650.060 - Small Project in Artificial Intelligence and Cybersecurity

An API for Mutual Information estimation

Francesco Bombassei De Bona Summer Term 2022/2023 12138677

1 Introduction

This project aims to provide a publicly available C++ API for the methodologies introduced in [3] for estimating Mutual Information. A section is dedicated to compare the proposed implementation with the histogram estimator method.

2 Background and terminology

2.1 Mutual Information

Mutual Information (MI) is a measure of the statistical dependence between two random variables. In the context of side-channel analysis, MI is used to quantify the dependence between an observed leakage and a secret key. Higher values of MI indicate a stronger dependence between the two variables. Thus, an higher MI value leads to a higher leakage of information about the secret key by observing a side-channel trace.

The MI between two random variables X and Y is defined as:

$$I(X;Y) = H(X) - H(X|Y) \tag{1}$$
$$= H(X) + H(Y) - H(X,Y) \tag{2}$$

where $H(X)$ is the entropy of X, $H(X|Y)$ is the conditional entropy of X given Y and is defined as

$$H(X|Y) = \begin{cases} \sum_y p(y)H(X|Y=y) & \text{if } Y \text{ is discrete} \\ \int_y p(y)H(X|Y=y)dy & \text{if } Y \text{ is continuous} \end{cases}$$

and $H(X,Y)$ is the joint entropy of X and Y.

2.2 Histogram estimator

The histogram estimator is a method for estimating the MI between two random variables X and Y. The core idea behind the histogram estimator is to discretize the continuous random variables into bins and then compute the mutual information based on the joint and marginal frequencies of these bins. The MI between X and Y is then estimated as:

$$I(X;Y) = H(Y) - H(Y|X) \tag{3}$$

where $H(Y)$ is the entropy of Y and $H(Y|X)$ is the conditional entropy of Y given X.

2.3 GKOV estimator

The GKOV estimator is a method for estimating the MI between two random variables X and Y introduced by Gao, Krishnan, Oh and Vishwanath [2]. This estimator can be used on a combination of discrete and continuous random variables. The GKOV estimator is defined as:

$$I_n(X;Y) = \frac{1}{n}\sum_{i=1}^{n}\hat{I}_i = \sum_{i=1}^{n}\left(\psi(\tilde{t}_i) + \log n - \log(n_{x,i}+1) - \log(n_{y,i}+1)\right) \tag{4}$$

where ψ is the digamma function.

3 Project Execution

The project aimed to develop a C++ API for the GKOV estimator, the histogram estimator, and a simulator for the leakage of a device. All the code was written in C++ and using an OO approach.

3.1 GKOV estimator

The GKOV estimator was implemented as described in [2]. The implementation is based on the class `GKOVEstimator` which is initialized with the callback function to compute the value t_n as described in [3]. The class `GKOVEstimator` provides the method `estimate` which takes as input the discrete variable X, the continuous variable Y, and the dimension of the two variables. The method `estimate` returns the estimated MI between X and Y. Under the hood, the method `estimate` computes the value t, builds the matrix data starting from the discrete and continuous variables, builds the search trees for the two single variables and the combined matrix, and finally computes the GKOV estimator as described in [3].

3.2 Histogram estimator

The histogram estimator was implemented to be used as a baseline for comparing the GKOV estimator. Following the description in Section 2.2, the histogram estimator was implemented as a class `HistogramEstimator`. The class `HistogramEstimator` is initialized with the number of dimensions of the continuous variable Y, the number of bins for each dimension, and the ranges of the bins. The underlying assumption about the ranges is that the bins are equally spaced. The class `HistogramEstimator` provides the method `estimate` which takes as input the discrete variable X, its probability distribution, the continuous variable Y, the size of the two variables and the dimension of the Y variable.

The method `estimate` makes usage of the methods `build_histogram`, `pdf_entropy`, and `conditional_entropy` to compute the MI between X and Y. `build_histogram` builds histogram of Y based on the dimensions of the variable and computes the pdf of the

histogram. To ensure the efficiency of the histogram estimator, the method makes usage of the GSL library [1] for building the 1D and 2D histograms. The method `pdf_entropy` computes the entropy of the pdf of Y. The method `conditional_entropy` computes the conditional entropy of Y given X. Finally, the method `estimate` returns the MI between X and Y computed as described in equation 3.

3.3 Simulator

The simulator was implemented to provide a way to test the GKOV estimator and the histogram estimator. This class has the purpose of simulating the leakage of a device using the following model:

$$\text{leakage} = \text{leakage\_function}(\text{crypto\_function}(key, \text{plaintext})) + \text{noise} \tag{5}$$

where `leakage_function` is the function that simulates the leakage of the device, `crypto_function` is the function that simulates the cryptographic function of the device, and `noise` is the noise added to the leakage. At time of writing, the simulator supports gaussian and laplacian noise distributions but the code was made to be easily extensible to other distributions. Instead, the cryptographic function is passed as argument of the methods of the simulator.

Regarding the type of trace that can be simulated, the simulator supports one dimensional traces.

Finally, the simulator saves the simulated traces in a HDF5 file to allow the user to use the traces for further analysis.

3.4 Utilities

The project provides a class that implements some utilities used by the other classes in the project.

In particular, the class `Utils` provides the methods:

- `flatten` and `to_gkov_format` to convert an n-dimensional array to, respectively, a 1-dimensional array and a 2-dimensional array;
- `compute_distribution` to compute the distribution of a discrete variable;
- `read_traces` and `write_traces` to read and write traces from and to a HDF5 file.

References

[1] Galassi, M., Ed. *GNU scientific library reference manual: for GSL version 1.12*, 3. ed ed. Network Theory, Bristol, 2009.

[2] Gao, W., Kannan, S., Oh, S., and Viswanath, P. Estimating mutual information for discrete-continuous mixtures, 2017.

[3] Roy, A., Chowdhury, A., and Oswald, E. Leakage Certification based on Consistent MI Estimation, 2022.