IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2024, No. 1, pp. 433–456. DOI:10.46586/tches.v2024.i1.433-456

Quantile: Quantifying Information Leakage

Vedad Hadžić[1], Gaëtan Cassiers[1], Robert Primas[2], Stefan Mangard[1] and Roderick Bloem[1]

[1] Graz University of Technology, Graz, Austria, first.last@iaik.tugraz.at
[2] Intel Labs, Hillsboro, USA, first.last@intel.com

Abstract. The masking countermeasure is very effective against side-channel attacks such as differential power analysis. However, the design of masked circuits is a challenging problem since one has to ensure security while minimizing performance overheads. The security of masking is often studied in the t-probing model, and multiple formal verification tools can verify this notion. However, these tools generally cannot verify large masked computations due to computational complexity.
We introduce a new verification tool named Quantile, which performs randomized simulations of the masked circuit in order to bound the mutual information between the leakage and the secret variables. Our approach ensures good scalability with the circuit size and results in proven statistical security bounds. Further, our bounds are quantitative and, therefore, more nuanced than t-probing security claims: by bounding the amount of information contained in the lower-order leakage, Quantile can evaluate the security provided by masking even when they are not 1-probing secure, i.e., when they are classically considered as insecure. As an example, we apply Quantile to masked circuits of Prince and AES, where randomness is aggressively reused.

Keywords: Side-channel attacks · Masking · Verification

1 Introduction

Since the rise of the Internet of Things (IoT), embedded devices have been integrated into a wide range of everyday services, making the protection of cryptographic keys on these devices an essential but challenging task. Physical side-channel attacks, such as power or electromagnetic analysis, may allow attackers to extract cryptographic keys by observing a device's power consumption or electromagnetic emission during cryptographic operations [KJJ99, QS01, CRR02].

One of the most prominent algorithmic countermeasures against physical side-channel attacks is masking. In a nutshell, masking is a secret-sharing technique that splits inputs, outputs, and intermediate values of cryptographic computations into $t+1$ random shares such that the observation of up to t shares does not reveal any information about their corresponding unmasked value [ISW03, GMK16, BBD+16, CS19].

Probing model. The security of masking is most often studied in the threshold probing model [ISW03]. In this model, computations are represented as arithmetic circuits, and an adversary may observe the value of up to t wires in the circuit. The implementation is then considered secure if any such observation is independent of unmasked values. While this model may seem simplistic and abstract, it has been shown that any t-probing secure circuit is also secure in the much more realistic noisy leakage model [PR13]. Furthermore, security in the t-probing model implies practical t-order security under some assumption of leakage independence for each value [BDF+17]. Together, these observations lead to the conclusion that, both in (not very tight) theoretical reductions and in practice, t-probing security implies security for concrete physical side-channel leakage.

Received: 2023-07-15 Accepted: 2023-09-15 Published: 2023-12-04

Despite the simplicity of the t-probing model, it is not always easy to prove the security of masked implementations in it, and multiple approaches have been proposed in the literature. The first approach, which works well for small circuits that implement a simple functionality such as a logic or arithmetic gate (named gadgets) and are well-structured, is to write proofs by hand. In order to extend these proofs to larger implementations, composable security definitions have been proposed that enable simple security proofs for the composition of multiple gadgets. Examples of such definitions include Strong Non-Interference (SNI) [BBD+16] and Probe-Isolation Non-Interference (PINI) [CS19]. The main appeals of this approach are its scalability to complex computations and the ease of security verification (which can be automated [CGLS21, CS21]), which in turn enables automatic masked circuit generation [BDM+20, KMMS22]. On the other hand, composition-based approaches often lead to less efficient circuits than non-composable constructions since they impose additional requirements on the gadgets and hinder cross-gadget optimizations, such as randomness reuse.

Another approach is to automate security proofs for masked circuits independently of a concrete masking scheme. Multiple formal verification tools [ANR18, BBC+19, KSM20, GHP+21] have been introduced for this purpose, and they all have the same high-level functionality: given a masked circuit description, verify that it is t-probing secure. Despite numerous optimizations, the application of such tools is typically limited to the verification of no more than a few rounds of a masked circuit and low masking orders. Some of these tools achieve their efficiency at the expense of having false-positive verification results (i.e., secure circuits for which leakage is reported). The PROLEAD verification tool [MM22] recently introduced an alternative approach to verification. Rather than formally proving independence, it is based on Monte Carlo sampling and statistical tests of independence. The statistical nature of the tool significantly improves its scalability towards more complex circuits with high logic depth but admits false positives caused only by the statistical test (the probability of false positives is therefore controllable with the parameters of the test). However, this technique introduces the risk of false negatives (*i.e.*, insecure circuits wrongly reported as secure), whose probability of occurring is harder to control due to the use of asymptotical statistical tests.

Practical security. Actual side-channel adversaries do not have access to probing leakage from some of the variables in a circuit but rather a noisy leakage on each of the variables. Further, such adversaries succeed if they recover a fixed amount of information (e.g., a key) by measuring multiple traces (*i.e.*, executions of the circuit), the security level being the number of traces needed. This contrasts with the t-probing adversary, which has to recover any (no matter how tiny) amount of information in one trace.

For example, let (x_0, x_1) be a masking of the secret x such that $x_0 \oplus x_1 = x$ and $\mathbf{Pr}[x_0 = 0 | x = 0] = \mathbf{Pr}[x_0 = 0 | x = 1] = 0.5$. Furthermore, let $(l_0, l_1) = (x_0 + n, x_1 + n')$ be corresponding physical leakage, where n and n' are independent Gaussian noise variables. In this case, the circuit is 1-probing secure, and therefore, an adversary observing only l_0 or l_1 does not learn anything about x. However, the circuit is not 2-probing secure: probing both x_0 and x_1 reveals x. With the noisy leakage (l_0, l_1), a second-order adversary may estimate the covariance of l_0 and l_1 using multiple leakage traces, yielding a good distinguisher for the value of x. In contrast, assume now that, for some small $\epsilon > 0$, $\mathbf{Pr}[x_0 = 0 | x = 0] = 0.5 + \epsilon$ and $\mathbf{Pr}[x_0 = 0 | x = 1] = 0.5 - \epsilon$. The circuit is no longer 1-probing secure: observing x_0 leaks information about the value of x. Practically, this bias can be exploited by a first-order adversary observing only l_0, but an exploit requires many traces (on the order of $1/\epsilon$). However, for the second-order attack using (l_0, l_1), a small bias does not have a significant impact. As a result, the practical first-order attack may require more traces than the second-order attack to be successful.

The previous example shows that the probing security order is not always a good predictor of the security level. Therefore, it might be better to adopt a metric that quantifies the leaked information more accurately.

Contributions. This paper presents a methodology to quantitatively assess the security of masked circuits. The methodology relies on the noisy leakage model [PR13], *i.e.*, we assume that the leakage is made of independent noisy leakages of each of the variables of the circuit. Then, we bound the mutual information between these variables and the secret using a statistical sampling-based technique implemented in an open-source tool named Quantile (QUANTifier of Information LEakage) and available at

https://github.com/vedadux/quantile .

Compared to the state-of-the-art t-probing security verification tools, our method scales well to large circuits and provides proven statistical bounds. Furthermore, beyond t-probing security, our methodology is also able to evaluate the security level in the noisy leakage model, taking into account the presence of noise in practical leakages.

Our contributions can be summarized as follows:

- We present a scalable sampling-based verification technique for masked circuits that bounds the mutual information between sensitive values and t-probing model leakage.
- We show how to turn these bounds into a lower bound on the number of attack traces needed for a worst-case adversary, allowing us to compare attacks at various orders. To the best of our knowledge, our technique is the first one to use a circuit's description to formally quantify "benign" masking imperfections, thanks to the notion of effective security order.
- We provide an optimized software implementation of our verification tool that exploits vector instructions and multi-core processors.
- We show the effectiveness of our verification approach by applying it to masked implementations of AES and Prince using different amounts of randomness reduction/reuse techniques to meet certain performance or efficiency goals in lightweight applications.

