Combining Static and Statistical Approaches to Quantitative Information Flow

Yusuke Kawamoto, Fabrizio Biondi, Axel Legay

Combining Static and Statistical Approaches to Quantitative Information Flow

Yusuke Kawamoto[1], Fabrizio Biondi[2], and Axel Legay[2]

[1] AIST, Japan
[2] INRIA, France

Abstract. Systems dealing with confidential data may leak some information by their observable outputs. *Quantitative information flow analysis* provides a method for quantifying the amount of such information leakage. To avoid the high computational cost of exhaustive search, *statistical analysis* has been studied to estimate information leakage by analyzing only a small but representative subset of the system's behavior. In this paper we propose a new *compositional statistical analysis method* for quantitative information flow that combines multiple statistical analyses with static *trace analysis*. We use partial knowledge of the system's source code or specification, therefore improving both quality and cost of the analysis. The new method can optimize the use of weighted statistical analysis by performing it on components of the system and appropriately adapting their weights. We show this approach combined with the precision of trace analysis produces better estimates and narrower confidence intervals than the state of the art.

1 Introduction

Detecting and quantifying information leaks in computer systems are important in evaluating and controlling the security and privacy levels. *Quantitative information flow analysis* provides techniques to compute the amount of leaked confidential information using information theory [4,12,13,19,20,23,24,31,32,36]. In leakage computation, the system is modeled as an *information-theoretical channel matrix* between the secret and observable output of the system; i.e., a matrix that represents the conditional probability distribution of observables given secrets.

Information leakage of white-box systems can be computed precisely by analyzing the system trace by trace. However, static analysis tools may not be able to compute the leakage when the system has too many possible traces, or when the analyst cannot access the source code of the whole system. To overcome these problems, statistical analysis methods have been developed to estimate information leakage by analyzing only a small but representative subset of the system's behavior [18,21,22]. More specifically, the analyst randomly runs the system a certain number of times and computes an approximated leakage result from the set of randomly generated traces of the system. Note that statistical analysis does not require the source code of the system, but the possibility to simulate it; therefore it can be applied to the case in which only a black-box view of the system is available.

Although the leakage values estimated by the statistical methods are not precise, the analyst is able to learn the quality of results from the confidence interval sizes of the estimates. On the other hand, the drawback of the statistical methods is that a large number of simulations is required to produce a reliable result (with a narrow confidence interval), especially when the size of the channel matrix is large.

In this paper we propose a new approach in which precise trace analysis and approximate statistical analysis are combined in a compositional way, exploiting the advantages of both methods.

The new method decomposes the system into *components* that are defined as closed (terminal) components of the control flow graph of the system, i.e., parts of the control flow graph in which the system terminates and produces output. We assume that some components of the system can be accessible as white-box while some others only as black-box, and similarly some components can easily be analyzed with trace analysis and some with statistical analysis. By applying an appropriate analysis method to each component and combining the results we obtain a compositional method that is more effective than trace or statistical analysis alone.

More specifically, we use trace analysis on the parts of the system that are convenient for trace analysis, producing a single precise sub-channel matrix (i.e., a channel matrix in which the sum of each row can be smaller than 1). Then we apply statistical analysis to each component that is not easy to analyze with trace analysis, either because it has too many traces or because its code is not accessible. Each statistical analysis also produces an empirical sub-channel matrix together with the variance of the estimate. Finally, we combine all the sub-channel matrices to obtain the full channel matrix of the system, and from it we obtain the leakage estimate and its confidence interval.

Note that the construction of the control flow graph of the system is possible only for the components providing white-box access. The control flow graph is constructed to drive the analysis, including to decide which components to analyze precisely or statistically. It is also used to adaptively initialize the statistical analysis with appropriate priors and number of simulations. This is reminiscent of the importance sampling techniques in statistical model checking.

1.1 Motivating Examples

We illustrate the motivation of our compositional method by the two examples in Fig. 1.

```
1 new secret h = Uniform(0...2^64 − 1);
2 if(h<10)
3   new r = (Uniform(0...2^64 − 1)%10) + 1;
4   print h % r;
5 else
6   new r = Uniform(0...9);
7   print r;
```

(a) Example with disjoint channel matrices

```
1 new secret h = Uniform(0...1);
2 new b = {true ↦ p, false ↦ 1 − p};
3 if(b)
4   print h;
5 else
6   print ~h;
```

(b) Example with overlapping channel matrices

Fig. 1: Motivating examples

Let us first consider the channel matrix for the program in Fig. 1a. The program's secret variable `h` has 2^{64} possible values and the program has 10 possible outputs, so the channel matrix would have 2^{64} rows and 10 columns. This is too large to analyze with statistical methods, that have a cost at least linear in the number of rows of the matrix.

Instead, let's consider the trace analysis of the program. Apparently the program only has two traces: one if the guard of the conditional `(h<10)` on line 2 evaluates to true, and the other if it evaluates to false. However, trace analysis requires the precise evaluation of the probabilities of non-secret variables. In this case, it means evaluating all possible traces produced by the assignment to the variable `r` on line 3. While in this case the variable `r` can only be assigned a number from 1 to 10, knowing the probability of each of the outputs of the function would require a trace analysis algorithm to analyze all 2^{64} traces in the random function, which is again unfeasible.

So it would seem that neither statistical analysis nor trace analysis would be able to produce results in reasonable time for this program: statistical analysis is sensitive to the number of secret values and trace analysis to the number of traces, and the program in Fig. 1a has a very large number of both secret values and traces.

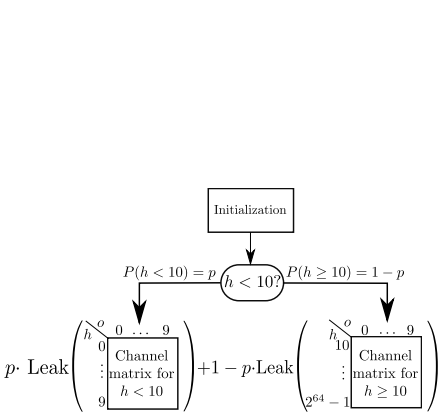
However, combining both the analysis methods can produce a new method that is more effective than each method alone. Consider the program in Fig. 1a again. Note that most of the traces of the program are produced on line 3, when the variable `r` is initialized. Also, on line 3 the guard of the conditional on line 2 must be true, meaning that the secret `h` has only 10 possible secrets, from 0 to 9. Consequently, initialization of the value of the variable `r` on line 3 is suitable for statistical analysis, since it has only a small number of secret values. Once lines 3 and 4 have been analyzed statistically, most of the traces of the program have been analyzed. This makes it possible for trace analysis to analyze all remaining traces and combine the results of the statistical analysis of lines 3 and 4, efficiently obtaining a leakage result for a program that neither analysis method alone would have been able to handle.

The example we have presented above shows the main motivation behind this paper: since trace analysis and statistical analysis have different strengths and weaknesses, combining them produces an analysis method able to deal with programs that neither technique would have been able to deal with alone.

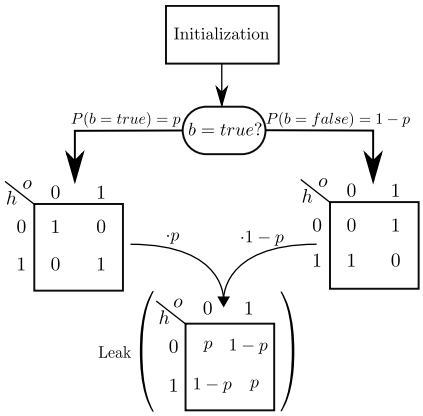
In the particular case of the example in Fig. 1a, note that the conditional `(h<10)` splits the system into two components with separate secret values. This means the two channel matrices for $\mathtt{h} < 10$ and $\mathtt{h} \geq 10$ are disjoint. Then the amount of information leaked by the system can be obtained by calculating the leakages of the disjoint components and weighting them by the probabilities of the components, as depicted in Fig. 2a. A similar procedure can be used when the components have disjoint output values.

However, in general, more than one components may have common secret and output values. Consider the program in Fig. 1b. The program has a secret bit `h`, and prints it with probability p or its negation with probability $1 - p$. The two cases have overlapping sub-channel matrices, since both of them have the same secret and observable domains $\{0, 1\}$. In this case the leakage value cannot be computed from those of the components [3]. Therefore it is necessary to compute a single channel matrix for the whole system to compute the leakage, as shown in Fig. 2b. If some of the channel matrices of the

[3] See [29,30] for more discussion on the compositionality of leakage measures.



(a) Compositional computation for the example in Fig. 1a. Since the two channel matrices are disjoint, the total leakage is the weighted sum of the leakage of the two components.



(b) Compositional computation for the example in Fig. 1b. Since the two channel matrices are overlapping, they need to be combined into a matrix for the whole system.

Fig. 2: Compositional leakage computation for the motivational examples.

components have been estimated statistically, the variance of the estimates has to be considered to compute the confidence interval of the estimate of the whole system.

