the secret and samples only that one, then applies the results for all secret values of the component. HyLeak recomputes the assignment of samples to the components for each 10% of the total samples, following the optimal sample sizes computed for abstraction-then-sampling components in Theorem 6.4.

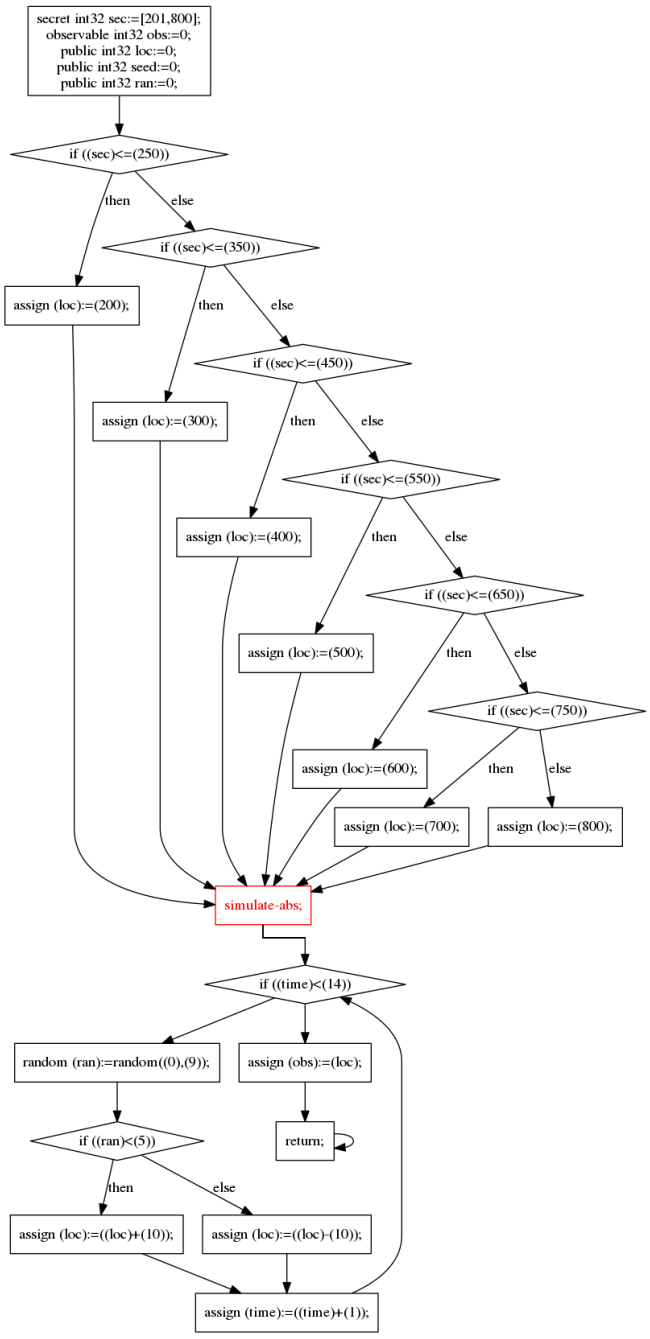


Fig. 4: Control flow graph for the input code of Fig. 3. The red node corresponds to the `simulate-abs` statement on Line 23 of Fig. 3. Each arrow entering the red node comes from a symbolic path with different secret and internal values, inducing a different component.

Step 4: Leakage Estimation

In this step the tool aggregates all the data collected by the precise and statistical analyses (performed in Steps 3) and estimates the Shannon leakage of the input program, together with evaluation of the estimation. This is explained in detail together with the program decomposition in Section 7.3.

1. Build the control flow graph of the system.
2. Mark all possible components based on each conditional branching. Each possible component must be a terminal as explained in Section 4.
3. For each possible component S_i, check whether it is deterministic or not (by syntactically checking an occurrence of a probabilistic assignment or a probabilistic function call). If it is, mark the component for precise analysis, since deterministic systems necessarily have a smaller number of traces.
4. For each possible component S_i, check whether S_i's output variables are independent of its input variables inside S_i (by *qualitative* information flow). If so, mark that the abstraction-then-sampling technique in Section 5.2 is to be used on the component, meaning that component S_i will be sampled on a single secret input value and the results will be applied to all secret values.
5. For each S_i, estimate an approximate range size $\#Z_i$ of its internal variables and $\#Y_i$ of its observable variables.
6. Looking from the leaves to the root of the control flow graph, estimate the cost of statistical and precise analyses, decide the decomposition into components, and mark each component for the cheaper analysis between the two. (For example, use a heuristics that marks each component S_i for precise analysis if $\#Z_i \leq \#X_i$ and for statistical analysis otherwise.)
7. Join together adjacent components if they are marked for precise analysis, or if they are marked for statistical analysis and have the same input and output ranges.
8. For each component, perform precise analysis or statistical analysis as marked.

Fig. 5. Procedure for implement the decomposition of a system in components given its source code described in Section 3. The actual implementation here uses the control flow graph of the system to guide the procedure.

7.3. On the Division into Components of the Random Walk Benchmark

In this section we briefly discuss how the Random Walk example in Figure 3 can be divided into components using the procedure in Figure 5. The procedure in Figure 5 shows in more detail how to implement the procedure described in Section 3. In the implementation, the construction of a the control flow graph of the system is used to guide the division in components.

The control flow graph generated by HyLeak is shown in Figure 4. Note that HyLeak has added a `simulate-abs` statement to the code, visible in the control flow graph.

The control flow graph in Fig. 4 helps understanding how HyLeak has implicitly divided the program into components. Note that the `simulate-abs` node marked in red has 7 in-edges. Each of these edges corresponds to a different symbolic path (i.e., a set of execution traces following the same edge), with different possible values for the secret variable `sec` and the internal variable `loc`. HyLeak has determined heuristically that at Line 23 the number of possible values of the observable variables (`TOT_OBS = 1`) is smaller than the number of possible values of internal variables (`TOT_INT = 601`), hence statistical simulation will be more efficient than precise analysis on these components. Also, in the code after line 23, HyLeak has determined that the values of the observable variables do not depend on the secret, hence each row of the sub-channel matrix for each of these components is identical, much like component S_1 in Fig. 2.

Hence, the abstraction-then-sampling technique of Section 5.2 can be applied, meaning that the behavior of each component will be simulated only for a single value of the secret in the set of possible secret values of the component, and the results will be applied to each row of the channel matrix.

Now that the analysis has gathered all the necessary information, HyLeak computes the leakage of the system under analysis. More specifically, it constructs an (approximate) joint posterior distribution of the secret and observable values of the input program from all the collected data produced by Step 3, as explained in Section 3. Then the tool estimates the Shannon leakage value from the joint distribution, including bias correction (see Sections 4 and 5). Finally, a 95% confidence interval for the estimated leakage value is computed to roughly evaluate the quality of the analysis.

In the example in Fig. 3, HyLeak outputs the prior Shannon entropy 8.9658, the posterior Shannon entropy 7.0428, the Shannon leakage (after bias correction) 1.9220, and the confidence interval [1.9214, 1.9226].

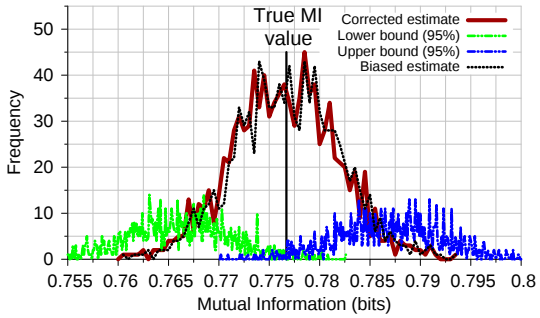


Fig. 6. Distribution of mutual information estimate and its confidence interval.

Secret \ Observable	0	1	2	3	4	5	6	7	8	9
0	0.2046	0.1102	0.0315	0.0529	0.1899	0.0064	0.0791	0.1367	0.0386	0.1501
1	0.0852	0.0539	0.1342	0.0567	0.1014	0.1254	0.0554	0.1115	0.0919	0.1844
2	0.1702	0.0542	0.0735	0.0914	0.0639	0.1322	0.1119	0.0512	0.1172	0.1343
3	0.0271	0.1915	0.0764	0.1099	0.0982	0.0761	0.0843	0.1364	0.0885	0.1116
4	0.0957	0.1977	0.0266	0.0741	0.1496	0.2177	0.0610	0.0617	0.0841	0.0318
5	0.0861	0.1275	0.1565	0.1193	0.1321	0.1716	0.0136	0.0984	0.0183	0.0766
6	0.0173	0.1481	0.1371	0.1037	0.1834	0.0271	0.1289	0.1690	0.0036	0.0818
7	0.0329	0.0825	0.0333	0.1622	0.1530	0.1378	0.0561	0.1479	0.0212	0.1731
8	0.1513	0.0435	0.0527	0.2022	0.0189	0.2159	0.0718	0.0063	0.1307	0.1067
9	0.0488	0.1576	0.1871	0.1117	0.1453	0.0349	0.0549	0.1766	0.0271	0.056

Fig. 7. Channel matrix for the experiments in Section 8.1.

8. Evaluation

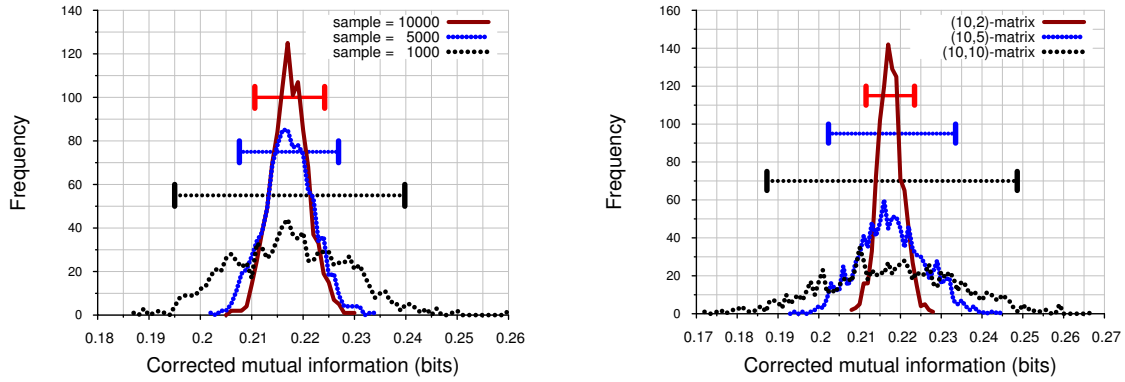
We evaluate experimentally the effectiveness of our hybrid method compared to the state of the art. We first discuss the cost and quality of the estimation, then test the hybrid method against fully precise/fully statistical analyses on Shannon leakage benchmarks.

8.1. On the Tradeoff between the Cost and Quality of Estimation

In the hybrid statistical estimation, the estimate takes different values probabilistically, because it is computed from a set of traces that are generated by executing a probabilistic system. Fig. 6 shows the sampling distribution of the mutual information estimate of the joint distribution in Fig. 1 in Section 1. The graph shows the frequency (on the y axis) of the mutual information estimates (on the x axis) when performing the estimation 1000 times. In each estimation we perform precise analysis on the component T and statistical analysis on S_1 and S_2 (with a sample size of 5000). The graph is obtained from 1000 samples each of which is generated by combining precise analysis on a component and statistical analysis on 2 components (using 5000 randomly generated traces). As shown in Fig. 6 the estimate after the correction of bias by Theorem 4.1 is closer to the true value. The estimate is roughly between the lower and upper bounds of the 95% confidence interval calculated using Theorem 4.2.

The interval size depends on the sample size in statistical analysis as shown in Fig. 8a. In Fig. 8a we illustrated the relationships between the size of the confidence interval and the sample size in the statistical analysis. We used an example with the randomly generated 10×10 channel matrix presented in Fig. 7 and the uniform prior. The graph shows the frequency (on the y axis) of the corrected mutual information estimates (on the x axis) that are obtained by estimating the mutual information value 1000, 5000 and 10000 times. When the sample size is k times larger then the confidence interval is $\sqrt{k}$ times narrower.

The interval size also depends on the amount of precise analysis as shown in Fig. 8b. If we perform precise analysis on larger components, then the sampling distribution becomes more centered (with shorter tails) and the confidence interval becomes narrower. For instance, in Fig. 8b we illustrated the relationships between the size of the confidence interval and the amount of precise analysis. The graph shows the frequency (on the y axis) of the corrected mutual information estimates (on the x axis) that are obtained by estimating the mutual information value 1000 times when statistical analysis is applied to a 10×2, 10×5 and 10×10 sub-matrix of the full 10×10 matrix. Using statistical analysis only on a smaller component (10×2 sub-matrix) yields a smaller confidence interval than using it on the whole system (10×10 matrix). More generally, if we perform precise analysis on larger components, then we have a smaller confidence interval. This means that the hybrid approach produces better estimates than the state of the art in statistical analysis. Due to the combination with precise analysis, the confidence interval estimated by our approach is smaller than LeakWatch [CKN14] for the same sample size.



(a) Estimates and sample sizes.

(b) Estimates and the ratio of precise analysis.

Fig. 8. Smaller intervals when increasing the sample size or the ratio of precise analysis.

8.2. Shannon Leakage Benchmarks

We compare the performance of our hybrid method with fully precise/statistical analysis on Shannon leakage benchmarks. Our implementations of precise and statistical analyses are variants of the state-of-the art tools QUAIL [BLTW13, BLTW] and LeakWatch [CKN14, CKNb] respectively. All experiments are performed on an Intel i7-3720QM 2.6GHz eight-core machine with 8GB of RAM running Fedora 21.

8.2.1. Random Walk

We analyze the random walk example presented in Section 7.1 for different values of number of steps MAX. We plot the computation times and the errors in leakage values computed by the three different methods in the graphs presented in Fig 10. These graph show again that the execution time of precise analysis grows exponentially to the number of steps MAX, while HyLeak and fully randomized analysis do not require much time even for large values of MAX. In the fully randomized analysis the error is always much larger than when using HyLeak.