The rest of the paper has the following structure. Section 2 covers the necessary preliminaries for our masking verification technique. In Section 3, we develop a method for bounding the number of attack traces needed to carry out a side-channel attack. This method is based on bounding the mutual information between a secret and physical leakage. Section 4 explains how approximations of the mutual information can be computed efficiently and what the overall workflow of Quantile looks like. In Section 5 we apply Quantile to masked implementations of AES and Prince using different randomness reduction/reuse techniques. Section 6 discusses related work, and finally, we conclude the paper in Section 7. Important proofs for our methodology are given in Appendix A and Appendix B.

2 Preliminaries

In the following, we briefly introduce the side-channel setting modeled through information channels and give an overview of entropy, mutual information, and basic estimators.

Notation. Throughout this work, we denote random variables with uppercase letters (X), their values with lowercase letters (x), and sets with calligraphic letters ($\mathcal{X}$). We use bold uppercase letters ($\boldsymbol{X}$) for vectors of random variables. We write $X \sim \mathcal{U}_{\mathcal{X}}$ when the random variable X follows a uniform distribution over the set $\mathcal{X}$. Similarly, we write $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$ for a random variable X that follows the normal distribution with mean μ and variance σ^2.

2.1 Side-Channel Leakage

We first formalize side-channel leakage of a cryptographic computation from an information-theoretic point of view. A cryptographic computation takes as input a secret S (e.g., a cryptographic key) and known data D (e.g., the plaintext) from the respective domains $\mathcal{S}$ and $\mathcal{D}$. In general, we assume that the computation is also masked, meaning it takes additional uniformly random inputs $M \sim \mathcal{U}_{\mathcal{M}}$ (e.g., masks), ultimately randomizing the computation even when the same secret and data are provided. We write X to denote a tuple of intermediate values of the computation.

A computation is said to be t-probing secure if all tuples X of size t are independent of S. Here, we assume that S is provided to the computation in an already masked representation and cannot be probed directly [ISW03]. Furthermore, we say that a t-probing secure computation has the security order t.

However, in reality, a side-channel attacker cannot observe X directly. Instead they observe physical leakage, which we model as the tuple L, where each element is the result of applying an independent noisy function [PR13] to the corresponding element of X.

In a side-channel attack, the secret is chosen uniformly at random, *i.e.*, $S \sim \mathcal{U}_{\mathcal{S}}$. The adversary then performs n_{Adv} computations with the same secret S but changing data $\boldsymbol{D} = (D_i)_{i=1}^{n_{\mathsf{Adv}}}$ and fresh (independent) masks $\boldsymbol{M} = (M_i)_{i=1}^{n_{\mathsf{Adv}}}$ with $M_i \sim \mathcal{U}_{\mathcal{M}}$. In the rest of this work, we additionally assume that each D_i is chosen independently and uniformly at random, *i.e.*, $D_i \sim \mathcal{U}_{\mathcal{D}}$. Each of the computations produces noisy leakage L_i, thus giving the adversary access to $\boldsymbol{L} = (L_i)_{i=1}^{n_{\mathsf{Adv}}}$. Finally, the adversary outputs a guess S' for the secret S. The success rate r_{Adv} of the attack is defined as $r_{\mathsf{Adv}} = \mathbf{Pr}\left[S = S'\right]$, while the attack order is the number of elements in L. The following Markov chain summarizes the attack process:

$$(S, \boldsymbol{D}, \boldsymbol{M}) \to (\boldsymbol{X}, \boldsymbol{D}) \to (\boldsymbol{L}, \boldsymbol{D}) \to S' \tag{1}$$

In this work, we will use the the Hamming weight of the probed variable with additive Gaussian noise as an example for the noisy leakage function in L. For such function, we can define the signal-to-noise ratio (SNR) as the variance of the deterministic part (the Hamming weight) divided by the variance of the noise [Man04].

2.2 Entropy Estimation

For a random variable X, its (Shannon) entropy represents the uncertainty in its outcome, represented as bits. For a discrete variable X, e.g., occurring in a digital computation, entropy $\mathrm{H}(X)$ is defined as

$$\mathrm{H}(X) = -\sum_{\substack{x \in \mathcal{X}, \\ \mathbf{Pr}[X=x] \neq 0}} \mathbf{Pr}\left[X = x\right] \log_2\left(\mathbf{Pr}\left[X = x\right]\right). \tag{2}$$

The above definition extends to the entropy $\mathrm{H}(X|Y=y)$ of $X|Y=y$, and the conditional entropy, as

$$\mathrm{H}(X|Y) = \sum_{y \in \mathcal{Y}} \mathbf{Pr}\left[Y = y\right] \mathrm{H}(X|Y=y). \tag{3}$$

Because of its frequent occurrence, we define the function $\mathrm{H}_{\mathrm{bin}} : (0,1) \to (0,1)$ as

$$\mathrm{H}_{\mathrm{bin}}(p) = -p\log_2(p) - (1-p)\log_2(1-p).$$

$\mathrm{H}_{\mathrm{bin}}$ is often referred to as the *binary entropy function* because it represents the the entropy of a variable X with domain $\mathcal{X} = \{x_0, x_1\}$ and probability $p = \mathbf{Pr}\left[X = x_0\right]$.

The mutual information between random variables X and Y is defined as $\mathrm{I}(X;Y) = \mathrm{H}(X) - \mathrm{H}(X|Y)$, whereas the mutual information conditioned on Z is defined similarly as

$\mathrm{I}(X;Y|Z) = \mathrm{H}(X|Z) - \mathrm{H}(X|Y,Z)$. The bounds for the mutual information presented in this paper rely on a very simple estimator for the entropy of a distribution. The so-called plug-in estimator first estimates the distribution of a random variable X through samples and computes (2) using this new distribution.

Definition 1 (Plug-in entropy estimator)**.** Given a vector $\boldsymbol{X}_n = (X_i)_{i=1}^n$ of independent and identically distributed random variables X_i, let $\widehat{X_n}$ be a new random variable distributed according to

$$\mathbf{Pr}\left[\widehat{X}_n = x\right] = \sum_{i=1}^{n} \frac{1}{n} \mathbb{1}_{\{x\}}(X_i),$$

where $\mathbb{1}_{\mathcal{A}}(x)$ is an *indicator function* with value 1 if and only if $x \in \mathcal{A}$, and 0 otherwise. The plug-in entropy estimator $\widehat{\mathrm{H}}_n(X)$ is defined as the entropy $\mathrm{H}\left(\widehat{X_n}\right)$ shown in (2).

This estimator is negatively biased everywhere, as shown by Paninski [Pan03].

Proposition 1 (Bias of the plug-in entropy estimator [Pan03, Prop. 1])**.** *For a discrete random variable X with support $\mathcal{X}$, the bias of the entropy estimator $\widehat{\mathrm{H}}_n(X)$ satisfies*

$$-\log_2\left(1 + \frac{|\mathcal{X}| - 1}{n}\right) \leq \mathbf{E}\left[\widehat{\mathrm{H}}_n(X)\right] - \mathrm{H}(X) \leq 0.$$

3 Bounding the Mutual Information

In this section, we develop a method for bounding the number of attack traces needed for a side-channel attack. This method is based on bounding the mutual information $\mathrm{I}(L;S|D)$. As a first step, we bound the noiseless mutual information $\mathrm{I}(X;S|D)$ in Section 3.1. Afterward, in Section 3.2, we show how such a bound can be integrated with knowledge of the noisy leakage function (e.g., knowledge of the SNR) to get a bound on $\mathrm{I}(L;S|D)$. Finally, we show how the latter can be mapped to an attack's success rate.