1.2 Contributions

This paper provides the following contributions:

- We propose a compositional method combining statistical and trace analyses based on partial knowledge of the behavior of a system (e.g., its control flow graph);
- We show (non-trivial) theorems on compositionally computing the confidence interval from multiple statistical simulations;
- We evaluate the quality of leakage estimates in this method, showing the estimates are more accurate than statistical analysis alone for the same number of simulations;
- We discuss how to decide whether we use statistical or trace analysis on the components of the system depending on their characteristics;
- We present an extension of the compositional method to adaptively optimize parameters of statistical simulations by evaluating the quality and cost of the analysis;
- We show the effectiveness of the compositional method in a case study on a protocol downgrade attack by comparing it with the state-of-the-art statistical analysis tool LeakWatch [22].

The rest of the paper is structured as follows. Section 2 introduces some background in trace analysis and statistical analysis for leakage computation. Section 3 describes the compositional statistical method representing the main technical contributions of the paper. Section 4 evaluates the quality and cost of the compositional statistical method, and derives its extension to adaptive optimization. Section 5 shows its effectiveness against the state of the art in statistical analysis on a protocol downgrade attack case study.

Section 6 discusses related work, and Section 7 discusses future work and concludes the paper. All proofs can be found in Appendix of the paper.

2 Background

We introduce the basic concepts that will be used in the rest of the paper. Given a set $\mathcal{S}$, we will denote the number of elements of $\mathcal{S}$ as $\#\mathcal{S}$.

2.1 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 in the studies on quantitative information flow to formalize information leakage in a system that processes confidential data: inputs and outputs of a channel are regarded as secrets and observables in the system respectively, and the channel represents some correlation between secrets and observables in the system.

A *discrete channel* is a triple $(\mathcal{X}, \mathcal{Y}, C)$ where $\mathcal{X}$ and $\mathcal{Y}$ are two finite sets of discrete secret and observable values respectively and C is an $\#\mathcal{X} \times \#\mathcal{Y}$ matrix where each element $C[x, y]$ represents the conditional probability of an observable y given a secret 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 secret values $\mathcal{X}$. Given a prior π over $\mathcal{X}$ and a channel C over $\mathcal{X}$ and $\mathcal{Y}$, the *joint probability distribution* of X and Y is defined by: $P[x, y] = \pi[x]C[x, y]$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

2.2 Information Leakage

The goal of quantitative information flow analysis is to quantify how much information is leaked to an attacker that observes the system's output and infers about the system's secret. We assume the attacker can execute and observe the system as well as (possibly partially) obtain the source code or specification of the system.

The amount of leaked information is defined as the reduction in the attacker's uncertainty about the secret by observing the output. The uncertainty is commonly modeled using the *Shannon entropy*. Formally, given a prior π on secrets X, the *prior uncertainty* (before observing the system's output) is defined as

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

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

$$H(X|Y) = -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log_2 p(x|y),$$

where $p(y) = \sum_{x' \in \mathcal{X}} \pi[x']C[x', y]$ and $p(x|y) = \frac{\pi[x]C[x,y]}{p(y)}$ if $p(y) \neq 0$. Then the information leakage is defined as the *mutual information* (i.e., the difference between the uncertainty before and after observing the output of the system):

$$I(X;Y) = H(X) - H(X|Y) = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \pi[x]C[x, y] \log_2 \left(\frac{\pi[x]C[x,y]}{\sum_{x' \in \mathcal{X}} C[x', y]} \right).$$

Data: multiset $Stat$ of final states from depth-first trace analysis
Result: Channel $C[\mathtt{h}, \mathtt{o}]$
Initialize $\pi[\mathtt{h}]$ and $C[\mathtt{h}, \mathtt{o}]$ to zero;
foreach $(pr, P) \in Stat$, $\bar{\mathtt{h}} \in \mathcal{D}(\mathtt{h})$ **do**
  Set $\pi[\mathtt{h} = \bar{\mathtt{h}}] \leftarrow \pi[\mathtt{h} = \bar{\mathtt{h}}] \ + \ pr \cdot P(\mathtt{h} = \bar{\mathtt{h}})$;
end
foreach $(pr, P) \in Stat$, $\bar{\mathtt{h}} \in \mathcal{D}(\mathtt{h})$, $\bar{\mathtt{o}} \in \mathcal{D}(\mathtt{o})$ **do**
  Set $C[\mathtt{h} = \bar{\mathtt{h}}, \mathtt{o} = \bar{\mathtt{o}}] \leftarrow C[\mathtt{h} = \bar{\mathtt{h}}, \mathtt{o} = \bar{\mathtt{o}}] \ + \ \dfrac{pr \cdot P(\mathtt{h} = \bar{\mathtt{h}}, \mathtt{o} = \bar{\mathtt{o}})}{\pi[\mathtt{h} = \bar{\mathtt{h}}]}$;
end
Return $C[\mathtt{h}, \mathtt{o}]$;

Algorithm 1: Channel matrix computation by trace analysis

2.3 Trace Analysis

Trace analysis is a technique for reasoning about a system by analyzing all possible execution traces of the system. The application to information leakage computation uses probabilistic inference algorithms [10]. To analyze all traces, this technique assumes the termination of the system and requires the source code or specification of the system.

To perform trace analysis for leakage, the probability distribution over the secret and observable variables is computed for each trace of the system, and the data collected from the traces are combined to obtain the joint distribution of the secrets and observables after searching all traces of the system. More specifically, the trace analysis performs a depth-first search of the traces in the system and for each trace produces its final state (pr, P) that consists of the probability pr of the trace and the joint probability distribution P of all variables after the execution of the trace.

The set $Stat$ of states is processed by Algorithm 1 to produce the channel matrix from the secret variable $\mathtt{h}$ to the secret variable $\mathtt{o}$ [4]. Here we assume that the system has a high-level variable $\mathtt{h}$ ranging over a domain $\mathcal{D}(\mathtt{h})$ of secret values and an observable variable $\mathtt{o}$ ranging over a domain $\mathcal{D}(\mathtt{o})$ of observable values [5]. In the algorithm we first compute the prior $\pi[\mathtt{h}]$ by weighting the projection on $\mathtt{h}$ of each joint distribution P by the probability pr of the trace that has generated P. Then we similarly compute the conditional probability of each possible value of $\mathtt{o}$ given each value of $\mathtt{h}$, obtaining the conditional distribution in the form of a channel matrix.

Importantly, the advantage of the trace analysis is to compute the exact leakage value, without bias or errors. On the other hand, the cost of trace analysis is proportional to the number of traces in the system, which is usually very large in real-world systems.

2.4 Statistical Analysis

Statistical analysis is a technique to compute entropy-based measures such as mutual information [14,18,33] and min-entropy leakage [22]. To perform statistical leakage

[4] Algorithm 1 is a straightforward adaptation of Algorithm 1 from [10] to produce a channel matrix instead of a posterior uncertainty value.

[5] The generalization to multiple secret or output variables is straightforward.

analysis, the system is randomly executed a certain number of times and the execution traces that record pairs of secrets and observables are collected. This set of traces is used to estimate the empirical joint probability distribution $\hat{P}$ over secrets $\mathcal{X}$ and observables $\mathcal{Y}$ and then to compute leakage value such as mutual information $I(X;Y)$.

In the statistical analysis the empirical distribution $\hat{P}$ is different from the true distribution P and therefore the leakage estimate may be very different from the true value. In fact, it is known that entropy-based measures such as mutual information and min-entropy leakage have some bias and error that depend on the number of collected traces, the size of channel matrices and other factors. To sort out this issue we use results on statistics to correct the bias of the estimated leakage value as well as compute the 95% confidence interval of the estimate. In this way we can quantify the error and guarantee the quality of the estimation, which differentiates our approach from *testing*.

The cost of the statistical analysis is proportional to $\#\mathcal{X} \times \#\mathcal{Y}$, the number of elements in the channel matrix. Therefore, this method is significantly more efficient than the trace analysis if the channel matrix is relatively small and the number of traces is very large. On the other hand, if the matrix is very large, the number of execution traces needs to be very large to obtain a reliable and small confidence interval of the leakage estimate. In particular, for a small sample size, statistical analysis does not detect rare events, i.e., traces with a low probability that affects the result. To overcome these difficulties we introduce a new approach that integrates both trace and statistical analyses in Section 3.

3 Compositional Statistical Approach to Estimating Mutual Information

To improve the efficiency of statistical estimation of mutual information we propose a new approach, called *compositional statistical approach*, that integrates both static and dynamic (statistical) analyses. One of the main advantages is that we can guarantee the quality of the outcome by providing its 95% confidence interval in spite that trace analysis and multiple statistical analyses are combined together. In this section we show how the new approach estimates mutual information and its confidence interval.

3.1 Overview

The goal of our new approach is to estimate mutual information and compute its confidence interval from a given system S. We assume that we can randomly run (some parts of) the system S and record their execution traces as well as learn the other part of S by static analysis (e.g. on the source code or specification). In particular, we assume the (exact) prior distribution π on secrets to be part of the model of the attacker. Since the prior represents knowledge that the attacker possesses about the secret before the attack, the analyst decides which prior to use according to the attacker they want to model.