8.2.2. Reservoir Sampling

```
1  const N; // number of elements
2  const K; // selection
3  secret array[N] of int1 s;
4  observable array[K] of int1 r;
5  public int32 j := 0;
6  for i in [0, K-1] do r[i] := s[i] ;
7  for i in [K, N-1] do
8  |   j := random(0,i);
9  |   if j<K then r[j] := s[i];
10 end
```

Fig. 9: Reservoir sampling.

The reservoir sampling problem [Vit85] consists of selecting K elements randomly from a pool of $N > K$ elements. We quantify the information flow of the commonly-used *Algorithm R* [Vit85], shown in Fig 9, for various values of N and $K = N/2$. In the algorithm, the first K elements are chosen as the sample, then each other element has a probability to replace one element in the sample. We plot the computation times and the errors in leakage values computed by the three different methods in the graphs presented in Fig 11. It shows that HyLeak hybrid approach performs faster than the full simulation approach and gives more precise results. Compared to the precise analysis, the hybrid approach run faster when increasing the model's complexity.

8.2.3. Multiple Lying Cryptographers Protocol

The lying cryptographers protocol is a variant of the dining cryptographer multiparty computation protocol [Cha88] in which a randomly-chosen cryptographer declares the opposite of what they would normally declare, i.e. they lie if they are not the payer, and do not lie if they are the payer. We consider three simultaneous lying cryptographers implementation in which 8 cryptographers run the protocol on three separate overlapping tables A, B and C with 4 cryptographers each. Table A hosts cryptographers 1 to 4, Table B hosts cryptographers 3 to 6, and Table C hosts cryptographers 5 to 8. The identity of the payer is the same in all tables.

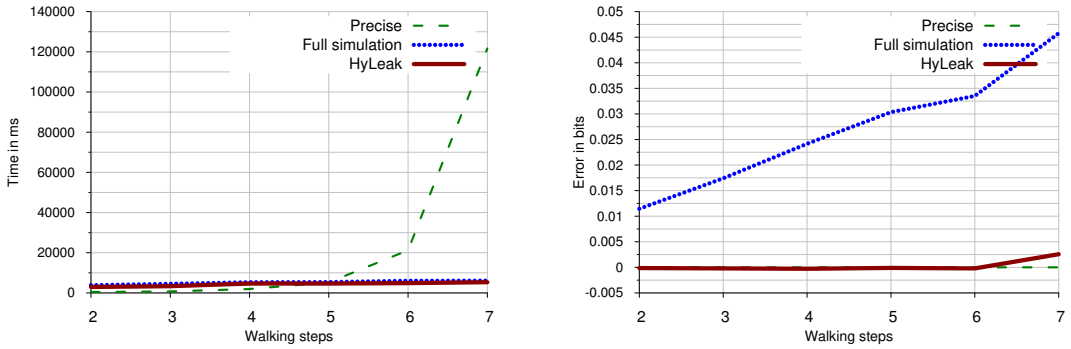


Fig. 10. Random walk experimental results.

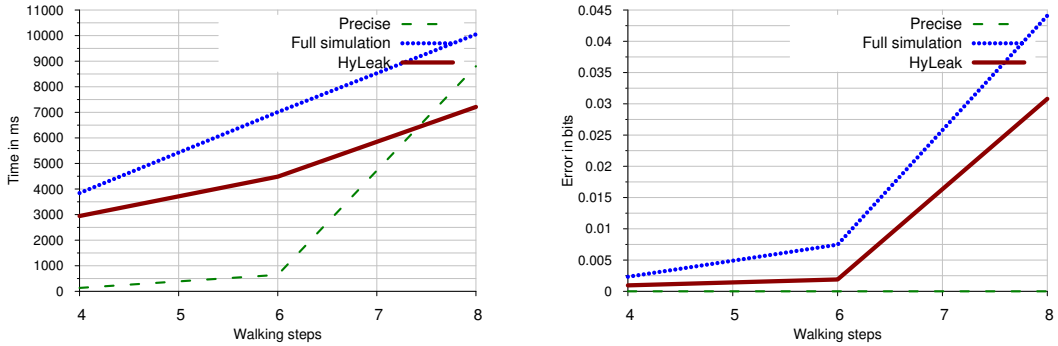


Fig. 11. Reservoir sampling experimental results.

```
1  const N:=4; // number of cryptographers at each table
2  const M:=8; // total number of cryptographers
   /* these bits represent the coin tosses for the three tables                    */
3  public array [N] of int1 coinA, coinB, coinC;
4  public int32 lies; // this is for the liar
   /* these bits represent the bits declared by the three cryptographers at each table */
5  public array [N] of int1 declA, declB, declC;
   /* these are the outputs at each table                                          */
6  observable int1 outputA := 0;
7  observable int1 outputB := 0;
8  observable int1 outputC := 0;
9  secret int32 h := [0, M]; // the secret has M+1 possible values
10 lies := random(1, M); for c in coinA do c := randombit(0.5);
11 for c in coinB do c:=randombit(0.5);
12 for c in coinC do c:=randombit(0.5);
13 for i in [0, N − 1] do
14     declA[i]:=coinA[i] xor coinA[(i + 1)%N];
15     if h == i + 1 then declA[i]:= ! declA[i];
16     if lies == i + 1 then declA[i]:= ! declA[i];
17     outputA := outputA xor declA[i];
18     declB[i]:=coinB[i] xor coinB[(i + 1)%N];
19     if h == i + 3 then declB[i]:= ! declB[i];;
20     if lies==i+3 then declB[i]:= ! declB[i];;
21     outputB := outputB xor declB[i];
22     declC[i] := coinC[i] xor coinC[(i + 1)%N];
23     if h == i + 5 then declC[i]:= ! declC[i];
24     if lies == i + 5 then declC[i]:= ! declC[i];
25     outputC := outputC xor declC[i];
26 end
```

Fig. 12. Multiple lying cryptographers.

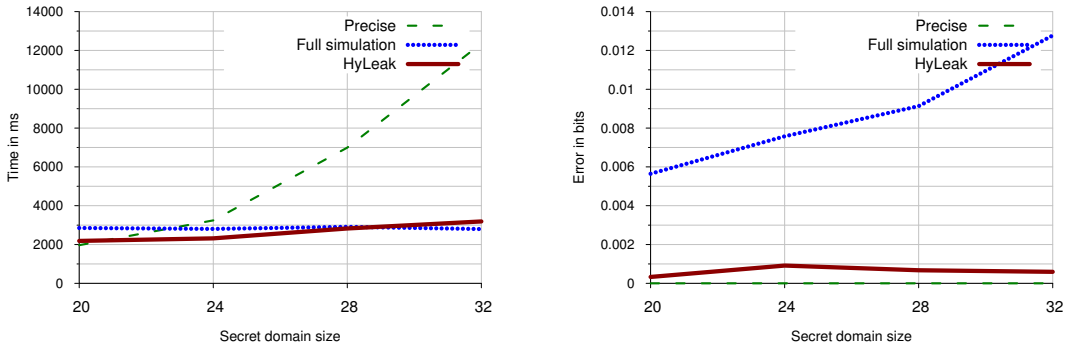


Fig. 13. Shifting window experimental results.

The division into components is executed following the principles in Section 7.3. The hybrid approach divides the protocol into 8 components, one for each possible liar. Then each component is analyzed statistically.

The results of the experiment are summarized in Table 3. Note that this example has some zeroes in the probabilities in the channel matrix, which makes the tool LeakWatch [CKN14, CKNb] compute an incorrect result due to incorrect bias estimation, as explained in Section 5.1. In fact, the leakage value obtained by LeakWatch using Corollary 5.1 is 0.36245, which is far from the correct value of 0.503 in Table 3. On the other hand, the error of our calculation using Proposition 5.2 is 1.87e-4 even with fully statistical analysis.

8.2.4. Shifting Window

```
1  const N; // number of elements
2  const W; // window size
3  secret int32 sec := [0,N-1];
4  observable int32 obs;
5  public int32 minS, sizeS, sizeS, minO,
     sizeO;
6  minS := random(0,N-W-1);
7  if sec ≥ minS then
8  |   sizeS := random(1,W);
9  |   if sec ≤ minS+sizeS then
10 |   |   minO := random(0,N-W-1);
11 |   |   sizeO := random(1,W);
12 |   |   obs := random(minO,minO+sizeO);
13 |   else
14 |   |   obs := random(0,N-1);
15 |   end
16 else
17 |   obs := random(0,N-1);
18 end
```

Fig. 14: Shifting Window.

In the Shifting Window example (Fig. 14) the secret `sec` can take N possible values, and an interval (called a "window") in the secret domain is randomly selected from 1 to W. If the value of the secret is inside the window, then another window is randomly chosen in a similar way and the program outputs a random value from this second window. Otherwise, the program outputs a random value over the secret domain.

In Fig. 13 we present the results of experiments on the shifting window when increasing the size of the secret domain. The execution time of precise analysis grows proportionally to the secret domain size N while HyLeak and fully randomized analysis do not require much time for a larger N. In the fully randomized analysis the error from the true value grows rapidly while in using HyLeak the error is much smaller.

8.2.5. Probabilistically Terminating Loop

```
1  const N; // number of secrets
2  const BOUND;
3  secret int32 sec := [0, N];
4  observable int32 obs;
5  public int32 time := 0;
6  public int2 terminate := 0;
7  public int32 rand;
8  while terminate ≠ 1 do
9  |   rand := random(1,N);
10 |   if sec ≤ rand then terminate := 1;
11 |   time := time+1;
12 end
13 if time < BOUND then
14 |   obs := time;
15 else
16 |   obs := BOUND;
17 end
```

Fig. 15: Probabilistically Terminating Loop.

The tool HyLeak can analyze programs that terminate only probabilistically. For instance, the program shown in Fig. 15 has a loop that terminates depending on the randomly generated value of the variable `rand`. No previous work has presented an automatic measurement of information leakage by probabilistic termination, as precise analysis cannot handle non-terminating programs, which typically causes non-termination of the analysis of the program. On the other hand, the stochastic simulation of this program supported in HyLeak terminates after some number of iterations in practice although it may take long for some program executions to terminate.
We analyze this model with the hybrid and full simulation approaches only, as the precise analysis does not terminate. The results are given in Table 3. It shows that also with this problem the hybrid approach performs faster than the full simulation approach.

8.2.6. Smart Grid Privacy

A smart grid is an energy network where users (like households) may consume or produce energy. In Fig. 16 we describe a simple model of a smart grid using the HyLeak language. This example is taken from [BLQ15]. The users periodically negotiate with a central aggregator in charge of balancing the total consumption among several users. In practice each user declares to the aggregator its consumption plan. The aggregator sums up the consumptions of the users and checks if it falls within admitted bounds. If not it answers to the users that the consumption is too low or too high by a certain amount, such that they adapt their demand. This model raises some privacy issues as some attacker can try to guess the consumption of a user, and for instance infer whether or not this particular user is at home.

In Fig. 17 we present the experiment results of this smart grid example for different numbers of users. HyLeak takes less time than both fully precise analysis and fully randomized analysis (as shown in the left figure). Moreover it is closer to the true value than fully randomized analysis especially when the number of users is larger (as shown in the right figure).

8.2.7. Benchmarks results

In Table 3 we show the results of all the benchmarks using fully precise, fully statistical and hybrid analyses, for a sample size of 50000 executions. Timeout is set at 10 minutes.

The results in Table 3 show the superiority of our hybrid approach compared to the state of the art. The hybrid analysis scales better than the precise analysis, since it does not need to analyze every trace of the system. Compared to fully statistical analysis, our hybrid analysis exploits precise analysis on components of the system where statistical estimation would be more expensive than precise analysis. This allows the hybrid analysis to focus the statistical estimation on components of the system where it converges faster, thus obtaining a smaller confidence interval in a shorter time.

9. Conclusions and Future Work

We have proposed a hybrid statistical estimation method for estimating mutual information by combining precise and statistical analysis, and for compositionally computing the bias and accuracy of the estimate. This naturally extends to the computation of Shannon entropy and conditional Shannon entropy, generalizing previous approaches on computing mutual information. The method automatically decomposes a system into components and determines which type of analysis is better for each component according to the components' properties.

We have also introduced an algorithm to adaptively find the optimal sample sizes for different components in the statistical analysis to minimize their variance and produce a more accurate estimate given a sample size. Moreover, we have presented how to reduce sample sizes by using prior knowledge about systems, including the abstraction-then-sampling technique with qualitative analysis. We haven shown how to leverage this information on the system to reduce the computation time and error of the estimation.

```
1  const N:=9; // the total number of users
2  const S; // the number of users we care about
3  const C:=3; // the possible consumptions level
4  const M:=0; // the consumption level of the attacker
5  const LOWT:=2; // the lower threshold
6  const HIGHT:=9; // the upper threshold
   /* the observable is the order given by the control system                    */
7  observable int32 order;
8  observable int1 ordersign;
   /* the secret is the consumption of each user we care about                   */
9  secret array [S] of int32 secretconsumption := [0, C-1];
   /* the other consumptions are just private                                    */
10 private array [N-(S+1)] of int32 privateconsumption := [0,C-1];
11 public int32 total := M; // this is the projected consumption
   /* count the secret consumptions                                              */
12 for i in [0,S-1] do
13 |   for j in [0,C-1] do
14 |   |   if secretconsumption[i] == j then total := total + j;
15 |   end
16 end
   /* count the private consumptions                                             */
17 for i in [0,N-S-1] do
18 |   for j in [0,C-1] do
19 |   |   if privateconsumption[i] == j then total := total + j;
20 |   end
21 end
22 if total < LOWT then
23 |   order := LOWT - total;
24 |   ordersign := 0;
25 else if total > HIGHT then
26 |   order := total - HIGHT;
27 |   ordersign := 1;
28 else
29 |   order := 0;
30 |   ordersign := 0;
31 end
```

Fig. 16. Smart Grid Example.