3.1 Information Leakage from an Intermediate Variable

In general, estimating the mutual information between two random variables is a difficult problem for continuous variables or discrete variables with a large domain size. Moreover, deriving good bounds is also difficult, since the estimators for mutual information can be biased either positively or negatively, depending on the distribution. In our case, even though S has a large domain, we know that it is uniform (and similarly for D), and X is a discrete variable with a relatively small domain. We exploit this knowledge to derive practically-relevant bounds on $\mathrm{I}(X;S|D)$. More precisely, we design a method to derive the bounds

$$\mathrm{I}_{\mathrm{LB}}(X;S|D) \leq \mathrm{I}(X;S|D) \leq \mathrm{I}_{\mathrm{UB}}(X;S|D)$$

such that each inequality holds with probability at least $1-\delta$, where δ is a confidence level, that is, the probability that the bound is incorrect. Informally, the core idea of the method is to use the equality $\mathrm{I}(X;S|D) = \mathrm{H}(X|D) - \mathrm{H}(X|S,D)$, and approximate both $\mathrm{H}(X|D)$ and $\mathrm{H}(X|S,D)$ independently. Here, we perform the approximations by summing only over n_D many values for D, respectively (S,D) in equation (3), and estimating both $\mathrm{H}(X|D=d)$ and $\mathrm{H}(X|S=s,D=d)$ using the plug-in entropy estimator with n_X samples of $X|D=d$, respectively $X|S=s,D=d$. This procedure gives us an estimator $\widehat{\mathrm{I}}(X;S|D)$ for the conditional mutual information $\mathrm{I}(X;S|D)$, and we can then analyze its bias and its variance to get the bounds $\mathrm{I}_{\mathrm{LB}}(X;S|D)$ and $\mathrm{I}_{\mathrm{UB}}(X;S|D)$.

The bias of this estimator comes from the bias of the plug-in estimators for the entropy (which are bounded in [Pan03]). Regarding the deviation, the asymptotic trend follows the central limit theorem. We derive nonasymptotic bounds in Appendix A, giving the following result as a corollary.

Corollary 1. *Let X, D and S be discrete random variables with domains $\mathcal{X}$, $\mathcal{D}$ and $\mathcal{S}$, and let D and S be distributed uniformly. Let n_D and n_X be positive integers and $\delta > 0$ be a real number. Let $\boldsymbol{D} = (D_i)_{i=1}^{n_D}$, $\boldsymbol{D'} = (D'_i)_{i=1}^{n_D}$, $\boldsymbol{S} = (S_i)_{i=1}^{n_D}$ be vectors of independent random variables distributed identically to D and S respectively. Furthermore, let $\boldsymbol{X}_i = (X_{i,j})_{j=1}^{n_X}$ and $\boldsymbol{X}'_i = \left(X'_{i,j}\right)_{j=1}^{n_X}$ be vectors of independent random variables distributed identically to $X|D = D_i$ and $X|S = S_i, D = D'_i$ respectively. Finally, let*

$$\widehat{\mathrm{H}}\left(X|D\right) = \frac{1}{n_D}\sum_{i=1}^{n_D}\widehat{\mathrm{H}}_{n_X}\left(X|D = D_i\right), \quad \widehat{\mathrm{H}}\left(X|S,D\right) = \frac{1}{n_D}\sum_{i=1}^{n_D}\widehat{\mathrm{H}}_{n_X}\left(X|S = S_i, D = D'_i\right)$$

be estimates for $\mathrm{H}\left(X|D\right)$ and $\mathrm{H}\left(X|D,S\right)$, with $\widehat{\mathrm{I}}\left(X;S|D\right) = \widehat{\mathrm{H}}\left(X|D\right) - \widehat{\mathrm{H}}\left(X|S,D\right)$ consequently being an estimate for $\mathrm{I}\left(X;S|D\right)$. Then $\mathrm{I}_{LB}\left(X;S|D\right) = \widehat{\mathrm{I}}\left(X;S|D\right) - \varepsilon$ and $\mathrm{I}_{UB}\left(X;S|D\right) = \widehat{\mathrm{I}}\left(X;S|D\right) + \varepsilon$, with

$$\varepsilon = \log_2\left(1 + \frac{|\mathcal{X}| - 1}{n_X}\right) + \sqrt{\log\left(\delta^{-1}\right) n_D^{-1}\left(\log_2^2|\mathcal{X}| + n_X \mathrm{H}_{bin}\left({n_X}^{-1}\right)^2\right)}$$

satisfy

$$\mathbf{Pr}\left[\mathrm{I}_{LB}\left(X;S|D\right) < \mathrm{I}\left(X;S|D\right)\right] > 1 - \delta \quad \textit{and} \quad \mathbf{Pr}\left[\mathrm{I}_{UB}\left(X;S|D\right) > \mathrm{I}\left(X;S|D\right)\right] > 1 - \delta.$$

Here, Corollary 1 gives bounds for $\mathrm{I}\left(X;S|D\right)$ of a single intermediate variable X with confidence $1 - \delta$. While it is tempting to then find the point $X_{\max}$ with maximal $\widehat{\mathrm{I}}\left(X_{\max};S|D\right)$ throughout the whole computation and claim that for all other intermediate computations X the mutual information is less than $\mathrm{I}_{\mathrm{UB}}\left(X_{\max};S|D\right)$, this ignores the fact that given long enough computations there is a good chance that some intermediate values violate their upper bounds. To account for this, we can apply a union-bound over all intermediate values, essentially dividing δ by the length of the computation. Both the derivation and proof are given in Appendix A.

3.2 From Probing to Noisy Leakage

Let us now assume that instead of observing directly the leaking variable X, we observe as leakage $L = f(X)$ for some noisy function $f : \mathcal{X} \to \mathcal{L}$ [DDF19]. We are therefore interested in bounding $\mathrm{I}\left(L;S\right)$, which then gives us a bound on the number of traces n_{Adv} an adversary needs to measure, in order to recover the value of S with some probability.

For $0 \leq p \leq 1$, we denote $\mathsf{id}_p : \mathcal{X} \to \mathcal{X} \cup \{\perp\}$ the randomized function that on input $x \in \mathcal{X}$ outputs x with probability p and $\perp$ otherwise. If there exists a (randomized) function $f^{\perp} : \mathcal{X} \cup \{\perp\} \to \mathcal{L}$ such that for all $x \in \mathcal{X}$ such that $f(x)$ has the same distribution as $f^{\perp}\left(\mathsf{id}_p\left(x\right)\right)$, then, by the data processing inequality,

$$\mathrm{I}\left(L;S|D\right) = \mathrm{I}\left(f^{\perp}\left(\mathsf{id}_p\left(X\right)\right);S|D\right) \leq \mathrm{I}\left(\mathsf{id}_p\left(X\right);S|D\right) = p\,\mathrm{I}\left(X;S|D\right). \tag{4}$$

The reduction of noisy leakage functions f to random probing functions $f^{\perp}(\mathsf{id}_p(\cdot))$ has been extensively studied in the literature [DDF19, PGMP19], and its discussion is out of the scope of this work.