To compute the joint distribution P of the whole system S, we decompose S into components S_0, S_1, ..., S_m, and compute the sub-distribution of each component S_i. When computing the sub-distributions, we apply trace analysis for some components to obtain their exact sub-distributions, and statistical analysis for the others to obtain their

empirical sub-distributions. Then we obtain an estimated joint probability distribution $\hat{P}$ of S by summing up these sub-distributions.

Finally, we combine the results of these analyses and compute the estimation of mutual information and its 95% confidence interval. In the computation of the confidence interval we introduce a new method by extending previous studies [14,18,33].

The detailed procedure of our mutual information estimate is as follows.

1. Given a system S we decompose S into $(m+1)$ components $S_0, S_1, \ldots, S_m$ by constructing a partial control flow graph of the system S and evaluating the cost of trace and statistical analyses on the components. We discuss this in more depth in Section 4.2.
2. According to this decomposition of S, we also decompose a given prior distribution π on secrets $\mathcal{X}$ into the *sub-priors* $\pi_1, \pi_2, \ldots, \pi_m$; i.e., we compute the sub-distributions π_i such that $\pi_i[x]$ is the probability of executing the component S_i with a secret x. Note that for all $x \in \mathcal{X}$, $\pi[x] = \sum_{i=0,1,\ldots,m} \pi_i[x]$.
3. For the component S_0 we perform trace analysis and obtain the (exact) joint sub-probability distribution Q over $\mathcal{X}$ and $\mathcal{Y}$ from all traces of S_0. Note that $Q[x, y]$ is the probability that the execution of S yields that of the component S_0 and has a secret x and an observable y.
4. For the other components $S_1, S_2, \ldots, S_m$ we apply statistical analyses and estimate their joint sub-distributions $R_1, R_2, \ldots, R_m$ over $\mathcal{X}$ and $\mathcal{Y}$. Let $\mathcal{I} = \{1, 2, \ldots, m\}$. We randomly run each component S_i to collect a certain number n_i of execution traces and construct the empirical sub-distribution $\hat{R}_i$ from the sample. Note that for any $i \in \mathcal{I}$ and $x \in \mathcal{X}$, the projection of $\hat{R}_i$ to $\mathcal{X}$ is fixed: $\sum_{y \in \mathcal{Y}} \hat{R}_i[x, y] = \pi_i[x]$.
5. Finally, we calculate the (whole) joint probability distribution $\hat{P} := Q + \sum_{i \in \mathcal{I}} \hat{R}_i$ and estimate the mutual information between secrets and observables as well as the 95% confidence interval for the estimate.

In the above fourth step the sampling is performed for each secret $x \in \mathcal{X}$ unlike the previous work on LeakWatch (that also estimates the prior statistically). For each $i \in \mathcal{I}$, let n_i be the total number of execution traces for estimating the component S_i. We define some *importance prior* λ_i that is a probability distribution over $\mathcal{X}$ and specifies how many runs of S_i are used for the statistical simulation of S_i with each secret $x \in \mathcal{X}$; i.e., the number of traces of S_i with a secret $x \in \mathcal{X}$ is given by $n_i \lambda_i[x]$. Here we take λ_i such that $\lambda_i[x] = 0$ if and only if $\pi_i[x] = 0$. Finally, we define the *weight* as $\xi_{ix} = \frac{\pi_i[x]}{\lambda_i[x]}$ if $\lambda_i[x] \neq 0$ and $\xi_{ix} = 0$ otherwise for each $i \in \mathcal{I}$ and $x \in \mathcal{X}$.

3.2 Estimate of Leakage and its Confidence Interval

In this section we explain how to compute mutual information and its confidence interval. Before the statistical simulation of each component S_i, we have obtained the prior π, the sub-prior π_i and the joint sub-distribution Q of S_0 by trace analysis, as well as specified the total number n_i of S_i's execution traces and the importance prior λ_i as parameters of the statistical simulation.

For each $i \in \mathcal{I}$, we let D_i be the (true) conditional probability distribution of observables given secrets, where each row of R_i is normalized; i.e., for each $y \in \mathcal{Y}$ and

$x \in \mathcal{X}$ s.t. $\pi_i[x] \neq 0$, we have $D_i[y|x] = \frac{R_i[x,y]}{\pi_i[x]}$. We write $\hat{D}_i$ to denote the empirical distribution of D_i (that we obtain by the statistical simulation).

For each $i \in \mathcal{I}$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, let $K_{ixy} = n_i \lambda_i[x] \hat{D}_i[y|x]$. Then K_{ixy} follows a multinomial distribution with parameters $n_i \lambda_i[x]$ and $D_i[y|x]$ for all $y \in \mathcal{Y}$. Let $\overline{K_{ixy}} = E(K_{ixy})$. Then $\overline{K_{ixy}} = n_i \lambda_i[x] D_i[y|x]$.

Theorem 1. *The estimated joint distribution $\hat{P}$ over $\mathcal{X}$ and $\mathcal{Y}$ of the whole system S is given by:*

$$\hat{P}[x,y] = Q[x,y] + \sum_{i \in \mathcal{I}} \frac{\xi_{ix} K_{ixy}}{n_i}.$$

Using the estimated distribution $\hat{P}$ we can compute the mutual information estimate $\hat{I}(X;Y)$, which is slightly larger than the true value $I(X;Y)$. The following theorem quantifies the bias $E(\hat{I}(X;Y)) - I(X;Y)$, therefore will be used to correct the estimate.

Theorem 2. *The expectation of the mutual information $E(\hat{I}(X;Y))$ is given by:*

$$E(\hat{I}(X;Y)) = I(X;Y) + \sum_{i \in \mathcal{I}} \frac{1}{2n_i^2} \sum_{y \in \mathcal{Y}^+} \left(\sum_{x \in \mathcal{D}_y} \frac{\overline{M_{ixy}}}{P[x,y]} - \frac{\sum_{x \in \mathcal{D}_y} \overline{M_{ixy}}}{\sum_{x \in \mathcal{D}_y} P[x,y]} \right) + \mathcal{O}(n_i^{-2})$$

where $\overline{M_{ixy}} = \xi_{ix}^2 \overline{K_{ixy}} \left(1 - \frac{\overline{K_{ixy}}}{n_i \lambda_i[x]}\right)$.

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

$$pe = \hat{I}(X;Y) - \sum_{i \in \mathcal{I}} \frac{1}{2n_i^2} \sum_{y \in \mathcal{Y}^+} \left(\sum_{x \in \mathcal{D}_y} \frac{\overline{M_{ixy}}}{P[x,y]} - \frac{\sum_{x \in \mathcal{D}_y} \overline{M_{ixy}}}{\sum_{x \in \mathcal{D}_y} P[x,y]} \right).$$

The quality of the mutual information 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 the number of execution traces is large enough. The following gives the variance of the distribution.

Theorem 3. *The variance of the mutual information is as follows:*

$$V(\hat{I}(X;Y)) = \sum_{\substack{i \in \mathcal{I} \\ x \in \mathcal{X}^+}} \frac{\pi_i[x]^2}{n_i \lambda_i[x]} \left(\sum_{y \in \mathcal{D}_x} D_i[y|x] \left(\log \frac{P[x,y]}{P[y]} \right)^2 - \left(\sum_{y \in \mathcal{D}_x} D_i[y|x] \log \frac{P[x,y]}{P[y]} \right)^2 \right) + \mathcal{O}(n_i^{-2})$$

where $P[y] = \sum_{x' \in \mathcal{X}'} P[x', y]$.

The confidence interval of the mutual information estimate is calculated using the variance v obtained by Theorem 3. 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:

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

In our leakage estimation in Section 3.1 we consider the confidence level 0.95, therefore use the z-score $z_{0.0025} = 1.96$ to compute the 95% confidence interval.

Note that Theorems 2 and 3 can be regarded as a generalization of [18]. In fact, when $\mathcal{I} = \{1\}$, $\overline{K_{1xy}} \neq 0$, $Q[x,y] = 0$ and $\lambda_1[x] = \pi[x]$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, then the formulas for expectation

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

and variance instantiated with these values coincide with Theorem 6.2 in [18].

4 Evaluation and Control of the Quality and Cost of Analyses

In this section we evaluate the quality and computational cost of the compositional statistical method for mutual information estimation. Then we present how we improve the quality and cost of analyses by adaptively changing parameters.

4.1 Quality of Analysis

The compositional statistical analysis method presented in this paper produces estimates with a better quality than the state of the art in statistical leakage estimation. Due to the combination of precise trace analysis with statistical analysis, the 95% confidence interval of mutual information estimated by compositional analysis in general smaller than the 95% confidence interval by LeakWatch for the same number of simulations. Additionally, the use of trace analysis can help mitigating the disrupting effect of rare events on statistical simulation. In the next section we will show a case study in which LeakWatch tends to converge to an incorrect leakage result, even increasing the number of simulations, because of the properties of the system under analysis. The compositional analysis instead converges to the correct result, thanks to the mitigation of rare events by precise trace analysis.