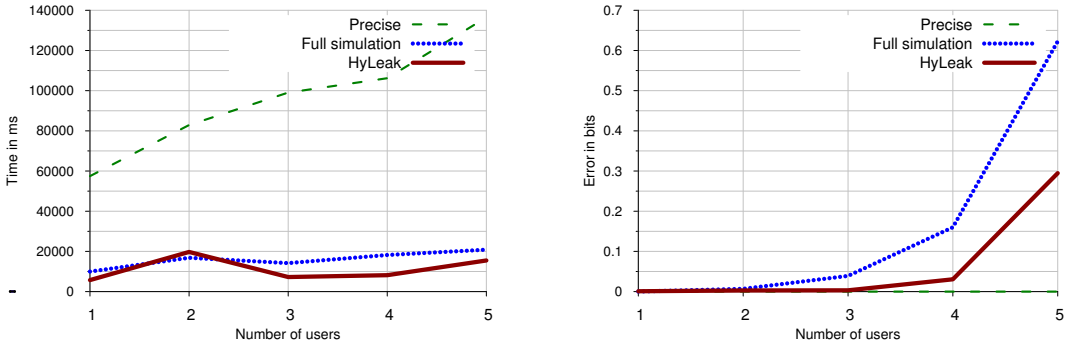


Fig. 17. Smart grid experimental results.

We have provided an implementation in the freely available tool HyLeak. We have shown both theoretical and experimental results to demonstrate that the proposed approach and implementation outperform the state of the art.

Future work includes developing theory and tools that extend our hybrid method to the analysis of other properties and integrate further symbolic abstraction techniques into our estimation method. Another possible application of the hybrid analysis is to compute information leakage among adaptive agents in the game-theoretic framework [ACKP17], in which each agent probabilistically chooses a strategy that is modeled as a component of a channel.

		Precise			Statistical			Hybrid		
		Time(s)	**Leakage**	**Error**	**Time(s)**	**Leakage**	**Error**	**Time(s)**	**Leakage**	**Error**
Random walk	**N=2**	0.467	2.17	0	3.85	2.19	1.15e-2	2.97	2.17	1.37e-4
	N=3	0.748	2.17	0	4.51	2.19	1.74e-2	3.35	2.17	2.05e-4
	N=4	1.93	2.17	0	5.34	2.2	2.42e-2	4.66	2.17	2.74e-4
	N=5	5.25	2.14	0	5.4	2.17	3.03e-2	4.66	2.14	1.14e-4
	N=6	21.2	2.11	0	6.05	2.14	3.35e-2	4.87	2.11	2.03e-4
	N=7	122	2.07	0	6.14	2.12	4.58e-2	5.34	2.07	2.57e-4
Reservoir	**N=4**	0.134	0.732	0	3.84	0.734	2.33e-3	2.95	0.731	9.55e-4
	N=6	0.645	0.918	0	7	0.926	7.47e-3	4.48	0.917	1.89e-3
	N=8	8.8	1.1	0	10.1	1.14	4.41e-2	7.21	1.13	3.08e-2
	N=10	timeout	n/a	n/a	15.7	1.62	n/a	11.1	1.61	n/a
	N=12	timeout	n/a	n/a	27.7	3.02	n/a	20.7	3.01	n/a
Lying crypto.		397	0.503	0	106	0.503	1.87e-4	78.8	0.503	1.37e-6
Shifting window	**N=20**	1.97	1.51e-2	0	2.85	2.08e-2	5.65e-3	2.19	1.48e-2	3.31e-4
	N=24	3.24	1.46e-2	0	2.81	2.21e-2	7.58e-3	2.32	1.55e-2	9.16e-4
	N=28	6.99	1.42e-2	0	2.91	2.33e-2	9.13e-3	2.83	1.35e-2	6.75e-4
	N=32	12.4	1.38e-2	0	2.8	2.66e-2	1.28e-2	3.19	1.33e-2	5.93e-4
Probabilistic termination	**N=5**	n/a	n/a	n/a	4.14	0.424	n/a	2.76	0.432	n/a
	N=7	n/a	n/a	n/a	4.13	0.455	n/a	3.08	0.454	n/a
	N=9	n/a	n/a	n/a	4.43	0.472	n/a	3.71	0.473	n/a
Smart grid	**S=1**	57.5	8.49e-2	0	9.96	8.51e-2	2.31e-4	5.74	8.59e-2	9.43e-4
	S=2	82.9	0.181	0	16.8	0.188	6.82e-3	19.7	0.178	2.85e-3
	S=3	99	0.293	0	14.2	0.332	3.9e-2	7.23	0.296	3.16e-3
	S=4	106	0.425	0	18.2	0.585	0.16	8.16	0.455	3.06e-2
	S=5	136	0.587	0	20.9	1.21	0.623	15.5	0.882	0.295

Table 3. Shannon leakage benchmark results using the three different methods (precise, full simulation and hybrid). The results contain the time (in seconds) taken for the analysis, the value of the leakage (in bits), and the error (in bits) compared to the true result, when the true result has been computed with the precise analysis. The result n/a means either that the experiment cannot be performed on this example, which is the case for the precise analysis of the probabilistic terminating loop, or that the error cannot be computed because the precise analysis was not successful.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number JP17K12667, by JSPS and Inria under the Japan-France AYAME Program, by the MSR-Inria Joint Research Center, by the Sensation European grant, and by région Bretagne.

References

[ACKP17] Mário S. Alvim, Konstantinos Chatzikokolakis, Yusuke Kawamoto, and Catuscia Palamidessi. Information leakage games. In *8th International Conference on Decision and Game Theory for Security (GameSec 2017)*, volume 10575 of *Lecture Notes in Computer Science*. Springer, 2017.

[Ada04] Christoph Adami. Information theory in molecular biology. *Physics of Life Reviews*, 1(1):3–22, April 2004.

[BHP12] Benoît Barbot, Serge Haddad, and Claudine Picaronny. Coupling and importance sampling for statistical model checking. In Cormac Flanagan and Barbara König, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*, volume 7214 of *Lecture Notes in Computer Science*, pages 331–346. Springer, 2012.

[BK11] Gilles Barthe and Boris Köpf. Information-theoretic bounds for differentially private mechanisms. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011*, pages 191–204. IEEE Computer Society, 2011.

[BKLT] Fabrizio Biondi, Yusuke Kawamoto, Axel Legay, and Louis-Marie Traonouez. HyLeak. https://project.inria.fr/hyleak/.

[BKLT17] Fabrizio Biondi, Yusuke Kawamoto, Axel Legay, and Louis-Marie Traonouez. Hyleak: Hybrid analysis tool for information leakage. In *15th International Symposium on Automated Technology for Verification and Analysis (ATVA'17)*, volume 10482 of *Lecture Notes in Computer Science*. Springer, 2017.

[BKR09] Michael Backes, Boris Köpf, and Andrey Rybalchenko. Automatic discovery and quantification of information leaks. In *30th IEEE Symposium on Security and Privacy (S&P 2009), 17-20 May 2009, Oakland, California, USA*, pages 141–153. IEEE Computer Society, 2009.

[BLMW15] Fabrizio Biondi, Axel Legay, Pasquale Malacaria, and Andrzej Wasowski. Quantifying information leakage of randomized protocols. *Theor. Comput. Sci.*, 597:62–87, 2015.

[BLQ15] Fabrizio Biondi, Axel Legay, and Jean Quilbeuf. Comparative analysis of leakage tools on scalable case studies. In Bernd Fischer and Jaco Geldenhuys, editors, *Model Checking Software - 22nd International Symposium, SPIN 2015, Stellenbosch, South Africa, August 24-26, 2015, Proceedings*, volume 9232 of *Lecture Notes in Computer Science*, pages 263–281. Springer, 2015.

[BLTW] Fabrizio Biondi, Axel Legay, Louis-Marie Traonouez, and Andrzej Wasowski. QUAIL. https://project.inria.fr/quail/.

[BLTW13] Fabrizio Biondi, Axel Legay, Louis-Marie Traonouez, and Andrzej Wasowski. QUAIL: A quantitative security analyzer for imperative code. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 702–707. Springer, 2013.

[BP14] Michele Boreale and Michela Paolini. On formally bounding information leakage by statistical estimation. In Sherman S. M. Chow, Jan Camenisch, Lucas Chi Kwong Hui, and Siu-Ming Yiu, editors, *Information Security - 17th International Conference, ISC 2014, Hong Kong, China, October 12-14, 2014. Proceedings*, volume 8783 of *Lecture Notes in Computer Science*, pages 216–236. Springer, 2014.

[Bri04] D. R. Brillinger. Some data analysis using mutual information. *Brazilian Journal of Probability and Statistics*, 18(6):163–183, 2004.

[CCG10] Konstantinos Chatzikokolakis, Tom Chothia, and Apratim Guha. Statistical measurement of information leakage. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 390–404. Springer, 2010.

[CFM+15] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. On parallel scalable uniform SAT witness generation. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9035 of *Lecture Notes in Computer Science*, pages 304–319. Springer, 2015.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[CHM01] David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative analysis of the leakage of confidential data. *Electr. Notes Theor. Comput. Sci.*, 59(3):238–251, 2001.

[CHM07] David Clark, Sebastian Hunt, and Pasquale Malacaria. A static analysis for quantifying information flow in a simple imperative language. *Journal of Computer Security*, 15(3):321–371, 2007.

[CK14] Tom Chothia and Yusuke Kawamoto. Statistical estimation of min-entropy leakage, April 2014. Manuscript.

[CKNa] Tom Chothia, Yusuke Kawamoto, and Chris Novakovic. leakiEst. http://www.cs.bham.ac.uk/research/projects/infotools/leakiest/.

[CKNb] Tom Chothia, Yusuke Kawamoto, and Chris Novakovic. LeakWatch. http://www.cs.bham.ac.uk/research/projects/infotools/leakwatch/.

[CKN13] Tom Chothia, Yusuke Kawamoto, and Chris Novakovic. A tool for estimating information leakage. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pages 690–695, 2013.

[CKN14] Tom Chothia, Yusuke Kawamoto, and Chris Novakovic. Leakwatch: Estimating information leakage from java programs. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II*, volume 8713 of *Lecture Notes in Computer Science*, pages 219–236. Springer, 2014.

[CKNP13] Tom Chothia, Yusuke Kawamoto, Chris Novakovic, and David Parker. Probabilistic point-to-point information leakage. In *2013 IEEE 26th Computer Security Foundations Symposium, New Orleans, LA, USA, June 26-28, 2013*, pages 193–205. IEEE Computer Society, 2013.

[CMS14] Rohit Chadha, Umang Mathur, and Stefan Schwoon. Computing information flow using symbolic model-checking. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 505–516. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

[CMV13] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. A scalable approximate model counter. In Christian Schulte, editor, *Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, pages 200–216. Springer, 2013.

[CPP08] Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Inf. Comput.*, 206(2-4):378–401, 2008.

[CS10] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory (2. ed.)*. A Wiley-Interscience publication. Wiley, 2006.

[CZ11] Edmund M. Clarke and Paolo Zuliani. Statistical model checking for cyber-physical systems. In Tevfik Bultan and Pao-Ann Hsiung, editors, *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011, Taipei, Taiwan, October 11-14, 2011. Proceedings*, volume 6996 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2011.

[Den76] Dorothy E. Denning. A lattice model of secure information flow. *Commun. ACM*, 19(5):236–243, 1976.

[ES13] Barbara Espinoza and Geoffrey Smith. Min-entropy as a resource. *Inf. Comput.*, 226:57–75, 2013.

[ESB09] Francisco Escolano, Pablo Suau, and Boyn Bonev. *Information Theory in Computer Vision and Pattern Recognition*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[FS14] Daniel J. Fremont and Sanjit A. Seshia. Speeding up smt-based quantitative program analysis. In Philipp Rümmer and Christoph M. Wintersteiger, editors, *Proceedings of the 12th International Workshop on Satisfiability Modulo Theories, SMT 2014, affiliated with the 26th International Conference on Computer Aided Verification (CAV 2014), the 7th International Joint Conference on Automated Reasoning (IJCAR 2014), and the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT 2014), Vienna, Austria, July 17-18, 2014.*, volume 1163 of *CEUR Workshop Proceedings*, pages 3–13. CEUR-WS.org, 2014.

[Gal68] Robert G. Gallager. *Information Theory and Reliable Communication*. John Wiley & Sons, Inc., New York, NY, USA, 1968.

[III91] James W. Gray III. Toward a mathematical foundation for information flow security. In *IEEE Symposium on Security and Privacy*, pages 21–35, 1991.

[Jen96] Finn V. Jensen. *Introduction to Bayesian Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1996.

[KB07] Boris Köpf and David A. Basin. An information-theoretic model for adaptive side-channel attacks. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 286–296. ACM, 2007.

[KBL16] Yusuke Kawamoto, Fabrizio Biondi, and Axel Legay. Hybrid statistical estimation of mutual information for quantifying information flow. In John S. Fitzgerald, Constance L. Heitmeyer, Stefania Gnesi, and Anna Philippou, editors, *FM 2016: Formal Methods - 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings*, volume 9995 of *Lecture Notes in Computer Science*, pages 406–425, 2016.

[KCP14] Yusuke Kawamoto, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Compositionality results for quantitative information flow. In Gethin Norman and William H. Sanders, editors, *Quantitative Evaluation of Systems - 11th International Conference, QEST 2014, Florence, Italy, September 8-10, 2014. Proceedings*, volume 8657 of *Lecture Notes in Computer Science*, pages 368–383. Springer, 2014.

[KCP17] Yusuke Kawamoto, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. On the compositionality of quantitative information flow. *Logical Methods in Computer Science*, 13(3:11):1–31, 2017.

[KG15] Yusuke Kawamoto and Thomas Given-Wilson. Quantitative information flow for scheduler-dependent systems. In Nathalie Bertrand and Mirco Tribastone, editors, *Proceedings Thirteenth Workshop on Quantitative Aspects of Programming Languages and Systems, QAPL 2015, London, UK, 11th-12th April 2015.*, volume 194 of *EPTCS*, pages 48–62, 2015.