Example 1. As an illustration, let us consider the case of single-bit leak X with $\mathcal{X} = \mathbb{F}_2$, that is added with Gaussian noise to obtain L, *i.e.*, $L = f(X) = \mathbb{1}_{\{0\}}(X) - \mathbb{1}_{\{1\}}(X) + Z$ with $Z \sim \mathcal{N}\left(0, \sigma^2\right)$. The portion of the distribution of L where we cannot distinguish $X = 0$ from $X = 1$ corresponds to the portion of id_p that maps to $\perp$, *i.e.*, 1 - p. The two parts of the probability distribution of L "meet" at $\mathbf{Pr}[L = 0|X = 0] = \mathbf{Pr}[L = 0|X = 1]$. Using $\mathsf{cdf}_{\mathcal{P}}(\cdot)$ as the cumulative distribution function of distribution $\mathcal{P}$ and $\mathsf{erf}(\cdot)$ for the error function, we have

$$\begin{aligned} 1 - p &= \mathbf{Pr}[L > 0|X = 1] + \mathbf{Pr}[L < 0|X = 0] \\ &= 2\mathsf{cdf}_{\mathcal{N}(0,\sigma^2)}(-1) = 1 + \mathsf{erf}\left(\frac{-1}{\sigma\sqrt{2}}\right). \end{aligned}$$

Therefore, usign the first term of the Taylor series for $\mathsf{erf}(\cdot)$, we have

$$p = -\mathsf{erf}\left(\frac{-1}{\sigma\sqrt{2}}\right) = \mathsf{erf}\left(\frac{1}{\sigma\sqrt{2}}\right) \leq \sigma^{-1}\sqrt{\frac{2}{\pi}}. \tag{5}$$

The simple leakage function $L = f(X)$ is later used in Section 5.2 to contextualize the experimental results in terms of noisy leakage.

3.3 Number of Attack Traces

The number of traces needed to mount an attack is the most common side-channel security metric. In this section, we show how to bound the number of traces n_{Adv} required to mount a secret-recovery attack with success rate r_{Adv}, using the mutual information $\mathrm{I}(L; S|D)$ between the observed leakage L and the secret S.

Lemma 1. *Let $n_{\mathsf{Adv}} \geq 0$ be an integer, S be a random variable with domain $\mathcal{S}$ and $\boldsymbol{D} = (D_i)_{i=1}^{n_{\mathsf{Adv}}}$ be a vector of random variables independent of S with the same domain $\mathcal{D}$. Furthermore, let $f : \mathcal{S} \times \mathcal{D} \to \mathcal{L}$ be a randomized function modelling a memoryless channel, and let $\boldsymbol{L} = (L_i)_{i=1}^{n_{\mathsf{Adv}}}$ be a vector of random variables with $L_i = f(S, D_i)$. Let $\mathsf{Adv} : \mathcal{D}^{n_{\mathsf{Adv}}} \times \mathcal{L}^{n_{\mathsf{Adv}}} \to \mathcal{S}$ be a (potentially randomized) function attempting to recover the value of S as $S' = \mathsf{Adv}(\boldsymbol{D}, \boldsymbol{L})$ and let $r_{\mathsf{Adv}} = \mathbf{Pr}[S' = S]$. Then, assuming $\mathrm{I}(L; S|D) \neq 0$,*

$$n_{\mathsf{Adv}} \geq \frac{\mathrm{H}(S) - (1 - r_{\mathsf{Adv}})\log_2(|\mathcal{S}| - 1) - \mathrm{H}_{bin}(r_{\mathsf{Adv}})}{\mathrm{I}(L; S|D)}. \tag{6}$$

This result is a version of [dCGRP19, Theorem 1], and the proof given in Appendix B is very similar to the original proof. The major difference is that we are giving a statement in the context of $\mathrm{I}(L; S|D)$, while they use $\mathrm{I}(X; L|D)$.[1]

Example 2. We illustrate Lemma 1 for a side-channel attack on ciher with an 128-bit key, e.g., AES [DR98] or Prince [BCG+12]. Assuming that the key is chosen uniformly at random, an adversary that wants a sucess rate of at least $r_{\mathsf{Adv}} = 50\%$ would need

$$n_{\mathsf{Adv}} > \frac{128 r_{\mathsf{Adv}} - \mathrm{H}_{\mathrm{bin}}(r_{\mathsf{Adv}})}{\mathrm{I}(L; S|D)} = \frac{63}{\mathrm{I}(L; S|D)}.$$

traces. This shows that the number of traces is inversely proportional to the mutual information, with a proportionality factor that is relatively small. This factor depends on the size and distribution of the key and the success rate, but it is anyway bounded by the entropy of the key. In Section 5.2, we use this lemma and a bound on the mutual information in order to get a lower bound on the number of attack traces n_{Adv}, *i.e.*, on the security level.

[1]Further, they assume uniform D, which is not done in Lemma 1 (although we generally assume uniform D in this work).

4 Computing the Approximation and Bounds

Approximating and bounding the mutual information $\mathrm{I}(S;X|D)$, as outlined in Section 3.1, requires a significant amount of samples for the involved random variables. In this section, we first present the critical insights for highly efficient sampling and introduce our simulation framework and its workflow.

4.1 Efficient Sampling

In order to get samples for the intermediate values X, it is necessary to execute the design of the cryptographic primitive. There are many state-of-the-art hardware simulators, and it might be tempting to just pick one of them and use them to simulate the designs and obtain the samples. However, these tools are intended for testing, debugging, and accurate timing estimation and are not meant for running a large number of simulations and aggregating them into histograms. Quantile's extremely efficient aggregating simulator is based on two key insights.

Code generation from symbolic simulation. The first critical insight is that many of the values in hardware design's execution do not change across simulations. This includes control signals and everything else that is independent of the data the hardware processes, *i.e.*, public data, secrets, and masks. In a sense, such intermediate values are constants and can be optimized away by *unrolling* the hardware circuit across the clock cycles of its execution. Here, the hardware circuit starts out with its data inputs being the only *unknown* values that must be treated *symbolically.* Furthermore, any time such *symbolic* signals are fed as inputs to a hardware cell, the cell output is computed *symbolically* as well. For example, an `AND` cell, with input symbols a and b, would produce output symbol $a \wedge b$. However, if the second input is constant 0 (respectively 1), it would produce output constant $a \wedge 0 = 0$ (respectively symbol $a \wedge 1 = a$). For clock cycle transitions, the symbolic register output signals in the current clock cycle are defined to be equal to the register input signal from the previous clock cycle. As a result, the number of clock cycles to simulate can is known beforehand since the end of the execution is triggered by a non-symbolic `finish` signal. After symbolically simulating a hardware circuit, we have effectively generated a straight-line symbolic trace of its execution, where all constants have been eliminated, and only the important computations are left. This trace can be turned into a very efficient statically compiled simulator.

Parallel simulations through bitslicing. The second critical insight is that the netlist, and therefore the execution trace, exclusively manipulates single-bit variables. We can therefore use the bitslicing technique to store values belonging to separate simulations inside a single architectural register. Moreover, many modern `x86_64` machines support the `SSE2`, `AVX2`, and `AVX512F` extensions that provide 128-, 256-, and 512-bit wide registers and vectorized instructions for all common bitwise logic operations. The only missing operation commonly used in netlists is a multiplexer, which can be simulated as $\mathrm{MUX}(s, i_0, i_1) = (\neg s \wedge i_0) \oplus (s \wedge i_1)$. Furthermore, the bit-level parallelism of bitsliced executions additionally enables the quick computation of 1-bit histograms by counting the number of bits set to 1 inside a register with the `popcntq` inctruction. This can be exploited because currently, Quantile is tailored to univariate leakage analysis. For support of multivariate leakage, the histogram computation needs to be adapted, either to do bit-manipulations and still use `popcntq`, or to unslice the parallel executions and count the element frequencies in a more traditional manner.

4.2 Framework Overview

We now briefly present the workflow of Quantile shown in Figure 1, with the main steps described below.