Fig. 3 shows the sampling distribution of the mutual information estimate of a system. The graph is obtained from 1000 samples each of which is generated by combining trace analysis on a component and statistical analysis on 3 components (using 5000 randomly generated traces). The estimate after the correction of bias is roughly between the lower and upper bounds of the 95% confidence interval calculated from Theorem 3. The corrected estimate is closer to the true value than the biased one.

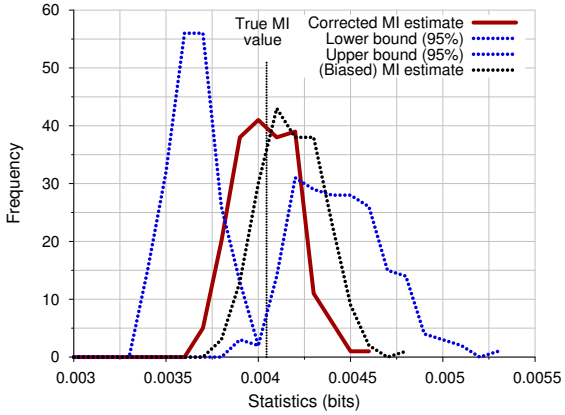
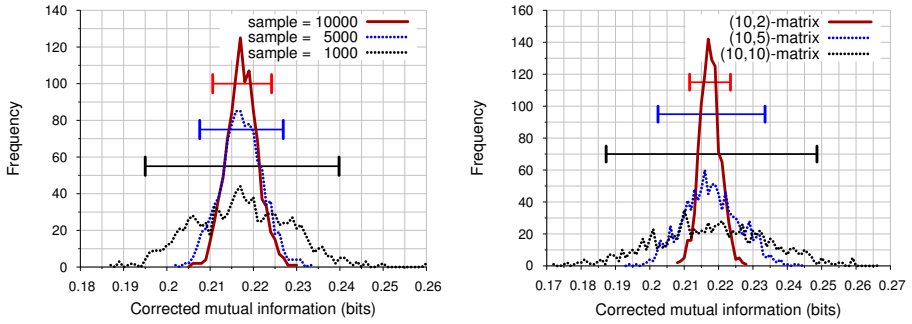


Fig. 3: Distribution of the corrected MI estimate and its confidence interval (3 statistical + 1 trace analyses)

The size of the confidence interval depends on the sample size in the statistical simulation. Fig. 4a illustrates this by using an example with a randomly generated 10×10 channel matrix and a uniform prior. When the sample size is k times larger then the confidence interval is $\sqrt{k}$ times narrower.

The size of the confidence interval also depends on the amount of trace analysis. If we perform trace analysis on a large component and statistical analysis on smaller components, then the sampling distribution becomes more centered (with shorter tails) and the confidence interval becomes narrower. For instance, Fig. 4b shows the statistical analysis only on a smaller component (with a 10×2 sub-channel matrix) yields a smaller confidence interval than that on the whole system (with the randomly generated 10×10 matrix). This means our compositional analysis has a better quality than LeakWatch in that the estimate has a smaller confidence interval for the same sample size.



(a) Corrected MI and sample size (b) Corrected MI and the ratio of trace analysis

Fig. 4: Smaller intervals when increasing the sample size or the ratio of trace analysis.

4.2 Computational Cost of Analysis

The evaluation of the computational cost of the compositional statistical analysis is divided into those of its three parts: the trace analysis, the statistical analysis, and the computation of mutual information estimate and its confidence interval.

The cost of trace analysis of a component is formalized using the number t of its traces and the average cost β of analyzing each trace. While these two values can be estimated by static analysis (e.g. using a control graph of the system), computing them precisely may be very hard. One of the main reasons is that they depend on how the system will behave with conditional branching and loops. In particular, branching and termination may be probabilistic due to the probability distributions on values of variables. Besides, the cost of operations on the distributions has also to be investigated to estimate the cost of each trace. On large distributions, common arithmetical operations may become intractable. In fact, such difficulty in reasoning about distributions is one of the reasons why statistical analysis is often more convenient than trace analysis.

The computational cost of the statistical analysis of each component S_i generally depends the sample size n_i (i.e. the number of random executions of S_i) and the average cost γ_i of executing the component. To obtain a statistically significant estimate, the sample size n_i for analyzing S_i needs to be larger than $(4 \times \#\mathcal{X}_i \times \#\mathcal{Y}_i)$ when S_i is formalized as an $\#\mathcal{X}_i \times \#\mathcal{Y}_i$ matrix [22]. When we choose the sample size n_i we take into account the trade-off between quality and cost: a larger sample size provides a smaller confidence interval of the estimate, therefore better quality of the analysis, while the computational cost increases proportionally to the sample size.

The values of β and γ are to be estimated on the particular system under analysis. For instance, in an experiment on the Dining Cryptographers protocol we have estimated $\beta = 1800$ traces per second and $\gamma = 3200$ simulations per second.

Once the channel matrices of the components have been computed by trace analysis or estimated by statistical analysis, we combine them into the channel matrix of the system to be able to compute information leakage. In the worst case this requires a number of operations proportional to the size of the channel matrix of the system times the number of components. However, the cost δ is reduced if some components have disjoint matrices, and can be further reduced by using efficient data structures for the compact representation of channel matrices, such as abstract channel matrices [15].

To summarize the above, the total cost of the analysis is given by the following:

$$cost = t\beta + \sum_{i \in \mathcal{I}} n_i \gamma_i + \delta.$$

4.3 Extension to Adaptive Analysis

In this section we present an extension of the compositional statistical analysis to improve the quality and computational cost. As discussed in the previous sections, the quality and cost of analyzing a system S depends on

- how we decompose the system S into components S_i for $i \in \mathcal{I}$,
- on which components S_i we perform the trace and statistical analyses, and
- how we determine parameters such as sample sizes n_i and importance priors λ_i.

This implies we may be able to improve the quality and cost by changing these factors.

The key is how we estimate the quality and cost of the analysis. Unlike the difficulty in estimating the computational cost, the quality can be easily estimated dynamically. We run part of statistical simulations to learn the variance of mutual information in each statistical analysis and then decide parameters for further statistical simulations.

More formally, let v_i be the following intermediate value of the variance calculated from traces of the component S_i:

$$v_i = \sum_{x \in \mathcal{X}^+} \frac{\pi_i[x]^2}{\lambda_i[x]} \left(\sum_{y \in \mathcal{D}_x} \hat{D}_i[y|x] \left(\log \frac{\hat{P}[x,y]}{\hat{P}[y]} \right)^2 - \left(\sum_{y \in \mathcal{D}_x} \hat{D}_i[y|x] \log \frac{\hat{P}[x,y]}{\hat{P}[y]} \right)^2 \right)$$

Then the variance estimate is given by $v = \sum_{i \in \mathcal{I}} \frac{v_i}{n_i}$. Since the values of $\hat{D}_i[y|x]$, $\hat{P}[x,y]$ and $\hat{P}[y]$ are computed from a set of randomly generated traces, they have large errors when the sample size is small. Nevertheless, we can roughly learn which components S_i need more simulations to get a smaller confidence interval; i.e., we can choose the sample sizes n_i's such that both the variance $v = \sum_{i \in \mathcal{I}} \frac{v_i}{n_i}$ and the total sample size $\sum_{i \in \mathcal{I}} n_i$ are small enough (e.g. within the budget of quality and cost).

5 Case Study: Protocol Downgrade Attack

We present a case study highlighting the advantages of the compositional statistical method for the leakage estimation compared to the state of the art.

5.1 Description of Protocol and Attack Scenario

The case study models a *downgrade attack* on a system that normally employs a secure communication protocol but also implements some less secure protocols, typically for backwards compatibility. In the downgrade attack the attacker is able to force the system to use one of these less secure protocols and to take advantage of the weakness to infer information about the secret.

In this case study, we consider a private key cryptosystem using one-time pad to transmit a 6-bit secret s. We assume that the secret s is distributed on the uniform prior π. The one-time pad algorithm consists of computing a ciphertext as the bitwise xor (exclusive disjunction) of the secret s with a randomly generated private key k of the

same length. If each key bit is independent and uniformly distributed, it is impossible to retrieve any information on the secret s from the ciphertext without having the key k.

However, the one-time pad implementation we consider includes debugging and legacy implementations of one-time pad where the key is not produced with the appropriate randomness conditions. Assume that the system has a 2-bit variable `mode` that determines how to produce the key as follows:

`mode` $= 0$ Normal behavior: the key bits are generated independently and uniformly.
`mode` $= 1$ Debug mode: the key is set to 000000, in fact deactivating encryption.
`mode` $= 2$ Lower-quality mode with the legacy Java randomness class `java.Random`.
`mode` $= 3$ Lower-quality mode with a weaker linear congruential generator.

The variable `mode` is set to 0, only using the secure implementation of the protocol. However, the attacker is able to tamper with the memory of the system, having a probability of changing the value of `mode`. We will model this by assuming that the attacker can write two random bits on the variable `mode`, potentially changing the value of the variable and forcing the system to generate the key with one of the unsafe systems described above. Note that the attacker does not know the new value of the variable `mode`. To model a more realistic attack, we will also assume the attacker's success rate is actually only 10%. This will reduce the leakage of the system and make it harder for an analyzer to detect the attack. The code model for the attack is presented in the Appendix.