[KMPS11] Min Gyung Kang, Stephen McCamant, Pongsin Poosankam, and Dawn Song. DTA++: dynamic taint analysis with targeted control-flow propagation. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society, 2011.

[KR10] Boris Köpf and Andrey Rybalchenko. Approximation and randomization for quantitative information-flow analysis. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*, pages 3–14. IEEE Computer Society, 2010.

[LCFS14] Zicong Liu, Zhenyu Chen, Chunrong Fang, and Qingkai Shi. Hybrid test data generation. In Pankaj Jalote, Lionel C. Briand, and André van der Hoek, editors, *36th International Conference on Software Engineering, ICSE '14, Companion Proceedings, Hyderabad, India, May 31 - June 07, 2014*, pages 630–631. ACM, 2014.

[LDB10] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In Howard Barringer, Yliès Falcone, Bernd Finkbeiner, Klaus Havelund, Insup Lee, Gordon J. Pace, Grigore Rosu, Oleg Sokolsky, and Nikolai Tillmann, editors, *Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings*, volume 6418 of *Lecture Notes in Computer Science*, pages 122–135. Springer, 2010.

[Mac02] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002.

[Mal07] Pasquale Malacaria. Assessing security threats of looping constructs. In Martin Hofmann and Matthias Felleisen, editors, *Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2007, Nice, France, January 17-19, 2007*, pages 225–235. ACM, 2007.

[ME08] Stephen McCamant and Michael D. Ernst. Quantitative information flow as network flow capacity. In Rajiv Gupta and Saman P. Amarasinghe, editors, *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, AZ, USA, June 7-13, 2008*, pages 193–205. ACM, 2008.

[Mod89] R. Moddemeijer. On estimation of entropy and mutual information of continuous distributions. *Signal Processing*, 16:233–248, 1989.

[MS07] Rupak Majumdar and Koushik Sen. Hybrid concolic testing. In *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007*, pages 416–426. IEEE Computer Society, 2007.

[NMS09] James Newsome, Stephen McCamant, and Dawn Song. Measuring channel capacity to distinguish undue influence. In Stephen Chong and David A. Naumann, editors, *Proceedings of the 2009 Workshop on Programming Languages and Analysis for Security, PLAS 2009, Dublin, Ireland, 15-21 June, 2009*, pages 73–85. ACM, 2009.

[Par07] Terence Parr. *The Definitive ANTLR Reference: Building Domain Specific Languages*. 2007.

[PM14] Quoc-Sang Phan and Pasquale Malacaria. Abstract model counting: a novel approach for quantification of information leaks. In Shiho Moriai, Trent Jaeger, and Kouichi Sakurai, editors, *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 283–292. ACM, 2014.

[PMPd14] Quoc-Sang Phan, Pasquale Malacaria, Corina S. Pasareanu, and Marcelo d'Amorim. Quantifying information leaks using reliability analysis. In Neha Rungta and Oksana Tkachuk, editors, *2014 International Symposium on Model Checking of Software, SPIN 2014, Proceedings, San Jose, CA, USA, July 21-23, 2014*, pages 105–108. ACM, 2014.

[PMTP12] Quoc-Sang Phan, Pasquale Malacaria, Oksana Tkachuk, and Corina S. Pasareanu. Symbolic quantitative information flow. *ACM SIGSOFT Software Engineering Notes*, 37(6):1–5, 2012.

[Smi09] Geoffrey Smith. On the foundations of quantitative information flow. In Luca de Alfaro, editor, *Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5504 of *Lecture Notes in Computer Science*, pages 288–302. Springer, 2009.

[VEB+16] Celina G. Val, Michael A. Enescu, Sam Bayless, William Aiello, and Alan J. Hu. Precisely measuring quantitative information flow: 10k lines of code and beyond. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 31–46. IEEE, 2016.

[Vit85] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985.

[Wei16] Alexander Weigl. Efficient sat-based pre-image enumeration for quantitative information flow in programs. In Giovanni Livraga, Vicenç Torra, Alessandro Aldini, Fabio Martinelli, and Neeraj Suri, editors, *Data Privacy Management and Security Assurance - 11th International Workshop, DPM 2016 and 5th International Workshop, QASA 2016, Heraklion, Crete, Greece, September 26-27, 2016, Proceedings*, volume 9963 of *Lecture Notes in Computer Science*, pages 51–58. Springer, 2016.

[Wil13] Mark M. Wilde. *Quantum Information Theory*. Cambridge University Press, New York, NY, USA, 1st edition, 2013.

[YT14] Hirotoshi Yasuoka and Tachio Terauchi. Quantitative information flow as safety and liveness hyperproperties. *Theor. Comput. Sci.*, 538:167–182, 2014.

A. Proofs

In this section we present the detailed proofs of our results.

Hereafter we denote by Q the joint sub-distribution obtained by summing Q_j's:

$$Q[x,y] \stackrel{\text{def}}{=} \sum_{j \in \mathcal{J}} Q_j[x,y] \quad .$$

We write q_{xy} to denote $Q[x,y]$ for abbreviation. Then q_{xy} is the probability that the execution of the system $\mathcal{S}$ yields one of T_j's and has input x and output y.

A.1. Proofs for the Mean Estimation Using the Abstraction-Then-Sampling

In this section we present the proof for Theorem 5.6 in Section 5.2, i.e., the result on mean estimation using the abstraction-then-sampling. To show the theorem we present and prove Propositions A.1, A.2, and A.3 below.

First, recall that $\mathcal{D}$ is defined as the set of pairs consisting of inputs and outputs that appear with non-zero probabilities in the execution of the whole system $\mathcal{S}$:

$$\mathcal{D} = \{(x,y) \in \mathcal{X} \times \mathcal{Y} : P_{XY}[x,y] > 0\} \ .$$

Recall also that $\mathcal{I} = \{1,2,\ldots,m\}$. Let $\mathcal{I}^\star = \{1,2,\ldots,m'\}$ for $m' \leq m$.

Proposition A.1 (Mean of joint entropy estimated using the abstraction-then-sampling). The expected value $\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X;Y)\Big]$ of the estimated joint entropy is given by:

$$\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X,Y)\Big] = H(X,Y) - \sum_{i \in \mathcal{I} \setminus \mathcal{I}^\star} \frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}} \varphi_{ixy}\Big) - \sum_{i \in \mathcal{I}^\star} \frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}} \psi_{ixy}\Big) + \mathcal{O}(n_i^{-2})$$

where $\varphi_{ixy} = \frac{D_i[x,y] - D_i[x,y]^2}{P_{XY}[x,y]}$ and $\psi_{ixy} \stackrel{\text{def}}{=} \frac{D_i[x,y]\pi_i[x] - D_i[x,y]^2}{P_{XY}[x,y]}$.

Proof. We use notations that we have introduced in the previous proofs. For each $i \in \mathcal{I}$, let $\mathcal{X}_i$ be the set of the elements of $\mathcal{X}$ that appear with non-zero probabilities in the component S_i.

As explained in Section 5.2 we apply the standard sampling technique (shown in Section 4) to the components S_i with $i \in \mathcal{I} \setminus \mathcal{I}^\star$, and the abstraction-then-sampling technique to the components S_i with $i \in \mathcal{I}^\star$. We briefly recall the two techniques below.

Using the standard sampling technique we compute the empirical sub-distribution $\hat{R}_i$ for S_i with $i \in \mathcal{I} \setminus \mathcal{I}^\star$ as follows. The analyst first runs S_i a certain number n_i of times to obtain the set of execution traces. Let K_{ixy} be the number of traces that have input $x \in \mathcal{X}$ and output $y \in \mathcal{Y}$. Then $n_i = \sum_{x\in\mathcal{X}, y\in\mathcal{Y}} K_{ixy}$. From these numbers K_{ixy} of traces we compute the empirical joint (full) distribution $\hat{D}_i$ of X and Y by:

$$\hat{D}_i[x,y] \stackrel{\text{def}}{=} \frac{K_{ixy}}{n_i} \ .$$

Since S_i is executed with probability θ_i, the sub-distribution $\hat{R}_i$ is given by $\hat{R}_i[x,y] \stackrel{\text{def}}{=} \theta_i \hat{D}_i[x,y] = \frac{\theta_i K_{ixy}}{n_i}$.

On the other hand, we use the abstraction-then-sampling sampling technique to compute the empirical sub-distribution $\hat{R}_i$ for S_i with $i \in \mathcal{I}^\star$ as follows. Recall that for each $i \in \mathcal{I}^\star$, $\pi_i[x]$ is the probability of having an input x in the component S_i. For each $i \in \mathcal{I}^\star$ all the non-zero rows of S_i's channel matrix are the same conditional distribution; i.e., for each $x, x' \in \mathcal{X}_i$ and $y \in \mathcal{Y}_i$, $\frac{P_{XY}[x,y]}{\pi_i[x]} = \frac{P_{XY}[x',y]}{\pi_i[x']}$ when $\pi_i[x] \neq 0$ and $\pi_i[x'] \neq 0$. Therefore it is sufficient to estimate only one of the rows. We execute the component S_i with an identical input $x \in \mathcal{X}$ n_i times to record the traces. Let $K_{i\cdot y}$ be the number of traces of the component S_i that outputs y. Then we define the empirical joint (full) distribution $\hat{D}_i$ of X and Y as:

$$\hat{D}_i[x,y] \stackrel{\text{def}}{=} \frac{\pi_i[x] K_{i\cdot y}}{n_i}.$$

Since S_i is executed with probability θ_i, the sub-distribution $\hat{R}_i$ is given by: $\hat{R}_i[x,y] \stackrel{\text{def}}{=} \theta_i \hat{D}_i[x,y] = \dfrac{\theta_i \pi_i[x] K_{i\cdot y}}{n_i}$.

Now the empirical joint probability distribution $\hat{P}_{XY}$ is computed from the above empirical sub-distributions $\hat{R}_i$ (obtained either by standard sampling or by abstraction-then-sampling) and the exact sub-distributions Q_j (obtained by precise analysis):

$$\hat{P}_{XY}[x,y] = q_{xy} + \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} \frac{\theta_i K_{ixy}}{n_i} + \sum_{i\in\mathcal{I}^\star} \frac{\theta_i \pi_i[x] K_{i\cdot y}}{n_i}. \tag{1}$$

Let $\boldsymbol{K}_{xy} = (K_{1\cdot y}, K_{2\cdot y}, \ldots, K_{m'\cdot y}, K_{(m'+1)xy}, K_{(m'+2)xy}, \ldots, K_{mxy})$, and $f_{xy}(\boldsymbol{K}_{xy})$ be the m-ary function:

$$f_{xy}(\boldsymbol{K}_{xy}) = \Big(q_{xy} + \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} \tfrac{\theta_i K_{ixy}}{n_i} + \sum_{i\in\mathcal{I}^\star} \tfrac{\theta_i \pi_i[x] K_{i\cdot y}}{n_i}\Big) \log\Big(q_{xy} + \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} \tfrac{\theta_i K_{ixy}}{n_i} + \sum_{i\in\mathcal{I}^\star} \tfrac{\theta_i \pi_i[x] K_{i\cdot y}}{n_i}\Big)$$

which equals $\hat{P}_{XY}[x,y] \log \hat{P}_{XY}[x,y]$. Then the empirical joint entropy is:

$$\hat{H}_{\mathcal{I}^\star}(X,Y) = -\sum_{(x,y)\in\mathcal{D}} \hat{P}_{XY}[x,y] \log \hat{P}_{XY}[x,y] = -\sum_{(x,y)\in\mathcal{D}} f_{xy}(\boldsymbol{K}_{xy}).$$

Let $\overline{K_{ixy}} = \mathbb{E}[K_{ixy}]$ for each $i \in \mathcal{I}$ and $\overline{\boldsymbol{K}_{xy}} = \mathbb{E}[\boldsymbol{K}_{xy}]$. Then $\overline{K_{ixy}} = n_i D_i[x,y] = \frac{n_i R_i[x,y]}{\theta_i}$, and $\overline{K_{i\cdot y}} = \frac{n_i D_i[x,y]}{\pi_i[x]} = \frac{n_i R_i[x,y]}{\theta_i \pi_i[x]}$. By the Taylor expansion of $f_{xy}(\boldsymbol{K}_{xy})$ (w.r.t. the multiple dependent variables $\boldsymbol{K}_{xy}$) at $\overline{\boldsymbol{K}_{xy}}$, we have:

$$\begin{aligned} f_{xy}(\boldsymbol{K}_{xy}) = {} & f_{xy}(\overline{\boldsymbol{K}_{xy}}) + \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} \tfrac{\partial f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{ixy}} (K_{ixy} - \overline{K_{ixy}}) + \sum_{i\in\mathcal{I}^\star} \tfrac{\partial f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{i\cdot y}} \pi_i[x] (K_{i\cdot y} - \overline{K_{i\cdot y}}) \\ & + \sum_{i,j\in\mathcal{I}\setminus\mathcal{I}^\star} \frac{1}{2} \tfrac{\partial^2 f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{ixy} \partial K_{jxy}} (K_{ixy} - \overline{K_{ixy}})(K_{jxy} - \overline{K_{jxy}}) + \sum_{\substack{i\in\mathcal{I}\setminus\mathcal{I}^\star \\ j\in\mathcal{I}^\star}} \frac{1}{2} \tfrac{\partial^2 f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{ixy} \partial K_{j\cdot y}} \pi_j[x] (K_{ixy} - \overline{K_{ixy}})(K_{j\cdot y} - \overline{K_{j\cdot y}}) \\ & + \sum_{i,j\in\mathcal{I}^\star} \frac{1}{2} \tfrac{\partial^2 f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{i\cdot y} \partial K_{j\cdot y}} \pi_i[x] \pi_j[x] (K_{i\cdot y} - \overline{K_{i\cdot y}})(K_{j\cdot y} - \overline{K_{j\cdot y}}) + \mathcal{O}(\boldsymbol{K}_{xy}^3) \end{aligned}$$