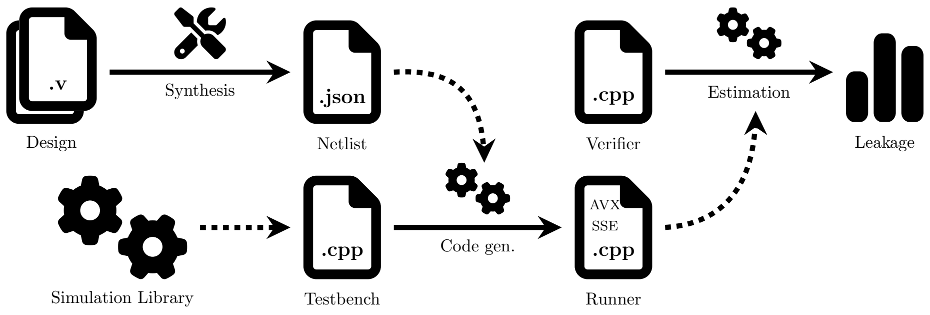

Figure 1: Workflow of the information leakage quantification framework.

Step 1 (Synthesis). The user provides a hardware design written in either Verilog, VHDL, or a mix of both, which is compiled into a JSON netlist using Yosys (with the GHDL plugin). The generated netlist is implemented using Yosys' generic gate library, for which we have written a custom symbolic simulation library.

Step 2 (Simulation). The user writes a testbench in C++ using the provided simulation library, setting the values of input signals and registers to constants or declaring them as *secret*, *data*, or *masks*. The simulation library handles the symbolic representation and simplification of intermediate values throughout the testbench execution.

Step 3 (Code Generation). The testbench symbolically unrolls the execution of the netlist unsing the provided constants, *secrets*, *data* and *masks*, generating a straight-line C++ program representing an execution of the netlist under the control of the testbench. The generated code is able to perform a bitsliced execution of the design, and chooses the optimal width for bitslicing, depending on whether `SSE2`, `AVX2` or `AVX512F` are available on the given machine.

Step 4 (Estimation). Finally, the generated code is used to sample S, D and X. After choosing the target confidence δ, the user either chooses the sampling quantities n_D and n_X directly, or lets the framework pick its best guess for some target error ε. The framework instantiates multiple simulation workers and pools their results to continuously update its mutual information estimate and report it to the user.

5 Analysis of Masked Ciphers

In this section, we apply our method to derive bounds for the information leakage of several cryptographic implementations. We show that QUANTILE can compute reasonable bounds for complete executions of cryptographic primitives, look at the consequences of randomness reuse, and investigate the security of low-randomness masking techniques. Then, we illustrate the bounds on the noisy leakage and number of attack traces. We discuss these results, showing why quantitative security bounds are more useful than simply evaluating the security order.

5.1 Bounds for Intermediate Variables

5.1.1 Securely Masked Ciphers

In the following, we give a brief overview of well-known masked hardware designs we have evaluated using our framework. Afterward, we discuss the results of our analysis shown in Table 1.

AES DOM [GMK16] implements a protected AES [DR98] using the domain oriented masking (DOM) scheme with two shares, where each share is assigned a domain and only cross-domain operations are re-shared using on-the-fly randomness. Since the DOM

Table 1: Summary of QUANTILE results for $\delta = 10^{-3}$ when designs use fresh masks (🎲), reuse masks after n rounds (♻n) or do not use fresh masks (🚫), with ✗ indicating $\mathrm{I}_{\mathrm{LB}}\left(X_{\max}; S|D\right) > 0$.

Design	Masks	Simulations	Error (ε)	$\widehat{\mathrm{I}}\left(X_{\max}; S\|D\right)$
AES DOM [GMK16]	🎲	5.24×10^{12}	$\pm 3.00 \times 10^{-4}$	$6{\times}10^{-6}$
	♻1			$6{\times}10^{-6}$
	🚫	7.94×10^{5}	$\pm 10^{-1}$	**0.99** ✗
Prince TI [BKN22]	🎲	5.24×10^{12}	$\pm 3.00 \times 10^{-4}$	$5{\times}10^{-6}$
	♻1			$\mathbf{3.70{\times}10^{-4}}$ ✗
	♻2			$5{\times}10^{-6}$
	🚫	7.94×10^{5}	$\pm 10^{-1}$	**0.99** ✗
Prince Nullfresh [SM21]	🚫	5.24×10^{12}	$\pm 3.00 \times 10^{-4}$	3×10^{-6}
AES 2-bit masking [GMKM18]	🚫			3×10^{-9}

scheme requires a register stage inside every AND gadget, the computation of a single S-Box requires 5 pipeline stages. The implementation uses one S-Box instance and serially applies it to all 16 bytes of the state. A round of AES therefore takes 20 clock cycles.

Prince TI [BKN22] implements a protected Prince [BCG+12] block cipher using CMS construction [RBN+15] with two shares, where re-sharing is only necessary at the end of a non-linear operation before the results are compressed back into two shares. This allows the implementation to implement an S-Box with only two stages. Since the S-Box is rather small, the design uses 16 S-Box instances applied to all 4-bit nibbles in parallel. Overall, one round of Prince only takes two clock cycles.

Results. For AES DOM and Prince TI with fresh masks, the approximated mutual information was very low ($\leq 10^{-5}$), and significantly smaller[2] than the error bounds of Corollary 1 (3×10^{-4}). This leads to the conclusion that the implementations' first-order leakage is small or nonexistent. As a sanity check, we also run the analysis on AES DOM and Prince TI without masks, *i.e.*, setting all masks to have value 0. Unsurprisingly, our analysis immediately determines mutual information of about 0.99 at many points in the computation, proving them insecure with a moderate error bound of 0.1.

5.1.2 Reusing Masks

Protecting an implementation with masking inevitably increases the size of a circuit, increases the latency due to synchronisation and (usually) requires a lot of fresh randomness. Generating enough randomness at each clock cycle generally requires bulky RNG modules that increase the overall design size. An alluring but dangerous idea for reducing the randomness requirements is to simply reuse masks. Done naively, this has the potential of undermining the probing security of the design, rendering the masking useless. However, there might be ways to cleverly reuse masks and retain practical security, if not even perfect probing security.

Results. As a simple preliminary experiment, we analyzed what happens to the security of AES DOM and Prince TI when randomness is reused across different rounds of the cipher. AES DOM did not show any signs of increased information leakage when reusing the same randomness is used for all the rounds of the cipher, yielding the same approximated mutual information. We suspect that this is due to the diffusion properties

[2]The non-tightness of the bound is expected, due to Corollary 1 relying on theorems that are not always fully tight, and due to the worst-case assumptions in the proof. In particular, Corollary 1 tolerates large discrepancies between $\mathrm{H}\left(X|D = d\right)$ for different values of d while, in our examples, $\mathrm{H}\left(X|D = d\right)$ is the same for all values of d since the plaintext is XORed to the key as the first step in the cipher.

Table 2: Bounds for effective security order transitions: above $\mathrm{SNR_t}$, the effective security order is 2. The corresponding mutual information (which is a bound for the first-order attack the exact value for the second-order attack) and number of attack traces are given.

Design	Masks	$\mathrm{SNR_t}$	$\mathrm{I_{UB}}\left(L_{\max}; S\|D\right)$	n_{Adv} (LB)
AES DOM [GMK16]	🎲 ♻1	4.89×10^{-3}	1.71×10^{-5}	3.70×10^{6}
Prince TI [BKN22]	🎲	4.87×10^{-3}	1.70×10^{-5}	3.71×10^{6}
	♻1	8.27×10^{-3}	4.86×10^{-5}	1.30×10^{6}
	♻2	4.87×10^{-3}	1.70×10^{-5}	3.71×10^{6}
Prince Nullfresh [SM21]	🚫	4.85×10^{-3}	1.68×10^{-5}	3.74×10^{6}
AES 2-bit masking [GMKM18]	🚫	4.82×10^{-3}	1.66×10^{-5}	3.79×10^{6}

of AES, as well as the large number of masks needed for the computation of each cipher round. In contrast, Prince TI becomes less secure when the same randomness is used in each round of the cipher, with approximated mutual information of 3.70×10^{-4}. According to Corollary 1, it is very likely (99.9%) that $\mathrm{I}\left(X; S|D\right) > 7.0 \times 10^{-5}$ and therefore that Prince TI is not probing secure when masks are reused in each round. We ran another experiment where randomness gets reused every two rounds, and got the same leakage estimate as for Prince TI with fresh masks.