5.2 Compositional Statistical Analysis of the Attack

The case study is appropriate for the compositional statistical method, because it allows us to divide the system into four components S_0, S_1, S_2, S_3 each corresponding to a possible value of the variable `mode`, analyze the components separately with appropriate techniques and parameters, and combine the results.

The components S_0 for `mode` $= 0$ and S_1 for `mode` $= 1$ are simple enough to be analyzed by trace analysis: in S_0 all of the traces produce the same conditional distribution over the outputs, and in S_1 each trace just outputs the value of the secret. The probability of executing the component S_0 is 0.925, since it gets executed when the attack fails (probability 0.9) and when it succeeds but the mode gets randomly set to 1 anyway (probability 0.25). The component S_1 is executed with probability 0.25 since it gets executed when the attack succeeds and the mode is randomly set to 1. We analyze these two cases using trace analysis, producing a precise joint sub-distribution matrix Q.

On the other hand, the component S_2 for `mode` $= 2$ and S_3 for `mode` $= 3$ are not easy for trace analysis: S_2 uses a pseudorandom generator function whose source code is assumed to be unavailable, while S_3 has 2^{32} traces and thus would be very expensive to analyze with trace analysis. Each of the components is executed with probability 0.25. We apply statistical analysis to these cases to produce the empirical sub-distribution matrices $\hat{R}_2$ for S_2 and $\hat{R}_3$ for S_3 respectively.

Finally, we will combine the precise joint sub-distribution matrix Q of the components S_0 and S_1 with the approximated joint sub-distribution matrices $\hat{R}_2$ and $\hat{R}_3$ of the components S_2 and S_3 to obtain the approximated joint distribution $\hat{P}$ of the whole system, together with the information leakage estimate and its 95% confidence interval.

5.3 Experimental Results

Fig. 5 presents the mutual information leakage computed by the state-of-art tool LeakWatch and by the compositional method and their confidence intervals for a number of simulations from 10000 to 150000. The results show that the compositional statistical method achieves a more stable result and a smaller confidence interval than LeakWatch for the same number of simulations. (Note that LeakWatch does not produce an estimate with less than $\sim$ 30000 simulations.)

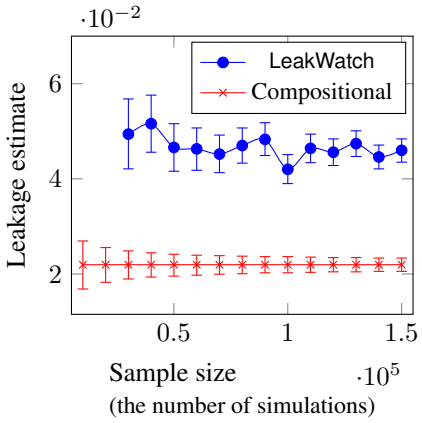


Fig. 5: Leakage estimates and confidence intervals with increasing number of simulations for the protocol downgrade case study.

The tool LeakWatch would converge to an incorrect leakage value with an incorrect confidence interval in some cases. This happens because the bias correction in LeakWatch is approximate and its error grows with the size of the channel matrix. On the other hand, the compositional statistical method employs a more accurate bias correction as well as it performs statistical simulations only where necessary, with the help of trace analysis providing a precise sub-distribution. Therefore, the compositional statistical analysis produces more reliable results than LeakWatch.

Note that we do not compare execution times here, since they depend on other factors in implementation that are irrelevant of comparing the two analysis approaches. In particular, unlike LeakWatch our prototype implementation for the compositional analysis has not been optimized yet; for instance, it has more overhead for I/O processing.

6 Related Work

The statistical approach to quantifying information leakage has been studied since the seminal work by Chatzikokolakis, Chothia and Gupta [18]. Chothia et al. have developed the statistical approach in tools including leakiEst [1,21] and LeakWatch [2,22]. The compositional statistical approach presented in this paper can be considered as an extension of such statistical estimation methods with the inclusion of component weighting and adaptive priors inspired by the recent successes in the application of importance sampling to statistical model checking techniques [8,25]. To the best of our knowledge, no prior work has applied importance sampling to the estimation of mutual information or any other information leakage measures for quantitative information flow.

While Fremont and Seshia have recently presented a polynomial time algorithm to approximate the weight of program traces with possible application to quantitative information leakage, their approach is limited to deterministic programs [27]. Recent progress in statistical program analysis includes an efficient and scalable algorithm for uniform generation of sample from a distribution defined as constraints, with important applications to constrained-random program verification [16,17]

The algorithms for precise computation of information leakage we use in this paper are based on the work of Biondi et al. on the computation of information leakage from

Markovian models and trace analysis [9], implemented in the QUAIL tool [3,11]. Phan et al. developed tools to compute channel capacity of programs written in the C or Java languages, i.e., the maximum leakage over all possible attackers [34,35]. McCamant et al. developed tools implementing dynamic quantitative taint analysis techniques for security, following an alternative approach to security not based on information theory [28]. The recent Moped-QLeak tool by Chadha et al. is able to efficiently compute information leakage of programs as long as it is possible to produce a complete symbolic representation of the program [15].

The problem of computing a channel matrix for a system is strictly related to computing a conditional probability distribution, in our case of the observables conditioned on the secrets. In recent years, the importance of computing conditional probability distributions on Markovian models such as Markov chains and Markov decision processes has been recognized by different authors, particularly for its importance in information leakage analysis [5,6]. Importantly, Baier et al. recently proved that maximal reachability probability under reachability conditions can be computed in polynomial time [7]. We expect the compositional statistical approach we present in this paper to be extendable to the approximated computation of the probability of PCTL or LTL properties.

7 Conclusions and Future Work

We have presented the compositional statistical approach that combines trace analysis and statistical analysis to compute the amount of information leakage of systems. We have shown a compositional method for computing the confidence interval of mutual information estimate from statistical simulations of multiple components of the system. We have discussed how the use of precise trace analysis significantly improves the quality of the estimation compared to statistical analysis alone. The results can also be used to perform an adaptive statistical analysis on the components by dynamically estimating the quality and cost of the analysis. We have presented both theoretical results and a case study to prove that the compositional approach outperforms the state of the art.

As we have shown in Section 4, increasing the percentage of the system analyzed by trace analysis significantly increases the quality of the result, since trace analysis provides a precise sub-distribution. Consequently, improving the efficiency of trace analysis enable the analyst to use it on a larger part of the system and obtain better results. We are developing theory and tools that integrate abstract techniques in program analysis into our compositional statistical approach.

References

1. leakiEst. http://www.cs.bham.ac.uk/research/projects/infotools/leakiest/.
2. LeakWatch. http://www.cs.bham.ac.uk/research/projects/infotools/leakwatch/.
3. QUAIL. https://project.inria.fr/quail/.
4. M. S. Alvim, K. Chatzikokolakis, C. Palamidessi, and G. Smith. Measuring information leakage using generalized gain functions. In S. Chong, editor, *CSF 2012. Proceedings*, pages 265–279. IEEE, 2012.
5. M. E. Andrés. Quantitative analysis of information leakage in probabilistic and nondeterministic systems. *CoRR*, abs/1111.2760, 2011.