To compute the expected value $\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X,Y)\Big]$ of the estimated joint entropy, it should be noted that:

- $\mathbb{E}\big[K_{ixy} - \overline{K_{ixy}}\big] = 0$, which is immediate from $\overline{K_{ixy}} = \mathbb{E}[K_{ixy}]$.
- $\mathbb{E}\big[K_{i\cdot y} - \overline{K_{i\cdot y}}\big] = 0$, which is immediate from $\overline{K_{i\cdot y}} = \mathbb{E}[K_{i\cdot y}]$.
- If $i \neq j$ then $\mathbb{E}\big[(K_{ixy} - \overline{K_{ixy}})(K_{jxy} - \overline{K_{jxy}})\big] = 0$, because K_{ixy} and K_{jxy} are independent.
- If $i \neq j$ then $\mathbb{E}\big[(K_{ixy} - \overline{K_{ixy}})(K_{j\cdot y} - \overline{K_{j\cdot y}})\big] = 0$, because K_{ixy} and $K_{j\cdot y}$ are independent.
- If $i \neq j$ then $\mathbb{E}\big[(K_{i\cdot y} - \overline{K_{i\cdot y}})(K_{j\cdot y} - \overline{K_{j\cdot y}})\big] = 0$, because $K_{i\cdot y}$ and $K_{j\cdot y}$ are independent.
- For each $i \in \mathcal{I} \setminus \mathcal{I}^\star$, $(K_{ixy} \colon (x,y) \in \mathcal{D})$ follows the multinomial distribution with the sample size n_i and the probabilities $D_i[x,y]$ for $(x,y) \in \mathcal{D}$, therefore

  $$\mathbb{E}\big[(K_{ixy} - \overline{K_{ixy}})^2\big] = \mathbb{V}[K_{ixy}] = n_i D_i[x,y](1 - D_i[x,y]).$$

- For each $i \in \mathcal{I}^\star$, $(K_{i\cdot y} \colon y \in \mathcal{Y}^+)$ follows the multinomial distribution with the sample size n_i and the probabilities $\frac{D_i[x,y]}{\pi_i[x]}$ for $(x,y) \in \mathcal{D}$, therefore

  $$\mathbb{E}\big[(K_{i\cdot y} - \overline{K_{i\cdot y}})^2\big] = \mathbb{V}[K_{i\cdot y}] = n_i \frac{D_i[x,y]}{\pi_i[x]} \Big(1 - \frac{D_i[x,y]}{\pi_i[x]}\Big).$$

  Hence the expected value of $f_{xy}(\boldsymbol{K}_{xy})$ is given by:

$$\mathbb{E}[f_{xy}(\boldsymbol{K}_{xy})] = f_{xy}(\overline{\boldsymbol{K}_{xy}}) + \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} \tfrac{1}{2} \tfrac{\partial^2 f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{ixy}^2} \mathbb{E}\big[(K_{ixy} - \overline{K_{ixy}})^2\big] + \sum_{i\in\mathcal{I}^\star} \tfrac{\pi_i[x]^2}{2} \tfrac{\partial^2 f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{i\cdot y}^2} \mathbb{E}\big[(K_{i\cdot y} - \overline{K_{i\cdot y}})^2\big] + \mathcal{O}(\boldsymbol{K}_{xy}^3).$$

Therefore the expected value of $\hat{H}_{\mathcal{I}^\star}(X,Y)$ is given by:

$$\begin{aligned}\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X,Y)\Big] &= H(X,Y) - \sum_{(x,y)\in\mathcal{D}}\Bigg(\sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\tfrac{1}{2}\tfrac{\theta_i^2}{n_i^2 P_{XY}[x,y]}n_i D_i[x,y]\,(1-D_i[x,y]) \\ &\qquad + \sum_{i\in\mathcal{I}^\star}\tfrac{\pi_i[x]^2}{2}\tfrac{\theta_i^2}{n_i^2 P_{XY}[x,y]}n_i\tfrac{D_i[x,y]}{\pi_i[x]}\Big(1-\tfrac{D_i[x,y]}{\pi_i[x]}\Big)+\mathcal{O}(n_i^{-2})\Bigg) \\ &= H(X,Y) - \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\tfrac{\theta_i^2}{2n_i}\sum_{(x,y)\in\mathcal{D}}\tfrac{D_i[x,y](1-D_i[x,y])}{P_{XY}[x,y]} - \sum_{i\in\mathcal{I}^\star}\tfrac{\theta_i^2}{2n_i}\sum_{(x,y)\in\mathcal{D}}\tfrac{D_i[x,y](\pi_i[x]-D_i[x,y])}{P_{XY}[x,y]} + \sum_{i\in\mathcal{I}}\mathcal{O}(n_i^{-2}).\end{aligned}$$

□

Proposition A.2 (Mean of marginal output entropy estimated using the abstraction-then-sampling). The expected value $\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(Y)\Big]$ of the empirical output entropy is given by:

$$\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(Y)\Big] = H(Y) - \sum_{i\in\mathcal{I}}\frac{\theta_i^2}{2n_i}\Big(\sum_{y\in\mathcal{Y}^+}\varphi_{iy}\Big)+\mathcal{O}(n_i^{-2})\,.$$

Proof. Recall that $\mathcal{D}$ is the set of pairs of inputs and outputs with non-zero probabilities, $\mathcal{D}_x=\{y\colon (x,y)\in\mathcal{D}\}$ and $\mathcal{D}_y=\{x\colon (x,y)\in\mathcal{D}\}$. For each $i\in\mathcal{I}\setminus\mathcal{I}^\star$ and $y\in\mathcal{Y}$ let $L_{i\cdot y}=\sum_{x\in\mathcal{D}_y}K_{ixy}$. Recall the empirical joint distribution $\hat{P}_{XY}$ in Equation (1) in the proof of Proposition A.1.

Now the empirical marginal distribution $\hat{P}_Y$ on outputs is given by:

$$\begin{aligned}\hat{P}_Y[y] = \sum_{x\in\mathcal{D}_y}\hat{P}_{XY}[x,y] &= \sum_{x\in\mathcal{D}_y}\Big(q_{xy}+\sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\frac{\theta_i K_{ixy}}{n_i}+\sum_{i\in\mathcal{I}^\star}\frac{\theta_i\pi_i[x]K_{i\cdot y}}{n_i}\Big)\\ &= \sum_{x\in\mathcal{D}_y}q_{xy}+\sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\frac{\theta_i L_{i\cdot y}}{n_i}+\sum_{i\in\mathcal{I}^\star}\frac{\theta_i K_{i\cdot y}}{n_i}.\end{aligned}$$

Let $\boldsymbol{K}_y=(L_{1\cdot y},L_{2\cdot y},\dots,L_{m'\cdot y},K_{m'+1\cdot y},K_{m'+2\cdot y},\dots,K_{m\cdot y})$, and $f_y(\boldsymbol{K}_y)$ be the following m-ary function:

$$f_y(\boldsymbol{K}_y)=\Big(\sum_{x\in\mathcal{D}_y}q_{xy}+\sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\tfrac{\theta_i L_{i\cdot y}}{n_i}+\sum_{i\in\mathcal{I}^\star}\tfrac{\theta_i K_{i\cdot y}}{n_i}\Big)\log\Big(\sum_{x\in\mathcal{D}_y}q_{xy}+\sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\tfrac{\theta_i L_{i\cdot y}}{n_i}+\sum_{i\in\mathcal{I}^\star}\tfrac{\theta_i K_{i\cdot y}}{n_i}\Big),$$

which equals $\hat{P}_Y[y]\log\hat{P}_Y[y]$.

Let $\mathcal{Y}^+$ be the set of outputs with non-zero probabilities. Then the empirical marginal entropy is:

$$\hat{H}_{\mathcal{I}^\star}(Y)=-\sum_{y\in\mathcal{Y}^+}\hat{P}_Y[y]\log\hat{P}_Y[y]=-\sum_{y\in\mathcal{Y}^+}f_y(\boldsymbol{K}_y).$$

Let $\overline{L_{i\cdot y}}=\mathbb{E}[L_{i\cdot y}]$ for each $i\in\mathcal{I}\setminus\mathcal{I}^\star$, and $\overline{\boldsymbol{K}_y}=\mathbb{E}[\boldsymbol{K}_y]$. Then $\overline{L_{i\cdot y}}=\sum_{x\in\mathcal{D}_y}\overline{K_{ixy}}$. By the Taylor expansion of $f_y(\boldsymbol{K}_y)$ (w.r.t. the multiple dependent variables $\boldsymbol{K}_y$) at $\overline{\boldsymbol{K}_y}$, we have:

$$\begin{aligned}f_y(\boldsymbol{K}_y) &= f_y(\overline{\boldsymbol{K}_y})+\sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\tfrac{\partial f_y(\overline{\boldsymbol{K}_y})}{\partial L_{i\cdot y}}(L_{i\cdot y}-\overline{L_{i\cdot y}})+\sum_{i\in\mathcal{I}^\star}\tfrac{\partial f_y(\overline{\boldsymbol{K}_y})}{\partial K_{i\cdot y}}(K_{i\cdot y}-\overline{K_{i\cdot y}})\\ &\quad+\sum_{i,j\in\mathcal{I}\setminus\mathcal{I}^\star}\frac{1}{2}\tfrac{\partial^2 f_y(\overline{\boldsymbol{K}_y})}{\partial L_{i\cdot y}\partial L_{j\cdot y}}(L_{i\cdot y}-\overline{L_{i\cdot y}})(L_{j\cdot y}-\overline{L_{j\cdot y}})+\sum_{\substack{i\in\mathcal{I}\setminus\mathcal{I}^\star\\ j\in\mathcal{I}^\star}}\frac{1}{2}\tfrac{\partial^2 f_y(\overline{\boldsymbol{K}_y})}{\partial L_{i\cdot y}\partial K_{j\cdot y}}(L_{i\cdot y}-\overline{L_{i\cdot y}})(K_{j\cdot y}-\overline{K_{j\cdot y}})\\ &\quad+\sum_{i,j\in\mathcal{I}^\star}\frac{1}{2}\tfrac{\partial^2 f_y(\overline{\boldsymbol{K}_y})}{\partial K_{i\cdot y}\partial K_{j\cdot y}}(K_{i\cdot y}-\overline{K_{i\cdot y}})(K_{j\cdot y}-\overline{K_{j\cdot y}})+\mathcal{O}(\boldsymbol{K}_y^3)\end{aligned}$$

Recall that $D_{Yi}[y]=\sum_{x\in\mathcal{X}}D_i[x,y]$. To compute the expected value $\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(Y)\Big]$ of the estimated marginal entropy, it should be noted that:

- $\mathbb{E}\big[L_{i\cdot y} - \overline{L_{i\cdot y}}\big] = 0$, which is immediate from $\overline{L_{i\cdot y}} = \mathbb{E}[L_{i\cdot y}]$.
- $\mathbb{E}\big[K_{i\cdot y} - \overline{K_{i\cdot y}}\big] = 0$, which is immediate from $\overline{K_{i\cdot y}} = \mathbb{E}[K_{i\cdot y}]$.
- If $i \neq j$ then $\mathbb{E}\big[(L_{i\cdot y} - \overline{L_{i\cdot y}})(L_{j\cdot y} - \overline{L_{j\cdot y}})\big] = 0$, because $L_{i\cdot y}$ and $L_{j\cdot y}$ are independent.
- $\mathbb{E}\big[(L_{i\cdot y} - \overline{L_{i\cdot y}})(K_{j\cdot y} - \overline{K_{j\cdot y}})\big] = 0$, because $L_{i\cdot y}$ and $K_{j\cdot y}$ are independent.
- If $i \neq j$ then $\mathbb{E}\big[(K_{i\cdot y} - \overline{K_{i\cdot y}})(K_{j\cdot y} - \overline{K_{j\cdot y}})\big] = 0$, because $K_{i\cdot y}$ and $K_{j\cdot y}$ are independent.
- For $i \in \mathcal{I} \setminus \mathcal{I}^\star$, $(L_{i\cdot y} \colon y \in \mathcal{Y}^+)$ follows the multinomial distribution with the sample size n_i and the probabilities $D_{Yi}[y]$ for $y \in \mathcal{Y}^+$, therefore

  $$\mathbb{E}\big[(L_{i\cdot y} - \overline{L_{i\cdot y}})^2\big] = \mathbb{V}[L_{i\cdot y}] = n_i D_{Yi}[y]\,(1 - D_{Yi}[y])\,.$$

- For $i \in \mathcal{I}^\star$, $(K_{i\cdot y} \colon y \in \mathcal{Y}^+)$ follows the multinomial distribution with the sample size n_i and the probabilities $D_{Yi}[y]$ for $y \in \mathcal{Y}^+$, therefore

  $$\mathbb{E}\big[(K_{i\cdot y} - \overline{K_{i\cdot y}})^2\big] = \mathbb{V}[K_{i\cdot y}] = n_i D_{Yi}[y]\,(1 - D_{Yi}[y])\,.$$

  Hence the expected value of $f_y(\boldsymbol{K}_y)$ is given by:

$$\mathbb{E}[f_y(\boldsymbol{K}_y)] = f_y(\overline{\boldsymbol{K}_y}) + \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} \frac{1}{2}\frac{\partial^2 f_y(\overline{\boldsymbol{K}_y})}{\partial L_{i\cdot y}^2}\,\mathbb{E}\big[(L_{i\cdot y} - \overline{L_{i\cdot y}})^2\big] + \sum_{i\in\mathcal{I}^\star} \frac{1}{2}\frac{\partial^2 f_y(\overline{\boldsymbol{K}_y})}{\partial K_{i\cdot y}^2}\,\mathbb{E}\big[(K_{i\cdot y} - \overline{K_{i\cdot y}})^2\big] + \mathcal{O}(\boldsymbol{K}_y^3).$$