5.1.3 Low-randomness Designs

In this section, we analyze recent low-randomness masking schemes that rely exclusively on the randomness coming from the initial input sharings.

Nullfresh [SM21] removes randomness from first-order masked computations. The Nullfresh method achieves this by noticing that the three-input computation $(a \wedge b) \oplus c$ does not require fresh randomness for a secure first-order sharing, assuming that c is uniformly distributed. Because the computation of an AND gate can be represented as $a \wedge b = (\neg a \wedge b) \oplus b$, it is also possible to create a secure first-order sharing of $a \wedge b$ analogously. Similarly, most quadratic and cubic 3- and 4-input S-boxes can be shared in a similar manner, eliminating the need for fresh randomness in ciphers that use them.

2-bit Masking [GMKM18] is a slightly older technique for achieving low-randomness implementations. The authors notice that it is possible to construct a first-order masked AND gate where the bulk of the computation happens in the first share of the output, and the second share is inherited from one of the inputs. This leads them to a technique where, given careful sharing choices, every input and intermediate value a in the original computation is shared as $(a \oplus m, m)$, where $m \in \{m_0, m_1, m_0 \oplus m_1\}$ and m_0 and m_1 are the only uniformly random values necessary for the security of the computation.

Results. We have analyzed a Nullfresh implementation of Prince and a 2-bit masking implementation of AES, and present the results in Table 1. Both of these implementations achieve stunningly low estimated conditional mutual information ($\leq 10^{-5}$), with the 2-bit masked AES achieving the lowest estimate 3×10^{-9} in our experiments. We suspect that this happens because only two masks are used overall, compared Nullfresh Prince which uses 192 masks for the initial sharing, leading to extremely low variance for the estimator $\widehat{\mathrm{H}}_{n_X}\left(X|S = s, D = d\right)$, due to there only being four different possibile values for the masks.

5.2 Noisy Leakage and Number of Traces

We now move on to the more realistic noisy leakage model, where we assume that the value of every intermediate bit in the computation leaks with additive Gaussian noise.

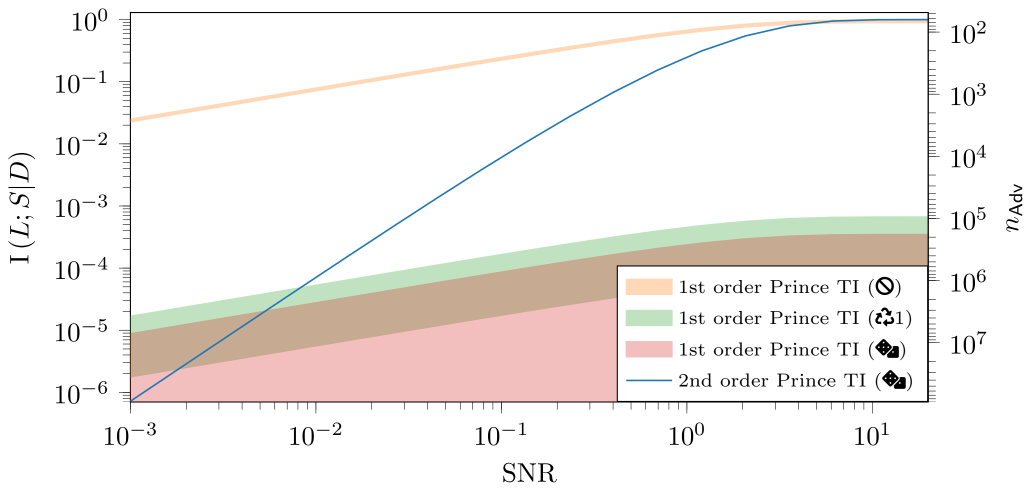

Figure 2: Noisy leakage mutual information $\mathrm{I}(L; S|D)$ of the most leaking signal in the Prince TI circuit for first- and second-order leakage, where each bit leaks independently with additive Gaussian noise according to SNR. The right-hand side axis shows the number of traces n_{Adv} needed for an attack with $r_{\mathsf{Adv}} = 50\%$ based on (6). The first-order leakage bounds are based on (4), with p taken from (5), and $\mathrm{I}(X; S|D)$ bounds taken from Table 1 (these upper and lower bounds give the interval shown in the plot).

Considering the presence of noise allows us to quantitatively compare the leakage at different orders in the bit-leakage with Gaussian noise model (i.e., each intermediate bit leaks independently with additive Gaussian noise).

For first-order leakage, we combine the bound $\mathrm{I}_{\mathrm{UB}}(X_{\max}; S|D)$ from Quantile (Table 1) with Equation 4 and Lemma 1. For second-order leakage, we assume the observation of noisy leakage of a pair of uniform shares representing a secret bit (or a bit in bijection with a key bit, given the plaintext). The mutual information $\mathrm{I}(X; S|D)$ is then easily computed with numerical integration, and the number of traces is derived with Lemma 1.

In Figure 2, we show these results for the Prince TI circuit, as a function of the SNR. Prince TI (🛇) shows that not using randomness, strongly degrades the security, making a first-order attack with few traces possible. For the Prince TI (♻1), the second-order leakage is larger than the first-order leakage when the SNR $> 8.27 \times 10^{-3}$. That is, for relatively high SNR values, a second-order attack requires fewer traces than a first-order attack. Consequently, the security of implementations with such SNR is dictated by the second-order attack, and reused randomness does not lower the security level in such cases.

This discussion relates to the notion of "effective security order" [DDF19, Sta20], which is the the order of the optimal attack (*i.e.*, the one that requires the lowest number of traces). This order can be larger than the "true" security order in the probing model where observations are noiseless. By contrast, the effective security order refers to concrete SNR values. For SNR $> 8.27 \times 10^{-3}$, Prince TI (♻1) has an effective security order of 2, while for SNR $\leq 8.27 \times 10^{-3}$, its effective security order is 1.

Finally, Table 2 shows the coordinates of the points where the first- and the second-order attacks intersect, *i.e.*, the SNR above which the effective security order is 2. Below that point, the effective security order might be 1 (as in the Prince TI with randomness re-use at every round), but might also be 2 (this is due to the overapproximation of the mutual information cf. Table 1). This table shows that for adversaries with less than 1 million attack traces, the second-order attack performs better, and therefore the first-order leakage is not an issue. Let us also note that the transition values of $\mathrm{SNR_t}$ are fairly low. Therefore,

Table 3: Single-core simulation performance of Verilator, PROLEAD and QUANTILE, applied to first-order protected designs AES DOM and Prince TI.

Design	Cells	Cycles	Simulations per second		
			Verilator	PROLEAD	QUANTILE
AES DOM [GMK16]	2510	219	5.92×10^3	4.57×10^3	4.02×10^5
Prince TI [BKN22]	6474	28	5.78×10^3	1.45×10^4	2.54×10^6

first-order leakage of these designs is only an issue in implementations with already high noise levels or a very low signal levels, e.g., due to dual-rail logic [LMW14].

5.3 Evaluation of QUANTILE's Simulator

We have evaluated the performace of QUANTILE's efficient sampling method against both Verilator and PROLEAD on AES DOM and Prince TI. The comparison only considers how quickly the simulators are able to generate full simulation traces of the given designs, and no additional statistics are gathered. The Verilator-based sampler uses a custom-written C++ testbench for the designs that iteratively runs simulations with random secrets, masks and data. As for PROLEAD, we have removed all of its analysis capabilities, probe gathering, statistics computation, and only ran the circuit simulation component. The evaluation was done on a machine equipped with an eight-core Intel Core i7-8550U CPU running at 1.8 GHz, 16 GiB of memory, and a 64-bit Linux system. The results are shown in Table 3 and indicate that our simulation technique is about two orders of magnitude faster than both Verilator and PROLEAD.