6. M. E. Andrés and P. van Rossum. Conditional probabilities over probabilistic and nondeterministic systems. In C. R. Ramakrishnan and J. Rehof, editors, *TACAS 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 157–172. Springer, 2008.
7. C. Baier, J. Klein, S. Klüppelholz, and S. Märcker. Computing conditional probabilities in markovian models efficiently. In E. Ábrahám and K. Havelund, editors, *TACAS 2014. Proceedings*, volume 8413 of *Lecture Notes in Computer Science*, pages 515–530. Springer, 2014.
8. B. Barbot, S. Haddad, and C. Picaronny. Coupling and importance sampling for statistical model checking. In C. Flanagan and B. König, editors, *TACAS 2012. Proceedings*, volume 7214 of *Lecture Notes in Computer Science*, pages 331–346. Springer, 2012.
9. F. Biondi, A. Legay, P. Malacaria, and A. Wasowski. Quantifying information leakage of randomized protocols. *Theor. Comput. Sci.*, 597:62–87, 2015.
10. F. Biondi, A. Legay, and J. Quilbeuf. Comparative analysis of leakage tools on scalable case studies. In B. Fischer and J. Geldenhuys, editors, *SPIN 2015, Proceedings*, volume 9232 of *Lecture Notes in Computer Science*, pages 263–281. Springer, 2015.
11. F. Biondi, A. Legay, L. Traonouez, and A. Wasowski. QUAIL: A quantitative security analyzer for imperative code. In N. Sharygina and H. Veith, editors, *CAV 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 702–707. Springer, 2013.
12. M. Boreale. Quantifying information leakage in process calculi. *Inf. Comput.*, 207(6):699–725, 2009.
13. M. Boreale, F. Pampaloni, and M. Paolini. Asymptotic information leakage under one-try attacks. In *Proc. of FOSSACS*, pages 396–410, 2011.
14. D. R. Brillinger. Some data analysis using mutual information. *Brazilian Journal of Probability and Statistics*, 18(6):163–183, 2004.
15. R. Chadha, U. Mathur, and S. Schwoon. Computing information flow using symbolic model-checking. In *FSTTCS 2014. Proceedings*, pages 505–516, 2014.
16. S. Chakraborty, D. J. Fremont, K. S. Meel, S. A. Seshia, and M. Y. Vardi. On parallel scalable uniform SAT witness generation. In C. Baier and C. Tinelli, editors, *TACAS 2015. Proceedings*, volume 9035 of *Lecture Notes in Computer Science*, pages 304–319. Springer, 2015.
17. S. Chakraborty, K. S. Meel, and M. Y. Vardi. A scalable approximate model counter. In C. Schulte, editor, *CP 2013. Proceedings*, volume 8124 of *Lecture Notes in Computer Science*, pages 200–216. Springer, 2013.
18. K. Chatzikokolakis, T. Chothia, and A. Guha. Statistical measurement of information leakage. In J. Esparza and R. Majumdar, editors, *TACAS 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 390–404. Springer, 2010.
19. K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. Anonymity protocols as noisy channels. *Inf. and Comp.*, 206(2–4):378–401, 2008.
20. K. Chatzikokolakis, C. Palamidessi, and P. Panangaden. On the Bayes risk in information-hiding protocols. *J. of Comp. Security*, 16(5):531–571, 2008.
21. T. Chothia, Y. Kawamoto, and C. Novakovic. A tool for estimating information leakage. In N. Sharygina and H. Veith, editors, *CAV 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 690–695. Springer, 2013.
22. T. Chothia, Y. Kawamoto, and C. Novakovic. Leakwatch: Estimating information leakage from java programs. In M. Kutylowski and J. Vaidya, editors, *ESORICS 2014. Proceedings, Part II*, volume 8713 of *Lecture Notes in Computer Science*, pages 219–236. Springer, 2014.
23. T. Chothia, Y. Kawamoto, C. Novakovic, and D. Parker. Probabilistic point-to-point information leakage. In *CSF 2013. Proceedings*, pages 193–205. IEEE, 2013.
24. D. Clark, S. Hunt, and P. Malacaria. Quantitative analysis of the leakage of confidential data. *Electr. Notes Theor. Comput. Sci.*, 59(3):238–251, 2001.

25. E. M. Clarke and P. Zuliani. Statistical model checking for cyber-physical systems. In T. Bultan and P. Hsiung, editors, *ATVA 2011. Proceedings*, volume 6996 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2011.
26. T. Cover and J. Thomas. *Elements of Information Theory*. A Wiley-Interscience publication. Wiley, 2006.
27. D. J. Fremont and S. A. Seshia. Speeding up smt-based quantitative program analysis. In P. Rümmer and C. M. Wintersteiger, editors, *SMT 2014. Proceedings*, volume 1163 of *CEUR Workshop Proceedings*, pages 3–13. CEUR-WS.org, 2014.
28. M. G. Kang, S. McCamant, P. Poosankam, and D. Song. DTA++: dynamic taint analysis with targeted control-flow propagation. In *NDSS 2011. Proceedings*, 2011.
29. Y. Kawamoto, K. Chatzikokolakis, and C. Palamidessi. Compositionality results for quantitative information flow. In *QEST 2014. Proceedings*, pages 368–383, 2014.
30. Y. Kawamoto and T. Given-Wilson. Quantitative information flow for scheduler-dependent systems. In *QAPL 2015. Proceedings*, volume 194, pages 48–62, 2015.
31. B. Köpf and D. A. Basin. An information-theoretic model for adaptive side-channel attacks. In *Proc. of CCS*, pages 286–296. ACM, 2007.
32. P. Malacaria. Assessing security threats of looping constructs. In *Proc. of POPL*, pages 225–235. ACM, 2007.
33. R. Moddemeijer. On estimation of entropy and mutual information of continuous distributions. *Signal Processing*, 16:233–248, 1989.
34. Q. Phan and P. Malacaria. Abstract model counting: a novel approach for quantification of information leaks. In S. Moriai, T. Jaeger, and K. Sakurai, editors, *AsiaCCS 2014. Proceedings*, pages 283–292. ACM, 2014.
35. Q. Phan, P. Malacaria, C. S. Pasareanu, and M. d'Amorim. Quantifying information leaks using reliability analysis. In N. Rungta and O. Tkachuk, editors, *SPIN 2014. Proceedings*, pages 105–108. ACM, 2014.
36. G. Smith. On the foundations of quantitative information flow. In L. de Alfaro, editor, *FOSSACS 2009. Proceedings*, volume 5504 of *Lecture Notes in Computer Science*, pages 288–302. Springer, 2009.

A Omitted Proofs

We present proofs omitted in the paper. We refer to literature [26] for the definitions of the variance $V(X)$ and the covariance $Cov(X, Y)$.

Proof (of Theorem 1). By definitions the joint sub-distribution $\hat{R}_i$ of each component S_i is given by:

$$\hat{R}_i[x, y] = \pi_i[x]\hat{D}_i[y|x] = \pi_i[x]\frac{K_{ixy}}{n_i\lambda_i[x]} = \frac{\xi_{ix}K_{ixy}}{n_i}.$$

Since the joint distribution of the whole system S is the summation of the distributions of all components, we obtain the following:

$$\hat{P}[x, y] = Q[x, y] + \sum_{i\in\mathcal{I}} \hat{R}_i[x, y] = Q[x, y] + \frac{\xi_{ix}K_{ixy}}{n_i}.$$

□

Proof (of Theorem 2). We derive the expectation of the estimate as follows. For abbreviation we write q_{xy} to denote $Q[x, y]$.

The empirical joint entropy is defined by

$$\hat{H}(X, Y) = -\sum_{x\in\mathcal{X}, y\in\mathcal{Y}} \left(q_{xy} + \sum_{i\in\mathcal{I}} \frac{\xi_{ix}K_{ixy}}{n_i}\right) \log\left(q_{xy} + \sum_{i\in\mathcal{I}} \frac{\xi_{ix}K_{ixy}}{n_i}\right).$$

We define the set of pairs consisting of secrets and observables that appear with non-zero probabilities:

$$\mathcal{D} = \left\{(x, y) \in \mathcal{X} \times \mathcal{Y} \colon q_{xy} + \sum_{i\in\mathcal{I}} \frac{\xi_{ix}K_{ixy}}{n_i} \neq 0\right\}.$$

By the Taylor expansion w.r.t. $\overline{K_{ixy}}$ for all $i \in \mathcal{I}$,

$$\begin{aligned}\hat{H}(X, Y) = -\sum_{(x,y)\in\mathcal{D}} (A_{xy} &+ \sum_{i\in\mathcal{I}} B_{ixy}(K_{ixy} - \overline{K_{ixy}}) \\ &+ \sum_{i,j\in\mathcal{I}, i\neq j} C_{ijxy}(K_{ixy} - \overline{K_{ixy}})(K_{jxy} - \overline{K_{jxy}}) \\ &+ \sum_{i\in\mathcal{I}} C_{iixy}(K_{ixy} - \overline{K_{ixy}})^2 + \mathcal{O}(n_i^{-2}))\end{aligned}$$

where

- $A_{xy} = \left(q_{xy} + \sum_{l\in\mathcal{I}} \frac{\xi_{lx}\overline{K_{lxy}}}{n_l}\right) \log\left(q_{xy} + \sum_{l\in\mathcal{I}} \frac{\xi_{lx}\overline{K_{lxy}}}{n_l}\right)$,
- $B_{ixy} = \frac{\xi_{ix}}{n_i}\left(1 + \log\left(q_{xy} + \sum_{l\in\mathcal{I}} \frac{\xi_{lx}\overline{K_{lxy}}}{n_l}\right)\right)$,

– $C_{ijxy} = \dfrac{\xi_{ix}\xi_{jx}}{2n_i n_j \left(q_{xy} + \sum_{l\in\mathcal{I}} \frac{\xi_{lx}\overline{K_{lxy}}}{n_l}\right)}$,

The expectation of $E(\hat{H}(X,Y))$ is given by:

$$\begin{aligned} E(\hat{H}(X,Y)) &= -\sum_{(x,y)\in\mathcal{D}} (A_{xy} + \sum_{i\in\mathcal{I}} B_{ixy} E(K_{ixy} - \overline{K_{ixy}}) \\ &\qquad + \sum_{i,j\in\mathcal{I},\, i\neq j} C_{ijxy} E\left((K_{ixy} - \overline{K_{ixy}})(K_{jxy} - \overline{K_{jxy}})\right) \\ &\qquad + \sum_{i\in\mathcal{I}} C_{iixy} E\left((K_{ixy} - \overline{K_{ixy}})^2\right) + \mathcal{O}(n_i^{-2})) \\ &= H(X,Y) - \sum_{i\in\mathcal{I}} \frac{1}{2n_i^2} \sum_{(x,y)\in\mathcal{D}} \frac{\xi_{ix}^2 \overline{K_{ixy}} \left(1 - \frac{\overline{K_{ixy}}}{n_i \lambda_i[x]}\right)}{q_{xy} + \sum_{l\in\mathcal{I}} \frac{\xi_{lx}\overline{K_{lxy}}}{n_l}} \end{aligned}$$

Note that if $i \neq j$ then K_{ixy} and K_{jxy} are independent, therefore $E((K_{ixy} - \overline{K_{ixy}})(K_{jxy} - \overline{K_{jxy}})) = 0$.