Therefore the expected value of $\hat{H}_{\mathcal{I}^\star}(Y)$ is given by:

$$\begin{aligned}\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(Y)\Big] &= H(Y) - \sum_{y\in\mathcal{Y}^+}\Big(\sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\tfrac{\theta_i^2}{2n_i^2 P_Y[y]}\mathbb{E}\big[(L_{i\cdot y} - \overline{L_{i\cdot y}})^2\big] + \sum_{i\in\mathcal{I}^\star}\tfrac{\theta_i^2}{2n_i^2 P_Y[y]}\mathbb{E}\big[(K_{i\cdot y} - \overline{K_{i\cdot y}})^2\big] + \mathcal{O}(n_i^{-2})\Big)\\ &= H(Y) - \sum_{i\in\mathcal{I}}\frac{\theta_i^2}{2n_i}\sum_{y\in\mathcal{Y}^+}\frac{D_{Yi}[y]\,(1 - D_{Yi}[y])}{P_Y[y]} + \mathcal{O}(n_i^{-2}).\end{aligned}$$

□

Proposition A.3 (Mean of marginal input entropy estimated using the abstraction-then-sampling). The expected value $\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X)\Big]$ of the empirical input entropy is given by:

$$\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X)\Big] = H(X) - \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\frac{\theta_i^2}{2n_i}\Big(\sum_{x\in\mathcal{X}^+}\varphi_{ix}\Big) + \mathcal{O}(n_i^{-2})\,.$$

Proof. For the components $i \in \mathcal{I}^\star$, the prior $\pi_i[x]$ is known to the analyst and used in the abstraction-then-sampling technique. Hence these components produce no bias in estimating $H(X)$. For the components $i \in \mathcal{I} \setminus \mathcal{I}^\star$, we derive the bias in a similar way to the proof of Proposition A.2. Hence the theorem follows. □

Theorem 5.6 (Mean of mutual information estimated using the abstraction-then-sampling). The expected value $\mathbb{E}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big]$ of the estimated mutual information is given by:

$$\mathbb{E}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big] = I(X;Y) + \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}}\varphi_{ixy} - \sum_{x\in\mathcal{X}^+}\varphi_{ix} - \sum_{y\in\mathcal{Y}^+}\varphi_{iy}\Big) + \sum_{i\in\mathcal{I}^\star}\frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}}\psi_{ixy} - \sum_{y\in\mathcal{Y}^+}\varphi_{iy}\Big) + \mathcal{O}(n_i^{-2})$$

where $\psi_{ixy} \stackrel{\text{def}}{=} \frac{D_i[x,y]\pi_i[x] - D_i[x,y]^2}{P_{XY}[x,y]}$.

Proof. By Propositions A.1, A.2, and A.3, we obtain the expected value of the estimated mutual information:

$$\begin{aligned}\mathbb{E}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big] &= \mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X)\Big] + \mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(Y)\Big] - \mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X,Y)\Big]\\ &= I(X;Y) + \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}}\tfrac{D_i[x,y] - D_i[x,y]^2}{P_{XY}[x,y]} - \sum_{x\in\mathcal{X}^+}\tfrac{D_{Xi}[x] - D_{Xi}[x]^2}{P_X[x]} - \sum_{y\in\mathcal{Y}^+}\tfrac{D_{Yi}[y] - D_{Yi}[y]^2}{P_Y[y]}\Big)\end{aligned}$$

$$+\sum_{i\in\mathcal{I}^\star}\frac{\theta_i^2}{2n_i}\Big(\sum_{x\in\mathcal{D}_x}\frac{D_i[x,y]\pi_i[x]-D_i[x,y]^2}{P_{XY}[x,y]}-\sum_{y\in\mathcal{Y}^+}\frac{D_{Yi}[y]-D_{Yi}[y]^2}{P_Y[y]}\Big)+\mathcal{O}(n_i^{-2}).$$

Therefore we obtain the theorem. □

A.2. Proofs for the Mean Estimation Using Only the Standard Sampling

In this section we present the proofs for Theorem 4.1 in Section 4.1 and Proposition 4.3 in Section 4.3.

Theorem 4.1 (Mean of estimated mutual information). The expected value $\mathbb{E}\Big[\hat{I}(X;Y)\Big]$ of the estimated mutual information is given by:

$$\mathbb{E}\Big[\hat{I}(X;Y)\Big]=I(X;Y)+\sum_{i\in\mathcal{I}}\frac{\theta_i^2}{2n_i}\Big(\sum_{(x,y)\in\mathcal{D}}\varphi_{ixy}-\sum_{x\in\mathcal{X}^+}\varphi_{ix}-\sum_{y\in\mathcal{Y}^+}\varphi_{iy}\Big)+\mathcal{O}(n_i^{-2})$$

where $\varphi_{ixy}=\frac{D_i[x,y]-D_i[x,y]^2}{P_{XY}[x,y]}$, $\varphi_{ix}=\frac{D_{Xi}[x]-D_{Xi}[x]^2}{P_X[x]}$ and $\varphi_{iy}=\frac{D_{Yi}[y]-D_{Yi}[y]^2}{P_Y[y]}$.

Proof. If $\mathcal{I}^\star=\emptyset$, then $\mathbb{E}\Big[\hat{I}(X;Y)\Big]=\mathbb{E}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big]$. Hence the claim follows from Theorem 5.6. □

Proposition 4.3 (Mean of estimated Shannon entropy). The expected value $\mathbb{E}\Big[\hat{H}(X)\Big]$ of the estimated Shannon entropy is given by:

$$\mathbb{E}\Big[\hat{H}(X)\Big]=H(X)-\sum_{i\in\mathcal{I}}\frac{\theta_i^2}{2n_i}\sum_{x\in\mathcal{X}^+}\frac{D_{Xi}[x]\,(1-D_{Xi}[x])}{P_X[x]}+\mathcal{O}(n_i^{-2})\,.$$

Proof. If $\mathcal{I}^\star=\emptyset$, then $\mathbb{E}\Big[\hat{H}(X)\Big]=\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X)\Big]$. Hence the claim follows from Proposition A.3. □

A.3. Proof for the Variance Estimation Using the Abstraction-Then-Sampling

In this section we present the proof for Theorem 5.7 in Section 5.2, i.e., the result on variance estimation using the abstraction-then-sampling.

To show the proofs we first calculate the covariances between random variables in Lemmas A.4, A.5, A.6, A.7, and A.8 as follows. Recall that the covariance $Cov[A,B]$ between two random variables A and B is defined by:

$$Cov[A,B]\stackrel{\text{def}}{=}\mathbb{E}[(A-\mathbb{E}[A])(B-\mathbb{E}[B])]\,.$$

Lemma A.4 (Covariance between K_{ixy} and $K_{i'x'y'}$). For any $i,i'\in\mathcal{I}\setminus\mathcal{I}^\star$, $Cov[K_{ixy},K_{i'x'y'}]$ is given by:

$$Cov[K_{ixy},K_{i'x'y'}]=\begin{cases}0 & \text{if } i\neq i'\\ n_iD_i[x,y](1-D_i[x,y]) & \text{if } i=i',\,x=x' \text{ and } y=y'\\ -n_iD_i[x,y]D_i[x',y'] & \text{otherwise.}\end{cases}$$

Proof. Let $i,i'\in\mathcal{I}\setminus\mathcal{I}^\star$. If $i\neq i'$ then K_{ixy} and $K_{i'x'y'}$ are independent, hence their covariance is 0. Otherwise, the theorem follows from that fact that for each $i\in\mathcal{I}\setminus\mathcal{I}^\star$, $(K_{ixy}\colon(x,y)\in\mathcal{D})$ follows the multinomial distribution with the sample size n_i and the probabilities $D_i[x,y]$ for $(x,y)\in\mathcal{D}$. □

Lemma A.5 (Covariance between K_{ixy} and $L_{i'\cdot y'}$). For any $i,i'\in\mathcal{I}\setminus\mathcal{I}^\star$, $Cov[K_{ixy},L_{i'\cdot y'}]$ is given by:

$$Cov[K_{ixy},L_{i'\cdot y'}]=\begin{cases}0 & \text{if } i\neq i'\\ n_iD_i[x,y](1-D_{Yi}[y]) & \text{if } i=i',\,x=x' \text{ and } y=y'\\ -n_iD_i[x,y]D_{Yi}[y'] & \text{otherwise.}\end{cases}$$

Proof. Let $i,i'\in\mathcal{I}\setminus\mathcal{I}^\star$. If $i\neq i'$ then K_{ixy} and $L_{i'\cdot y'}$ are independent, hence their covariance is 0. Otherwise, the

covariance $Cov[K_{ixy}, L_{i\cdot y'}]$ is calculated as:

$$Cov[K_{ixy}, L_{i\cdot y'}] = Cov\Big[K_{ixy}, \sum_{x' \in \mathcal{D}_y} K_{ix'y}\Big] = \sum_{x' \in \mathcal{D}_y} Cov[K_{ixy}, K_{ix'y}]\,.$$

Hence, when $y = y'$:

$$\begin{aligned} Cov[K_{ixy}, L_{i\cdot y'}] &= n_i D_i[x,y](1 - D_i[x,y]) - \sum_{x' \in \mathcal{D}_y \setminus \{x\}} n_i D_i[x,y] D_i[x',y] \\ &= n_i D_i[x,y] - \sum_{x' \in \mathcal{D}_y} n_i D_i[x,y] D_i[x',y] \\ &= n_i D_i[x,y](1 - D_{Yi}[y]). \end{aligned}$$

When $y \neq y'$:

$$Cov[K_{ixy}, L_{i\cdot y'}] = -\sum_{x' \in \mathcal{D}_y} n_i D_i[x,y] D_i[x',y] = -n_i D_i[x,y] D_{Yi}[y].$$

□

Lemma A.6 (Covariance between $L_{i\cdot y}$ and $L_{i'\cdot y'}$). For any $i \in \mathcal{I} \setminus \mathcal{I}^\star$, $Cov[L_{i\cdot y}, L_{i\cdot y'}]$ is given by:

$$Cov[L_{i\cdot y}, L_{i'\cdot y'}] = \begin{cases} 0 & \text{if } i \neq i' \\ n_i D_{Yi}[y](1 - D_{Yi}[y]) & \text{if } i = i',\ x = x' \text{ and } y = y' \\ -n_i D_{Yi}[y] D_{Yi}[y'] & \text{otherwise.} \end{cases}$$

Proof. Let $i, i' \in \mathcal{I} \setminus \mathcal{I}^\star$. If $i \neq i'$ then $L_{i\cdot y}$ and $L_{i'\cdot y'}$ are independent, hence their covariance is 0. Otherwise, the covariance is calculated as:

$$Cov[L_{i\cdot y}, L_{i\cdot y}] = Cov\Big[\sum_{x \in \mathcal{D}_y} K_{ixy}, \sum_{x' \in \mathcal{D}_y} K_{ix'y}\Big] = \sum_{x \in \mathcal{D}_y} \sum_{x' \in \mathcal{D}_y} Cov[K_{ixy}, K_{ix'y}]\,.$$

Hence, when $y = y'$:

$$Cov[L_{i\cdot y}, L_{i\cdot y}] = \sum_{x \in \mathcal{D}_y} \Big(n_i D_i[x,y](1 - D_i[x,y]) - \sum_{x' \in \mathcal{D}_y \setminus \{x\}} n_i D_i[x,y] D_i[x',y] \Big) = n_i D_{Yi}[y](1 - D_{Yi}[y]).$$

When $y \neq y'$:

$$Cov[L_{i\cdot y}, L_{i\cdot y'}] = -\sum_{x \in \mathcal{D}_y} \sum_{x' \in \mathcal{D}_{y'}} n_i D_i[x,y] D_i[x',y'] = -n_i D_{Yi}[y] D_{Yi}[y'].$$

□

Lemma A.7 (Covariance between $L_{i\cdot y}$ and $K_{i'\cdot y'}$). For any $i \in \mathcal{I} \setminus \mathcal{I}^\star$ and any $i' \in \mathcal{I}^\star$, $Cov[L_{i\cdot y}, K_{i'\cdot y'}] = 0$.

Proof. The claim is immediate from the fact that $L_{i\cdot y}$ and $K_{i'\cdot y'}$ are independent. □

Lemma A.8 (Covariance between $K_{i\cdot y}$ and $K_{i'\cdot y'}$). For any $i, i' \in \mathcal{I}^\star$, $Cov[K_{i\cdot y}, K_{i'\cdot y'}]$ is given by:

$$Cov[K_{i\cdot y}, K_{i'\cdot y'}] = \begin{cases} 0 & \text{if } i \neq i' \\ n_i D_{Yi}[y](1 - D_{Yi}[y]) & \text{if } i = i' \text{ and } y = y' \\ -n_i D_{Yi}[y] D_{Yi}[y'] & \text{otherwise.} \end{cases}$$

Proof. Let $i, i' \in \mathcal{I}^\star$. If $i \neq i'$ then $K_{i\cdot y}$ and $K_{i'\cdot y'}$ are independent, hence their covariance is 0. Otherwise, the claim follows from that fact that for each $i \in \mathcal{I}^\star$, $(K_{i\cdot y} \colon y \in \mathcal{Y}_i^+)$ follows the multinomial distribution with the sample size n_i and the probabilities $D_{Yi}[y]$ for $y \in \mathcal{Y}_i^+$. □

Theorem 5.7 (Variance of mutual information estimated using the abstraction-then-sampling). The variance $\mathbb{V}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big]$ of the estimated mutual information is given by:

$$\mathbb{V}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big] = \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\frac{\theta_i^2}{n_i}\Bigg(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1+\log\tfrac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)^2 - \Big(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1+\log\tfrac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)\Big)^2\Bigg)$$
$$+\sum_{i\in\mathcal{I}^\star}\frac{\theta_i^2}{n_i}\Bigg(\sum_{y\in\mathcal{Y}^+} D_{Yi}[y]\gamma_{ixy}^2 - \Big(\sum_{y\in\mathcal{Y}^+} D_{Yi}[y]\gamma_{ixy}\Big)^2\Bigg) + \mathcal{O}(n_i^{-2})$$

where $\gamma_{ixy} \stackrel{\text{def}}{=} \log P_Y[y] - \sum_{x\in\mathcal{X}} \pi_i[x]\log P_{XY}[x,y]$.