A good simulation performance is important due to the large number of executions needed for tight approximation bounds. For example, the Prince TI experiments in Table 1 need 5.24×10^{12} simulations of the full cipher. Each of these experiments ran for about 54.31 h on a server machine with an 44-core Intel Xeon E5-2699 CPU, clocked at 2.20 GHz, that supports `AVX2` instructions[3]. As the workload is highly parallelizable, parts of the experiments can also be run on multiple machines and then merged for analysis.

6 Related Work

6.1 Formal Verification Tools

So-called formal verification tools automate the proof of t-probing security (and related notions) independently of a concrete masking scheme. In essence, these tools enumerate all possible sets of t probes in the circuit and, for each of these sets, check whether the distribution of its values depends on the secret inputs. A common limitation of this family of tools is that they are limited in the circuit size and security order they can handle due to computational cost because the number of probe sets increases quickly with t and with the circuit size. Moreover, checking a single set can be a difficult problem when the circuit has a high logical depth.

Exact tools. The simplest but least efficient of the formal tools is VerMI [ANR18], which verifies the independence by enumerating all the randomness values to compute the exact distribution of the probes. The SILVER tool [KSM20] performs this independence verification using binary decision diagrams, which results in a more efficient algorithm overall. These two tools are exact, meaning they always correctly report whether a circuit is t-probing secure or not.

[3]We ran QUANTILE with only 29 worker threads, as more workers incurred cache contention penalties.

Coco [GHP+21, HB21] improves upon the performance of exact tools by introducing two optimizations. First, instead of computing the distributions of wires, it computes correlation sets. Using an implicit representation of these sets encoded in a SAT solver, this technique typically avoids the exponential scaling of the complete distribution computation but over-approximates the leakage (leading to false positives). Second, the list of probe sets to explore can be encoded in the SAT solver, leading to optimization opportunities within the solver, compared to performing explicit checks for every set of probes.

MaskVerif [BBC+19] is another over-approximating formal tool that uses the following property: if e is an expression in which a random bit r does not appear, then replacing every instance of r in a computation by $r \oplus e$ does not change the distribution of the computed values. This property enables it to simplify the algebraic expressions of the probed wires in a similar way to Gaussian elimination until secrets no longer appear in the expressions. Otherwise, the verification fails. This restricted simplification technique and its heuristics lead to a highly efficient algorithm at the cost of false positives. Further enhancing the performance, maskVerif opportunistically verifies the security of sets of more than t probes, which allows it to reduce the total number of sets to verify.

Comparison to Quantile. In general, VerMI and SILVER can usually only be applied to components of a single cipher round like S-Boxes. The over-approximating tools Coco and maskVerif can verify a few rounds of a masked cipher, with maskVerif being considered more efficient overall, especially at higher masking orders. We have tried replicating our results from Table 1 with maskVerif by translating the code Quantile generates for the *runner* (cf. Figure 1) into maskVerif's input language. When looking at complete executions from Table 1, maskVerif goes out of memory on a machine with 120 GiB of RAM. When given round-reduced versions, it verified at most one round of AES DOM, and at most five forward rounds of Prince TI within 8.97 h. In any case, these tools are focused on t-probing security and cannot verify imperfect masking.

6.2 PROLEAD

PROLEAD [MM22], similarly to Quantile, is based on simulating the target circuit and collecting statistics about the values of all the wires. Its main difference to Quantile is that it uses a statistical test (a G-test) whose null hypothesis is the independence between the observed values X and the input secrets S. Furthermore, for the value of S, it performs a fixed-vs-random test: the simulations are grouped in two classes: one where S is fixed to all-zeros, while the other one uses uniformly and freshly sampled S. From the test statistic, PROLEAD can derive a p-value.

Compared to formal verification techniques, PROLEAD shares its main advantage with Quantile: linear scalability with the circuit size. Its false positive rate is also controlled with a p-value, which can be reduced by increasing the number of samples in the test. In contrast, the false positives produced by Coco and maskVerif are deterministic and cannot be worked around with increased computational resources.

Whereas the false positives of PROLEAD are easily controlled with its p-value (with precautions considering the number of different tests performed, as explained in Section 3.1), the false negatives are more problematic. Such false negatives have two root causes: a G-test's intrinsic false negative rate and the use of a fixed-vs-random test.

First, controlling the probability of false negatives of the G-test is more challenging than controlling false positives since their occurrence depends not only on the parameters of the statistical test (such as the number of samples and the significance threshold) but also on the effect size, *i.e.*, on how much the distribution of the set of probes changes when the native values vary. PROLEAD computes a false negative rate for its result by assuming a "small effect" size of $\phi = 0.1$, following [Coh88] (we refer to [Coh88, MM22] for the definition of the effect size ϕ). If the effect size is smaller than $\phi = 0.1$, the false negative rate will be higher than the one computed by PROLEAD. It is crucial to meaningfully define the effect size threshold such that an effect below the threshold can be neglected considering the application domain, as explained in [Coh88] (where PROLEAD's "small" $\phi = 0.1$ threshold is taken from): *"The terms "small", "medium", and "large" are relative, not only to each other, but to the area of behavioral science [...]"*. Since side-channel analysis techniques are designed to detect tiny key-dependent variations in the leakage, it is not unreasonable to assume that a negligible effect size for side-channel analysis is much smaller than one for behavioral science. A better way to select the effect size threshold would be to ensure that leakage from a wire whose key dependence matches that effect size is benign, *i.e.*, that the number of traces needed to mount a successful attack exploiting that leakage is above a targeted security level.

Second, by running its fixed-vs-random test, PROLEAD checks whether the wire distributions are the same when (i) the native secret inputs are zero and (ii) the native secret inputs are uniformly distributed. These distributions may be equal (or very close, as discussed in the previous paragraph), while there is a dependence: other (*i.e.*, some nonzero) fixed native secret values may violate the equality. In such a case, PROLEAD will wrongly report the absence of leakage even when requesting a very low false negative probability. This problem can be solved by iterating the test multiple times while changing the fixed input values or performing a G-test with more rows in the contingency table. Both of these solutions will lead to worse performance (due to requiring more samples) and make the definition of effect size more complicated.

6.3 Mutual information estimation and bounds

The problem of estimating mutual information has attracted attention for many years. However, perhaps surprisingly, the particular problem we are interested in (confidence interval for the mutual information between discrete variables with a large domain and unknown distribution) has not received attention.

A discussion of the properties of simple estimators can be found in [Pan03], and some more detailed discussion also appears in [MCHS23]. A few recent works designed improved estimators (we focus on the discrete case) [APP13, SSK15, HS19], but they do not provide a theoretical convergence analysis for their estimators.

A different line of work considers bounds on the mutual information, assuming that some of the properties of the joint distribution are known (e.g., minimum and maximum values of some marginals or conditionals) [DG97, PDSS16]. Besides the issue of computing such values for our problem, these bounds introduce a fixed gap (due to exploiting a few characteristics of the distributions) that does not shrink with the number of samples.

7 Conclusion and Future Work

In this paper, we introduced Quantile. Using a nonasymptotic statistical theory, Quantile computes statistical bounds on the mutual information between the secrets and the value of a wire in a circuit or between the secret and noisy leakage of this wire. Quantile enabled us to evaluate the effective security order of a masked implementation concretely. The verification method implemented in Quantile scales efficiently to large circuits.