Next we calculate $E(\hat{H}(Y))$ as follows. Let $\mathcal{D}_x = \{y\colon (x,y) \in \mathcal{D}\}$ and $\mathcal{D}_y = \{x\colon (x,y) \in \mathcal{D}\}$. For each $i = 1, 2, \ldots, m$ and $y \in \mathcal{Y}$ let $L_{i\cdot y} = \sum_{x\in\mathcal{D}_y} \xi_{ix} K_{ixy}$ and $\overline{L_{i\cdot y}} = \sum_{x\in\mathcal{D}_y} \xi_{ix}\overline{K_{ixy}}$. Then the empirical entropy of observables is defined by

$$\hat{H}(Y) = -\sum_{y\in\mathcal{Y}^+} \left(\sum_{x\in\mathcal{D}_y} q_{xy} + \sum_{i\in\mathcal{I}} \frac{L_{i\cdot y}}{n_i}\right) \log \left(\sum_{x\in\mathcal{D}_y} q_{xy} + \sum_{i\in\mathcal{I}} \frac{L_{i\cdot y}}{n_i}\right)$$

where $\mathcal{Y}^+ = \left\{ y \in \mathcal{Y}\colon \sum_{x\in\mathcal{D}_y} q_{xy} + \sum_{i\in\mathcal{I}} \frac{L_{i\cdot y}}{n_i} \neq 0 \right\}$. By the Taylor expansion w.r.t. $\overline{L_{i\cdot y}}$ for all $i \in \mathcal{I}$,

$$\begin{aligned} \hat{H}(Y) = -\sum_{y\in\mathcal{Y}^+} (A_{\cdot y} &+ \sum_{i\in\mathcal{I}} B_{i\cdot y}(L_{i\cdot y} - \overline{L_{i\cdot y}}) \\ &+ \sum_{i,j\in\mathcal{I},\, i\neq j} C_{ij\cdot y}(L_{i\cdot y} - \overline{L_{i\cdot y}})(L_{j\cdot y} - \overline{L_{j\cdot y}}) \\ &+ \sum_{i\in\mathcal{I}} C_{ii\cdot y}(L_{i\cdot y} - \overline{L_{i\cdot y}})^2 + \mathcal{O}(n_i^{-2})) \end{aligned}$$

where

– $A_{\cdot y} = \left(\sum_{x\in\mathcal{D}_y} q_{xy} + \sum_{l\in\mathcal{I}} \frac{\overline{L_{l\cdot y}}}{n_l}\right) \log \left(\sum_{x\in\mathcal{D}_y} q_{xy} + \sum_{l\in\mathcal{I}} \frac{\overline{L_{l\cdot y}}}{n_l}\right)$,

– $B_{i\cdot y} = \dfrac{1}{n_i}\left(1 + \log\left(\sum_{x\in\mathcal{D}_y} q_{xy} + \sum_{l\in\mathcal{I}} \frac{\overline{L_{l\cdot y}}}{n_l}\right)\right)$,

– $C_{ij\cdot y} = \dfrac{1}{2n_i n_j \left(\sum_{x\in\mathcal{D}_y} q_{xy} + \sum_{l\in\mathcal{I}} \frac{\overline{L_{l\cdot y}}}{n_l}\right)}$,

The expectation of $E(\hat{H}(Y))$ is given by:

$$\begin{aligned}
E(\hat{H}(Y)) = -\sum_{y\in\mathcal{Y}^+} (A_{\cdot y} + &\sum_{i\in\mathcal{I}} B_{i\cdot y} E(L_{i\cdot y} - \overline{L_{i\cdot y}}) \\
&+ \sum_{i,j\in\mathcal{I},\, i\neq j} C_{ij\cdot y} E\left((L_{i\cdot y} - \overline{L_{i\cdot y}})(L_{j\cdot y} - \overline{L_{j\cdot y}})\right) \\
&+ \sum_{i\in\mathcal{I}} C_{ii\cdot y} E\left((L_{i\cdot y} - \overline{L_{i\cdot y}})^2\right) + \mathcal{O}(n_i^{-2})) \\
= H(Y) - \sum_{i\in\mathcal{I}} \sum_{y\in\mathcal{Y}^+} C_{ii\cdot y} &\sum_{x\in\mathcal{D}_y} \xi_{ix}^2 \overline{K_{ixy}} \left(1 - \frac{\overline{K_{ixy}}}{n_i \lambda_i[x]}\right) + \mathcal{O}(n_i^{-2}).
\end{aligned}$$

The expectation of the mutual information $E(\hat{I}(X;Y))$ is given by:

$$\begin{aligned}
&E(\hat{I}(X;Y)) \\
&= H(X) + E(\hat{H}(Y)) - E(\hat{H}(X,Y)) \\
&= I(X;Y) + \sum_{i\in\mathcal{I}} \frac{1}{2n_i^2} \sum_{y\in\mathcal{Y}^+} \left(\sum_{x\in\mathcal{D}_y} \frac{\overline{M_{ixy}}}{P[x,y]} - \frac{\sum_{x\in\mathcal{D}_y} \overline{M_{ixy}}}{\sum_{x\in\mathcal{D}_y} P[x,y]} \right) + \mathcal{O}(n_i^{-2})
\end{aligned}$$

where $\overline{M_{ixy}} = \xi_{ix}^2 \overline{K_{ixy}} \left(1 - \frac{\overline{K_{ixy}}}{n_i \lambda_i[x]}\right)$. □

Proof (of Theorem 3). We derive the variance of the estimate as follows.
For each $i, i' \in \{1, 2, \ldots, m\}$, $x, x' \in \mathcal{X}$ and $y, y' \in \mathcal{Y}$ the covariance $Cov(K_{ixy}, K_{i'x'y'})$ is as follows:

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

If $i \neq i'$ or $x \neq x'$, then K_{ixy} and $K_{i'x'y'}$ are independent for any y, y', hence $Cov(K_{ixy}, K_{i'x'y'}) = 0$. Therefore if $i \neq i'$ then $Cov(L_{i\cdot y}, L_{i'y'}) = 0$.

$$\begin{aligned}
&Cov(L_{i\cdot y}, L_{iy'}) \\
&= Cov\left(\textstyle\sum_{x\in\mathcal{D}_y} \xi_{ix} K_{ixy},\, \sum_{x'\in\mathcal{D}_{y'}} \xi_{ix'} K_{ix'y'}\right) \\
&= \textstyle\sum_{x\in\mathcal{D}_y} Cov\left(\xi_{ix} K_{ixy}, \xi_{ix} K_{ixy'}\right) \\
&= \textstyle\sum_{x\in\mathcal{D}_y} \xi_{ix}^2\, Cov\left(K_{ixy}, K_{ixy'}\right) \\
&= \begin{cases}
-n_i \sum_{x\in\mathcal{D}_y} \xi_{ix}^2 \lambda_i[x] D_i[y|x] D_i[y'|x] & \text{if } y \neq y' \\
n_i \sum_{x\in\mathcal{D}_y} \xi_{ix}^2 \lambda_i[x] D_i[y|x](1 - D_i[y|x]) & \text{otherwise.}
\end{cases}
\end{aligned}$$

The variance of $\hat{H}(X,Y)$ is given by the following.

$$
\begin{aligned}
&V(\hat{H}(X,Y)) \\
&= E\left(\hat{H}(X,Y)^2\right) - \left(E(\hat{H}(X,Y))\right)^2 \\
&= \sum_{(x,y)\in\mathcal{D}} \sum_{(x',y')\in\mathcal{D}} \sum_{i\in\mathcal{I}} B_{ixy}B_{ix'y'} E\left((K_{ixy} - \overline{K_{ixy}})(K_{ix'y'} - \overline{K_{ix'y'}})\right) + \mathcal{O}(n_i^{-2}) \\
&= \sum_{i\in\mathcal{I}} \sum_{(x,y)\in\mathcal{D}} n_i\lambda_i[x]\left(B_{ixy}^2 D_i[y|x](1 - D_i[y|x]) - \sum_{y'\in\mathcal{D}_x\setminus\{y\}} B_{ixy}B_{ixy'} D_i[y|x]D_i[y'|x]\right) + \mathcal{O}(n_i^{-2}) \\
&= \sum_{i\in\mathcal{I}} n_i \sum_{(x,y)\in\mathcal{D}} \lambda_i[x]D_i[y|x]B_{ixy}\left(B_{ixy} - \sum_{y'\in\mathcal{D}_x} B_{ixy'}D_i[y'|x]\right) + \mathcal{O}(n_i^{-2})
\end{aligned}
$$

The variance of $\hat{H}(Y)$ is given by the following.