Proof. We first define B_{ixy} and $B_{i\cdot y}$ by the following:

- $B_{ixy} \stackrel{\text{def}}{=} \frac{\partial f_{xy}(\overline{\boldsymbol{K}_{xy}})}{\partial K_{ixy}} = \frac{\theta_i}{n_i}(1+\log P_{XY}[x,y])$.
- For each $i\in\mathcal{I}\setminus\mathcal{I}^\star$, $B_{i\cdot y} \stackrel{\text{def}}{=} \frac{\partial f_y(\overline{\boldsymbol{K}_y})}{\partial L_{i\cdot y}} = \frac{\theta_i}{n_i}(1+\log P_Y[y])$.
- For each $i\in\mathcal{I}^\star$, $B_{i\cdot y} \stackrel{\text{def}}{=} \frac{\partial f_y(\overline{\boldsymbol{K}_y})}{\partial K_{i\cdot y}} = \frac{\theta_i}{n_i}(1+\log P_Y[y])$.

Then the variance of $\hat{H}_{\mathcal{I}^\star}(X,Y)$ is obtained from Lemmas A.4 and A.8 as follows:

$$\begin{aligned}
\mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(X,Y)\Big] &= \mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X,Y)^2\Big] - \Big(\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(X,Y)\Big]\Big)^2\\
&= \sum_{i,i'\in\mathcal{I}\setminus\mathcal{I}^\star}\sum_{(x,y)\in\mathcal{D}}\sum_{(x',y')\in\mathcal{D}} B_{ixy}B_{i'x'y'}\,Cov[K_{ixy},K_{i'x'y'}]\\
&\quad+\sum_{i,i'\in\mathcal{I}^\star}\sum_{(x,y)\in\mathcal{D}}\sum_{(x',y')\in\mathcal{D}} \pi_i[x]B_{ixy}\pi_{i'}[x']B_{i'x'y'}\,Cov[K_{i\cdot y},K_{i'\cdot y'}] + \mathcal{O}(n_i^{-2})\\
&= \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\sum_{(x,y)\in\mathcal{D}} n_i\Big(B_{ixy}^2 D_i[x,y](1-D_i[x,y]) - \sum_{(x',y')\in\mathcal{D}\setminus\{(x,y)\}} B_{ixy}B_{ix'y'}D_i[x,y]D_i[x',y']\Big)\\
&\quad+\sum_{i\in\mathcal{I}^\star}\sum_{(x,y)\in\mathcal{D}} n_i\Big(\pi_i[x]^2B_{ixy}^2 D_{Yi}[y](1-D_{Yi}[y]) - \sum_{(x',y')\in\mathcal{D}\setminus\{(x,y)\}} \pi_i[x]\pi_i[x']B_{ixy}B_{ix'y'}D_{Yi}[y]D_{Yi}[y']\Big)\\
&\quad+\mathcal{O}(n_i^{-2})\\
&= \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} n_i\sum_{(x,y)\in\mathcal{D}} D_i[x,y]B_{ixy}\Big(B_{ixy} - \sum_{(x',y')\in\mathcal{D}} B_{ix'y'}D_i[x',y']\Big)\\
&\quad+\sum_{i\in\mathcal{I}^\star} n_i\Bigg(\sum_{y\in\mathcal{Y}^+} D_{Yi}[y]\Big(\sum_{x\in\mathcal{D}_y}\pi_i[x]^2B_{ixy}^2\Big) - \Big(\sum_{(x,y)\in\mathcal{D}} D_{Yi}[y]\pi_i[x]B_{ixy}\Big)^2\Bigg) + \mathcal{O}(n_i^{-2}).
\end{aligned}$$

The variances of $\hat{H}_{\mathcal{I}^\star}(Y)$ is obtained from Lemmas A.6 and A.8 as follows:

$$\begin{aligned}
\mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(Y)\Big] &= \mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(Y)^2\Big] - \Big(\mathbb{E}\Big[\hat{H}_{\mathcal{I}^\star}(Y)\Big]\Big)^2\\
&= \sum_{y,y'\in\mathcal{Y}^+}\Big(\sum_{i,i'\in\mathcal{I}\setminus\mathcal{I}^\star} B_{i\cdot y}B_{i'\cdot y'}\,Cov[L_{i\cdot y},L_{i'\cdot y'}] + \sum_{i,i'\in\mathcal{I}^\star} B_{i\cdot y}B_{i'\cdot y'}\,Cov[K_{i\cdot y},K_{i'\cdot y'}]\Big) + \mathcal{O}(n_i^{-2})\\
&= \sum_{i\in\mathcal{I}} n_i\sum_{y\in\mathcal{Y}^+} D_{Yi}[y]B_{i\cdot y}\Big(B_{i\cdot y} - \sum_{y'\in\mathcal{Y}^+} B_{i\cdot y'}D_{Yi}[y']\Big) + \mathcal{O}(n_i^{-2})\\
&= \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} n_i\sum_{(x,y)\in\mathcal{D}} D_i[x,y]B_{i\cdot y}\Big(B_{i\cdot y} - \sum_{(x',y')\in\mathcal{D}} B_{iy'}D_i[x',y']\Big)
\end{aligned}$$

$$+\sum_{i\in\mathcal{I}^\star} n_i \sum_{y\in\mathcal{Y}^+} D_{Yi}[y]B_{i\cdot y}\Big(B_{i\cdot y} - \sum_{y'\in\mathcal{Y}^+} B_{i\cdot y'}D_{Yi}[y']\Big) + \mathcal{O}(n_i^{-2}).$$

Similarly, for $\mathcal{B}_{ix\cdot} = \frac{\theta_i}{n_i}\Big(1+\log P_X[x]\Big)$, the variance of $\hat{H}_{\mathcal{I}^\star}(X)$ is given by:

$$\mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(X)\Big] = \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} n_i \sum_{(x,y)\in\mathcal{D}} D_i[x,y]B_{ix\cdot}\Big(B_{ix\cdot} - \sum_{(x',y')\in\mathcal{D}} B_{ix'\cdot}D_i[x',y']\Big) + \mathcal{O}(n_i^{-2}),$$

which is symmetric to $\mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(Y)\Big]$ only w.r.t. $\mathcal{I}\setminus\mathcal{I}^\star$ [5].

The covariance between $\hat{H}_{\mathcal{I}^\star}(X,Y)$ and $\hat{H}_{\mathcal{I}^\star}(Y)$ is obtained from Lemmas A.5 and A.8 as follows:

$$\begin{aligned}
Cov\Big[\hat{H}_{\mathcal{I}^\star}(Y), \hat{H}_{\mathcal{I}^\star}(X,Y)\Big] &= \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\sum_{(x,y)\in\mathcal{D}}\sum_{y'\in\mathcal{Y}^+} B_{ixy}B_{i\cdot y'}\,Cov[K_{ixy}, L_{i\cdot y'}] \\
&\quad + \sum_{i\in\mathcal{I}^\star}\sum_{(x,y)\in\mathcal{D}}\sum_{y'\in\mathcal{Y}^+} \pi_i[x]B_{ixy}B_{i\cdot y'}\,Cov[K_{i\cdot y}, K_{i\cdot y'}] + \mathcal{O}(n_i^{-2}) \\
&= \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} n_i \sum_{(x,y)\in\mathcal{D}} D_i[x,y]B_{ixy}\Big(B_{i\cdot y} - \sum_{(x',y')\in\mathcal{D}} B_{i\cdot y'}D_i[x',y']\Big) \\
&\quad + \sum_{i\in\mathcal{I}^\star} n_i \sum_{(x,y)\in\mathcal{D}} D_{Yi}[y]\pi_i[x]B_{ixy}\Big(B_{i\cdot y} - \sum_{y'\in\mathcal{Y}^+} B_{i\cdot y'}D_{Yi}[y']\Big) + \mathcal{O}(n_i^{-2})
\end{aligned}$$

Similarly, the covariance between $\hat{H}_{\mathcal{I}^\star}(X,Y)$ and $\hat{H}_{\mathcal{I}^\star}(X)$ is given by:

$$Cov\Big[\hat{H}_{\mathcal{I}^\star}(X), \hat{H}_{\mathcal{I}^\star}(X,Y)\Big] = \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} n_i \sum_{(x,y)\in\mathcal{D}} D_i[x,y]B_{ixy}\Big(B_{ix\cdot} - \sum_{(x',y')\in\mathcal{D}} B_{ix'\cdot}D_i[x',y']\Big) + \mathcal{O}(n_i^{-2})$$

The covariance between $\hat{H}_{\mathcal{I}^\star}(X)$ and $\hat{H}_{\mathcal{I}^\star}(Y)$ is given by:

$$Cov\Big[\hat{H}_{\mathcal{I}^\star}(X), \hat{H}_{\mathcal{I}^\star}(Y)\Big] = \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} n_i \sum_{(x,y)\in\mathcal{D}} D_i[x,y]B_{ix\cdot}\Big(B_{i\cdot y} - \sum_{(x',y')\in\mathcal{D}} B_{i\cdot y'}D_i[x',y']\Big) + \mathcal{O}(n_i^{-2})$$

Therefore the variance of the mutual information is as follows:

$$\begin{aligned}
\mathbb{V}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big] &= \mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(X) + \hat{H}_{\mathcal{I}^\star}(Y) - \hat{H}_{\mathcal{I}^\star}(X,Y)\Big] \\
&= \mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(X)\Big] + \mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(Y)\Big] + \mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(X,Y)\Big] + 2Cov\Big[\hat{H}_{\mathcal{I}^\star}(X), \hat{H}_{\mathcal{I}^\star}(Y)\Big] \\
&\quad - 2Cov\Big[\hat{H}_{\mathcal{I}^\star}(X), \hat{H}_{\mathcal{I}^\star}(X,Y)\Big] - 2Cov\Big[\hat{H}_{\mathcal{I}^\star}(Y), \hat{H}_{\mathcal{I}^\star}(X,Y)\Big] \\
&= \sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star} n_i \sum_{(x,y)\in\mathcal{D}} D_i[x,y] \\
&\qquad \Big(B_{ix\cdot}\Big(B_{ix\cdot} - \sum_{(x',y')\in\mathcal{D}} B_{ix'\cdot}D_i[x',y']\Big) + B_{i\cdot y}\Big(B_{i\cdot y} - \sum_{(x',y')\in\mathcal{D}} B_{i\cdot y'}D_i[x',y']\Big) \\
&\qquad + B_{ixy}\Big(B_{ixy} - \sum_{(x',y')\in\mathcal{D}} B_{ix'y'}D_i[x',y']\Big) + 2B_{ix\cdot}\Big(B_{i\cdot y} - \sum_{(x',y')\in\mathcal{D}} B_{i\cdot y'}D_i[x',y']\Big) \\
&\qquad - 2B_{ixy}\Big(B_{ix\cdot} - \sum_{(x',y')\in\mathcal{D}} B_{ix'\cdot}D_i[x',y']\Big) - 2B_{ixy}\Big(B_{i\cdot y} - \sum_{(x',y')\in\mathcal{D}} B_{i\cdot y'}D_i[x',y']\Big)\Big)
\end{aligned}$$

[5] Note that the abstraction-then-sampling relies on partial knowledge on the prior, i.e., the analyst knows $\pi_i[x]$ for all $i\in\mathcal{I}^\star$, hence $\mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(X)\Big]$ has no term for $\mathcal{I}^\star$. On the other hand, the standard sampling here does not use knowledge on the prior.

$$
\begin{aligned}
&+\sum_{i\in\mathcal{I}^\star} n_i \sum_{y\in\mathcal{Y}^+} D_{Yi}[y]\Big(B_{i\cdot y}\Big(B_{i\cdot y}-\sum_{y'\in\mathcal{Y}^+} B_{i\cdot y'}D_{Yi}[y']\Big)\\
&\qquad\qquad +\sum_{x\in\mathcal{D}_y}\pi_i[x]B_{ixy}\sum_{x'\in\mathcal{X}^+}\pi_i[x']\Big(B_{ix'y}-\sum_{y'\in\mathcal{D}_{x'}} B_{ix'y'}D_{Yi}[y']\Big)\\
&\qquad\qquad -2\sum_{x\in\mathcal{D}_y}\pi_i[x]B_{ixy}\Big(B_{i\cdot y}-\sum_{y'\in\mathcal{Y}^+} B_{i\cdot y'}D_{Yi}[y']\Big)\Big)+\mathcal{O}(n_i^{-2})\\
&=\sum_{i\in\mathcal{I}\setminus\mathcal{I}^\star}\frac{\theta_i^2}{n_i}\Bigg(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1+\log\tfrac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)^2-\Big(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1+\log\tfrac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)\Big)^2\Bigg)\\
&\quad+\sum_{i\in\mathcal{I}^\star}\frac{\theta_i^2}{n_i}\Bigg(\sum_{y\in\mathcal{Y}^+} D_{Yi}[y]\Big(\log P_Y[y]-\sum_{x\in\mathcal{X}}\pi_i[x]\log P_{XY}[x,y]\Big)^2\\
&\qquad\qquad-\Big(\sum_{y\in\mathcal{Y}^+} D_{Yi}[y]\Big(\log P_Y[y]-\sum_{x\in\mathcal{X}}\pi_i[x]\log P_{XY}[x,y]\Big)\Big)^2\Bigg)+\mathcal{O}(n_i^{-2})
\end{aligned}
$$

□

A.4. Proof for the Variance Estimation Using Only the Standard Sampling

In this section we present the proofs for Theorem 4.2 in Section 4.2 and Proposition 4.4 in Section 4.3.

Theorem 4.2 (Variance of estimated mutual information). The variance $\mathbb{V}\Big[\hat{I}(X;Y)\Big]$ of the estimated mutual information is given by:

$$
\mathbb{V}\Big[\hat{I}(X;Y)\Big]=\sum_{i\in\mathcal{I}}\frac{\theta_i^2}{n_i}\Bigg(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1+\log\tfrac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)^2-\Big(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]\Big(1+\log\tfrac{P_X[x]P_Y[y]}{P_{XY}[x,y]}\Big)\Big)^2\Bigg)+O(n_i^{-2}).
$$

Proof. If $\mathcal{I}^\star=\emptyset$, then $\mathbb{E}\Big[\hat{I}(X;Y)\Big]=\mathbb{E}\Big[\hat{I}_{\mathcal{I}^\star}(X;Y)\Big]$. Hence the claim follows from Theorem 5.7. □

Proposition 4.4 (Variance of estimated Shannon entropy). The variance $\mathbb{V}\Big[\hat{H}(X)\Big]$ of the estimated Shannon entropy is given by:

$$
\mathbb{V}\Big[\hat{H}(X)\Big]=\sum_{i\in\mathcal{I}}\frac{\theta_i^2}{n_i}\Bigg(\sum_{x\in\mathcal{X}^+} D_{Xi}[x]\Big(1+\log P_X[x]\Big)^2-\Big(\sum_{x\in\mathcal{X}^+} D_{Xi}[x]\Big(1+\log P_X[x]\Big)\Big)^2\Bigg)+\mathcal{O}(n_i^{-2})\,.
$$

Proof. Let $\mathcal{I}^\star=\emptyset$ and $\mathcal{B}_{ix\cdot}=\frac{\theta_i}{n_i}\Big(1+\log P_X[x]\Big)$. Then by the proof of Theorem 5.7 in Appendix A.3, we have:

$$
\begin{aligned}
\mathbb{V}\Big[\hat{H}_{\mathcal{I}^\star}(X)\Big]&=\sum_{i\in\mathcal{I}} n_i\sum_{(x,y)\in\mathcal{D}} D_i[x,y]B_{ix\cdot}\Big(B_{ix\cdot}-\sum_{(x',y')\in\mathcal{D}} B_{ix'\cdot}D_i[x',y']\Big)+\mathcal{O}(n_i^{-2})\\
&=\sum_{i\in\mathcal{I}} n_i\Big(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]B_{ix\cdot}^2-\Big(\sum_{(x,y)\in\mathcal{D}} D_i[x,y]B_{ix\cdot}\Big)^2\Big)+\mathcal{O}(n_i^{-2})\\
&=\sum_{i\in\mathcal{I}}\frac{\theta_i^2}{n_i}\Bigg(\sum_{x\in\mathcal{X}^+} D_{Xi}[x]\Big(1+\log P_X[x]\Big)^2-\Big(\sum_{x\in\mathcal{X}^+} D_{Xi}[x]\Big(1+\log P_X[x]\Big)\Big)^2\Bigg)+\mathcal{O}(n_i^{-2}).
\end{aligned}
$$

□

A.5. Proofs for Adaptive Analysis

In this section we present the proofs for the results in Section 6. To prove these it suffices to show the following proposition:

Proposition A.9. Let $v_1, v_2, \ldots, v_m$ be m positive real numbers. Let $n, n_1, n_2, \ldots, n_m$ be $(m+1)$ positive real numbers such that $\sum_{i=1}^m n_i = n$. Then

$$\sum_{i=1}^{m} \frac{v_i}{n_i} \geq \frac{1}{n}\left(\sum_{i=1}^{m} \sqrt{v_i}\right)^2.$$

The equality holds when $n_i = \frac{\sqrt{v_i}n}{\sum_{j=1}^m \sqrt{v_j}}$ for all $i = 1, 2, \ldots, m$.

Proof. The proof is by induction on m. When $m = 1$ the equality holds trivially. When $m = 2$ it is sufficient to prove

$$\frac{v_1}{n_1} + \frac{v_2}{n_2} \geq \frac{\left(\sqrt{v_1} + \sqrt{v_2}\right)^2}{n_1 + n_2}. \tag{2}$$

By $n_1, n_2 > 0$, this is equivalent to $(n_1+n_2)(n_2v_1 + n_1v_2) \geq n_1n_2\left(\sqrt{v_1} + \sqrt{v_2}\right)^2$. We obtain this by:

$$\begin{aligned}(n_1+n_2)(n_2v_1 + n_1v_2) - n_1n_2\left(\sqrt{v_1} + \sqrt{v_2}\right)^2 &= (n_1+n_2)n_2v_1 + (n_1+n_2)n_1v_2 - n_1n_2\left(v_1 + 2\sqrt{v_1v_2} + v_2\right)\\ &= n_2^2v_1 + n_1^2v_2 - 2n_1n_2\sqrt{v_1v_2}\\ &= \left(n_2\sqrt{v_1} - n_1\sqrt{v_2}\right)^2\\ &\geq 0.\end{aligned}$$

Next we prove the inductive step as follows.

$$\begin{aligned}\sum_{i=1}^{m} \frac{v_i}{n_i} &= \left(\sum_{i=1}^{m-1} \frac{v_i}{n_i}\right) + \frac{v_m}{n_m}\\ &\geq \frac{1}{n_1 + \ldots + n_{m-1}}\left(\textstyle\sum_{i=1}^{m-1} \sqrt{v_i}\right)^2 + \frac{\sqrt{v_m}^2}{n_m} && \text{(by induction hypothesis)}\\ &\geq \frac{1}{(n_1 + \ldots + n_{m-1}) + n_m}\left(\sqrt{\left(\textstyle\sum_{i=1}^{m-1} \sqrt{v_i}\right)^2} + \sqrt{v_m}\right)^2 && \text{(by Equation (2))}\\ &= \frac{1}{n_1 + \ldots + n_m}\left(\textstyle\sum_{i=1}^{m-1} \sqrt{v_i} + \sqrt{v_m}\right)^2\\ &= \frac{1}{n}\left(\textstyle\sum_{i=1}^{m} \sqrt{v_i}\right)^2.\end{aligned}$$

Finally, when $n_i = \frac{\sqrt{v_i}n}{\sum_{j=1}^m \sqrt{v_j}}$ for all $i = 1, 2, \ldots, m$, then $\sum_{i=1}^m \frac{v_i}{n_i} = \sum_{i=1}^m \frac{v_i(\sum_{j=1}^m \sqrt{v_j})}{\sqrt{v_i}n} = \frac{1}{n}\left(\sum_{i=1}^m \sqrt{v_i}\right)^2$. □

Theorem 6.4 (Optimal sample sizes using the abstraction-then-sampling). Let $v_i^\star$ be the following intermediate variance of the component S_i:

$$v_i^\star = \begin{cases} \theta_i^2\left(\displaystyle\sum_{(x,y)\in\mathcal{D}} \hat{D}_i[x,y]\left(1 + \log\frac{\hat{P}_X[x]\hat{P}_Y[y]}{\hat{P}_{XY}[x,y]}\right)^2 - \left(\sum_{(x,y)\in\mathcal{D}} \hat{D}_i[x,y]\left(1 + \log\frac{\hat{P}_X[x]\hat{P}_Y[y]}{\hat{P}_{XY}[x,y]}\right)\right)^2\right) & \text{if } i \in \mathcal{I}\setminus\mathcal{I}^\star\\ \theta_i^2\left(\displaystyle\sum_{y\in\mathcal{Y}^+} \hat{D_{Yi}}[y]\hat{\gamma}_{ixy}^2 - \left(\sum_{y\in\mathcal{Y}^+} \hat{D_{Yi}}[y]\hat{\gamma}_{ixy}\right)^2\right) & \text{if } i \in \mathcal{I}^\star\end{cases}$$

Given the total sample size n, the variance of the estimated mutual information is minimized if, for all $i \in \mathcal{I}$ and $x \in \mathcal{X}$, the sample size n_i is given by: $n_i = \frac{\sqrt{v_i^\star}n}{\sum_{j=1}^m \sqrt{v_j^\star}}$.

Proof. By Proposition A.9 the variance $v = \sum_{i\in\mathcal{I}} \frac{v_i^\star}{n_i}$ of estimated mutual information is minimised when $n_i = \frac{\sqrt{v_i^\star}n}{\sum_{j=1}^m \sqrt{v_j^\star}}$. Hence the theorem follows. □

Theorem 6.1 (Optimal sample sizes). Given the total sample size n and the above intermediate variance v_i of the component S_i for each $i \in \mathcal{I}$, the variance of the mutual information estimate is minimized if, for all $i \in \mathcal{I}$, the sample size n_i for S_i is given by: $n_i = \frac{\sqrt{v_i}n}{\sum_{j=1}^m \sqrt{v_j}}$.

Proof. Let $\mathcal{I}^\star = \emptyset$. Then this theorem immediately follows from Theorem 6.4. □

Proposition 6.3 (Optimal sample sizes when knowing the prior). For each $i \in \mathcal{I}$ and $x \in \mathcal{X}$, let v_{ix} be the following intermediate variance of the component S_{ix}.

$$v_{ix} = \theta_{ix}^2 \left(\sum_{y\in\mathcal{D}_x} \hat{D}_i[y|x] \left(\log \frac{\widehat{P}_Y[y]}{\widehat{P}_{XY}[x,y]} \right)^2 - \left(\sum_{y\in\mathcal{D}_x} \hat{D}_i[y|x] \left(\log \frac{\widehat{P}_Y[y]}{\widehat{P}_{XY}[x,y]} \right) \right)^2 \right).$$

Given the total sample size n, the variance of the estimated mutual information is minimized if, for all $i \in \mathcal{I}$ and $x \in \mathcal{X}$, the sample size n_i and the importance prior λ_i satisfy: $n_i\lambda_i[x] = \frac{\sqrt{v_{ix}}n}{\sum_{j=1}^m \sqrt{v_{jx}}}$.

Proof. By Proposition A.9 the variance $v = \sum_{i\in\mathcal{I}} \sum_{x\in\mathcal{X}^+} \frac{v_{ix}}{n_i}$ of estimated mutual information is minimised when $n_i\lambda_i[x] = \frac{\sqrt{v_{ix}}n}{\sum_{j=1}^m \sqrt{v_{jx}}}$. Hence the theorem follows. □

Proposition 6.2 (Optimal sample sizes for Shannon entropy estimation). Given the total sample size n and the above intermediate variance v_i' of the component S_i for each $i \in \mathcal{I}$, the variance of the Shannon entropy estimate is minimized if, for all $i \in \mathcal{I}$, the sample size n_i for S_i satisfies $n_i = \frac{\sqrt{v_i'}n}{\sum_{j=1}^m \sqrt{v_j'}}$.

Proof. By Proposition A.9 the variance $v = \sum_{i\in\mathcal{I}} \frac{v_i'}{n_i}$ of estimated Shannon entropy is minimised when $n_i = \frac{\sqrt{v_i'}n}{\sum_{j=1}^m \sqrt{v_j'}}$. Hence the proposition follows. □

A.6. Proof for the Estimation Using the Knowledge of Priors

In this section we present the proof for Propositions 5.2 and 5.3 in Section 5.1.2, i.e., the result on variance estimation using the knowledge of the prior.

Proposition 5.2 (Mean of mutual information estimated using the knowledge of the prior). The expected value $\mathbb{E}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right]$ of the estimated mutual information is given by:

$$\mathbb{E}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right] = I(X;Y) + \sum_{i\in\mathcal{I}} \frac{1}{2n_i} \sum_{y\in\mathcal{Y}^+} \left(\sum_{x\in\mathcal{D}_y} \frac{M_{ixy}}{P_{XY}[x,y]} - \frac{\sum_{x\in\mathcal{D}_y} M_{ixy}}{P_Y[y]} \right) + \mathcal{O}(n_i^{-2}).$$

Proof. Since the precise prior is provided to the analyst, we have

$$\mathbb{E}\left[\hat{I}(X;Y)\right] = \mathbb{E}[H(X)] + \mathbb{E}\left[\hat{H}(Y)\right] - \mathbb{E}\left[\hat{H}(X,Y)\right].$$

By using the results on $\mathbb{E}\left[\hat{H}(Y)\right]$ and $\mathbb{E}\left[\hat{H}(X,Y)\right]$ in the proof of Theorem 4.1, we obtain the proposition. □

Proposition 5.3 (Variance of mutual information estimated using the knowledge of the prior). The variance $\mathbb{V}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right]$ of the estimated mutual information is given by:

$$\mathbb{V}\left[\hat{I}_{\Theta,\Lambda}(X;Y)\right] = \sum_{i\in\mathcal{I}} \sum_{x\in\mathcal{X}^+} \frac{\theta_{ix}^2}{n_i\lambda_i[x]} \left(\sum_{y\in\mathcal{D}_x} D_i[y|x] \left(\log \frac{P_Y[y]}{P_{XY}[x,y]} \right)^2 - \left(\sum_{y\in\mathcal{D}_x} D_i[y|x] \left(\log \frac{P_Y[y]}{P_{XY}[x,y]} \right) \right)^2 \right) + \mathcal{O}(n_i^{-2}).$$

Proof. Since the precise prior is provided to the analyst, we have

$$\begin{aligned}\mathbb{V}\Big[\hat{I}(X;Y)\Big] &= \mathbb{V}\Big[H(X) + \hat{H}(Y) - \hat{H}(X,Y)\Big] \\ &= \mathbb{V}\Big[\hat{H}(Y)\Big] + \mathbb{V}\Big[\hat{H}(X,Y)\Big] - 2\mathit{Cov}\Big[\hat{H}(Y), \hat{H}(X,Y)\Big].\end{aligned}$$

By using the results on $\mathbb{V}\Big[\hat{H}(Y)\Big]$, $\mathbb{V}\Big[\hat{H}(X,Y)\Big]$ $\mathit{Cov}\Big[\hat{H}(Y), \hat{H}(X,Y)\Big]$ in the proof of Theorem 4.2, we obtain the proposition. □