$$
\begin{aligned}
&V(\hat{H}(Y)) \\
&= E\left(\hat{H}(Y)^2\right) - \left(E(\hat{H}(Y))\right)^2 \\
&= \sum_{y\in\mathcal{Y}^+} \sum_{y'\in\mathcal{Y}^+} \sum_{i\in\mathcal{I}} B_{i\cdot y}B_{i\cdot y'} E\left((L_{i\cdot y} - \overline{L_{i\cdot y}})(L_{i\cdot y'} - \overline{L_{i\cdot y'}})\right) + \mathcal{O}(n_i^{-2}) \\
&= \sum_{i\in\mathcal{I}} n_i \sum_{y\in\mathcal{Y}^+} (B_{i\cdot y}^2 \sum_{x\in\mathcal{D}_y} \xi_{ix}^2\lambda_i[x]D_i[y|x](1 - D_i[y|x]) \\
&\qquad\qquad - \sum_{y'\in\mathcal{D}_x\setminus\{y\}} B_{i\cdot y}B_{i\cdot y'} \sum_{x\in\mathcal{D}_y} \xi_{ix}^2\lambda_i[x]D_i[y|x]D_i[y'|x]) \\
&\qquad + \mathcal{O}(n_i^{-2}) \\
&= \sum_{i\in\mathcal{I}} n_i \sum_{(x,y)\in\mathcal{D}} \lambda_i[x]D_i[y|x]\xi_{ix}^2 B_{i\cdot y}\left(B_{i\cdot y} - \sum_{y'\in\mathcal{D}_x} B_{i\cdot y'}D_i[y'|x]\right) + \mathcal{O}(n_i^{-2})
\end{aligned}
$$

The covariance between $\hat{H}(X,Y)$ and $\hat{H}(Y)$ is given by:

$$
\begin{aligned}
&Cov(\hat{H}(Y), \hat{H}(X,Y)) \\
&= \sum_{i\in\mathcal{I}} \sum_{(x,y)\in\mathcal{D}} \sum_{y'\in\mathcal{Y}^+} B_{ixy}B_{i\cdot y'} Cov(K_{ixy}, L_{i\cdot y'}) + \mathcal{O}(n_i^{-2}) \\
&= \sum_{i\in\mathcal{I}} \sum_{(x,y)\in\mathcal{D}} \sum_{y'\in\mathcal{Y}^+} B_{ixy}B_{i\cdot y'} Cov\left(K_{ixy}, \sum_{x'\in\mathcal{D}_{y'}} \xi_{ix'}K_{ix'y'}\right) + \mathcal{O}(n_i^{-2}) \\
&= \sum_{i\in\mathcal{I}} \sum_{(x,y)\in\mathcal{D}} \xi_{ix}B_{ixy} \sum_{y'\in\mathcal{D}_x} B_{i\cdot y'} Cov(K_{ixy}, K_{ixy'}) + \mathcal{O}(n_i^{-2}) \\
&= \sum_{i\in\mathcal{I}} n_i \sum_{(x,y)\in\mathcal{D}} \lambda_i[x]D_i[y|x]\xi_{ix}B_{ixy}\left(B_{i\cdot y} - \sum_{y'\in\mathcal{D}_x} B_{i\cdot y'}D_i[y'|x]\right) + \mathcal{O}(n_i^{-2})
\end{aligned}
$$

Therefore the variance of the mutual information is as follows:

$$
\begin{aligned}
& V(\hat{I}(X;Y)) \\
=& V\Big(H(X) + \hat{H}(Y) - \hat{H}(X,Y)\Big) \\
=& V(\hat{H}(Y)) + V(\hat{H}(X,Y)) - 2Cov(\hat{H}(Y), \hat{H}(X,Y)) \\
=& \sum_{i\in\mathcal{I}} n_i \sum_{(x,y)\in\mathcal{D}} \lambda_i[x] D_i[y|x]((B_{ixy} - \xi_{ix}B_{i\cdot y})^2 \\
& \qquad + \sum_{y'\in\mathcal{D}_x} D_i[y'|x](2\xi_{ix}B_{ixy}B_{i\cdot y'} - \xi_{ix}^2 B_{i\cdot y}B_{i\cdot y'} - B_{ixy}B_{ixy'})) \\
& \qquad + \mathcal{O}(n_i^{-2}) \\
=& \sum_{\substack{i\in\mathcal{I} \\ x\in\mathcal{X}^+}} n_i\lambda_i[x](\sum_{y\in\mathcal{D}_x} D_i[y|x](B_{ixy} - \xi_{ix}B_{i\cdot y})^2 \\
& \qquad - \sum_{y,y'\in\mathcal{D}_x} D_i[y|x]D_i[y'|x](B_{ixy} - \xi_{ix}B_{i\cdot y})(B_{ixy'} - \xi_{ix}B_{i\cdot y'})) + \mathcal{O}(n_i^{-2}) \\
=& \sum_{\substack{i\in\mathcal{I} \\ x\in\mathcal{X}^+}} n_i\lambda_i[x] \left(\sum_{y\in\mathcal{D}_x} D_i[y|x](B_{ixy} - \xi_{ix}B_{i\cdot y})^2 - \left(\sum_{y\in\mathcal{D}_x} D_i[y|x](B_{ixy} - \xi_{ix}B_{i\cdot y}) \right)^2 \right) + \mathcal{O}(n_i^{-2}) \\
=& \sum_{\substack{i\in\mathcal{I} \\ x\in\mathcal{X}^+}} \frac{\pi_i[x]^2}{n_i\lambda_i[x]} \left(\sum_{y\in\mathcal{D}_x} D_i[y|x] \left(\log\frac{P[x,y]}{P[y]} \right)^2 - \left(\sum_{y\in\mathcal{D}_x} D_i[y|x] \log\frac{P[x,y]}{P[y]} \right)^2 \right) + \mathcal{O}(n_i^{-2})
\end{aligned}
$$

where $P[y] = \sum_{x'\in\mathcal{X}'} P[x',y]$. □

B Channel Matrix

In Fig. 6 we present the channel matrix used for the experiments for Fig 4 in Section 4.

C Details of Protocol Downgrade Attack Case Study

C.1 Code Model for Protocol Downgrade Attack

In Fig. 7 we present the full code model for the protocol downgrade attack case study presented in Section 5. We use a Java-like language including a `Uniform(S)` operator returning a uniform distribution on a set of values `S`.

The secret is a randomly generated array of 6 bits, represented as boolean variables `secret[i]`.

The `mode` variable defined in line 13 determines the algorithm used to generate the key for the one-time pad, and is initialized to 0. Lines 14 to 18 model the downgrade attack, having a 10% probability of copying the first two bits of the secret on the `mode` variable.

Then the key is generated according to the `mode` variable as described in Section 5, and the ciphertext is encoded and printed.

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. 6: Channel matrix for the experiments in Section 4.

C.2 Setup for Experiements

Experiments are run on a system running Ubuntu Linux 15.04 on a 2.0 GHz (i7-4750HQ) quad-core Intel Core i7 Haswell processor with 16 GB of RAM.

LeakWatch is invoked with the command

```
java -jar leakwatch-0.5.jar -th 4 -n [SIMS] ModalOneTimePad
```

where the parameter `-th 4` is used to take advantage of the multicore architecture. This reduces the computation time but not the number of simulations, that is enforced by the `-n [SIMS]` parameter.

For the compositional statistical approach, we have computed the precise results for the trace analysis and divided equally the simulations on the two components to be analyzed statistically, and we have combined the results and computed the confidence intervals with the technique described in Section 3.

```
import java.util.Random;

public class ModalOneTimePad {
    static int bitsize = 6;

    public static void main (String[] args){
        boolean[] secret = new boolean[bitsize];
        for (int i=0; i<bitsize; i++){
            secret [i]= Uniform(true,false)          }
        int mode = 0;
    if (goodrnd.nextDouble()<0.1){
            mode = Uniform(0...3);
        }
        boolean[] encoded = new boolean[secret.length];
        boolean[] key = new boolean[secret.length];
        switch (mode){
            case 0 : key = generateGoodKey();
                break;

            case 1 :
                break;

            case 2 : key = generateBadKey();
                break;

            case 3: key = generateModuloKey();
                break;
        }
        for (int i=0; i<bitsize; i++){
            encoded [i]= secret[i] ^ key[i];
        }
        System.out.println(valueOf(encoded));
    }

  private static boolean[] generateGoodKey() {
        boolean[] key = new boolean[bitsize];
        for (int i=0; i<bitsize; i++){
            key [i]=Uniform(true,false);
        }
        return key;
    }

    private static boolean[] generateBadKey() {
        Random badrnd = new Random();
        boolean[] key = new boolean[bitsize];
        for (int i=0; i<bitsize; i++){
            key [i]= badrnd.nextBoolean();
        }
        return key;
    }

    private static boolean[] generateModuloKey() {
        boolean[] key = new boolean[bitsize];
        int rand[] = new int[bitsize+1]; //rand[0] is the seed
        rand[0] = Uniform(0...63);
        for (int i=0 ; i <bitsize; i++) {
            rand[i+1] = ((rand[i] * 4801) + 83)%64;
            key[i] = (rand[1]) > 31;
        }
        return key;
    }
}
```

Fig. 7: Code model for the protocol downgrade attack case study