We now discuss future work opportunities. First, although the theory of Section 3 works with any leakage domain, the implementation of Quantile is currently limited to univariate bit leakage. Although efficiently extending it to multivariate leakage is a challenging engineering problem, it would enable the analysis of higher-order masked circuits or analysis within the robust probing model [FGP$^+$18] (*i.e.*, take glitches and transitions into account). Such extensions to Quantile would also enable it to give security bounds against soft analytical side-channel attacks (SASCA) [VGS14], which exploit information from multiple sharings in a single attack.

Next, we observed that our bounds are not very tight, mainly due to worst-case assumptions on the statistical distributions when computing the confidence intervals. Replacing such assumptions with the observed distributions in the samples (while preserving the provable bounds) would enable tighter bounds, *i.e.*, improved security bounds at a lower sampling cost.

Finally, the randomness reuse case studies we performed are fairly simple and non-optimized. We believe that QUANTILE can be used to design efficient masked circuits with smaller imperfections than our examples, e.g., by shuffling the random bits between the rounds.

Acknowledgements

This work was supported by the *Austrian Research Promotion Agency* (FFG) through the BRIDGE project AWARE. This work has been funded in part by SGS.

References

[ANR18] Victor Arribas, Svetla Nikova, and Vincent Rijmen. Vermi: Verification tool for masked implementations. In *25th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2018, Bordeaux, France, December 9-12, 2018*, pages 381–384, 2018.

[APP13] Evan Archer, Il Memming Park, and Jonathan W. Pillow. Bayesian and quasi-bayesian estimators for mutual information from discrete data. *Entropy*, 15:1738–1755, 2013.

[BBC$^+$19] Gilles Barthe, Sonia Belaïd, Gaëtan Cassiers, Pierre-Alain Fouque, Benjamin Grégoire, and François-Xavier Standaert. maskverif: Automated verification of higher-order masking in presence of physical defaults. In *European Symposium on Research in Computer Security – ESORICS*, pages 300–318, 2019.

[BBD$^+$16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In *Conference on Computer and Communications Security – CCS*, pages 116–129, 2016.

[BCG$^+$12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In *Advances in Cryptology – ASIACRYPT*, pages 208–225, 2012.

[BDF$^+$17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In *Advances in Cryptology – EUROCRYPT*, pages 535–566, 2017.

[BDM$^+$20] Sonia Belaïd, Pierre-Évariste Dagand, Darius Mercadier, Matthieu Rivain, and Raphaël Wintersdorff. Tornado: Automatic generation of probing-secure masked bitsliced implementations. In *Advances in Cryptology – EUROCRYPT*, pages 311–341, 2020.

[BKN22] Dusan Bozilov, Miroslav Knezevic, and Ventzislav Nikov. Optimized threshold implementations: securing cryptographic accelerators for low-energy and low-latency applications. *J. Cryptogr. Eng.*, 12:15–51, 2022.

[CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Trans. Computers*, 70:1677–1690, 2021.

[Coh88] Jacob Cohen. *Statistical power analysis for the behavioral sciences.* Routledge, 1988.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems – CHES*, pages 13–28, 2002.

[CS19] Gaëtan Cassiers and François-Xavier Standaert. Towards globally optimized masking: From low randomness to low noise rate or probe isolating multiplications with reduced randomness and security against horizontal attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pages 162–198, 2019.

[CS21] Gaëtan Cassiers and François-Xavier Standaert. Provably secure hardware masking in the transition- and glitch-robust probing model: Better safe than sorry. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pages 136–158, 2021.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory - Second Edition.* John Wiley & Sons, 2006.

[dCGRP19] Eloi de Chérisey, Sylvain Guilley, Olivier Rioul, and Pablo Piantanida. Best information is most successful mutual information and success rate in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pages 49–79, 2019.

[DDF19] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. *J. Cryptol.*, 32:151–177, 2019.

[DG97] S.S. Dragomir and C.J. Goh. Some bounds on entropy measures in information theory. *Applied Mathematics Letters*, 10:23–28, 1997.

[DR98] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In *Smart Card Research and Advanced Applications – CARDIS*, pages 277–284, 1998.

[FGP$^+$18] Sebastian Faust, Vincent Grosso, Santos Merino Del Pozo, Clara Paglialonga, and François-Xavier Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pages 89–120, 2018.

[GHP$^+$21] Barbara Gigerl, Vedad Hadzic, Robert Primas, Stefan Mangard, and Roderick Bloem. Coco: Co-design and co-verification of masked software implementations on cpus. In *USENIX Security Symposium*, pages 1469–1468, 2021.

[GMK16] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In *Conference on Computer and Communications Security – CCS*, page 3, 2016.

[GMKM18] Hannes Groß, Lauren De Meyer, Martin Krenn, and Stefan Mangard. Masking the AES with only two random bits. *IACR Cryptol. ePrint Arch.*, page 1007, 2018.

[HB21] Vedad Hadzic and Roderick Bloem. COCOALMA: A versatile masking verifier. In *Formal Methods in Computer Aided Design, FMCAD 2021, New Haven, CT, USA, October 19-22, 2021*, pages 1–10, 2021.

[HS19] Damián G. Hernández and Inés Samengo. Estimating the mutual information between two discrete, asymmetric variables with limited samples. *Entropy*, 21:623, 2019.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology – CRYPTO*, pages 463–481, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Advances in Cryptology – CRYPTO*, pages 388–397, 1999.

[KMMS22] David Knichel, Amir Moradi, Nicolai Müller, and Pascal Sasdrich. Automated generation of masked hardware. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pages 589–629, 2022.

[KSM20] David Knichel, Pascal Sasdrich, and Amir Moradi. SILVER - statistical independence and leakage verification. In *Advances in Cryptology – ASIACRYPT*, pages 787–816, 2020.

[LMW14] Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs. Gate-level masking under a path-based leakage metric. In *Cryptographic Hardware and Embedded Systems – CHES*, pages 580–597, 2014.

[Man04] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In *Topics in Cryptology – CT-RSA*, pages 222–235, 2004.

[MCHS23] Loïc Masure, Gaëtan Cassiers, Julien M. Hendrickx, and François-Xavier Standaert. Information bounds and convergence rates for side-channel security evaluators. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pages 522–569, 2023.

[MM22] Nicolai Müller and Amir Moradi. PROLEAD A probing-based hardware leakage detection tool. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pages 311–348, 2022.

[Pan03] Liam Paninski. Estimation of entropy and mutual information. *Neural Comput.*, 15:1191–1253, 2003.

[PDSS16] Pantelimon George Popescu, Sever Silvestru Dragomir, Emil Slusanschi, and Octavian Stanasila. Bounds for kullback-leibler divergence. *Electronic Journal of Differential Equations*, pages 1–6, 2016.

[PGMP19] Thomas Prest, Dahmun Goudarzi, Ange Martinelli, and Alain Passelègue. Unifying leakage models on a rényi day. In *Advances in Cryptology – CRYPTO*, pages 683–712, 2019.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *Advances in Cryptology – EUROCRYPT*, pages 142–159, 2013.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, pages 200–210, 2001.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In *Advances in Cryptology – CRYPTO*, pages 764–783, 2015.

[Rig15] Philippe Rigollet. 18. s997: High dimensional statistics. *(Lecture Notes), Cambridge, MA, USA: MIT Open-CourseWare*, 2015.

[SM21] Aein Rezaei Shahmirzadi and Amir Moradi. Re-consolidating first-order masking schemes nullifying fresh randomness. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, pages 305–342, 2021.

[SSK15] Junhee Seok and Yeong Seon Kang. Mutual information between discrete variables with many categories using recursive adaptive partitioning. *Scientific reports*, 5:10981, 2015.

[Sta20] François-Xavier Standaert. Security models and metrics for analyzing masked implementations, 2020.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In *Advances in Cryptology – ASIACRYPT*, pages 282–296, 2014.

[VH14] Ramon Van Handel. Probability in high dimension. Technical report, PRINCETON UNIV NJ, 2014.

A Proof of Corollary 1

A.1 Sub-Gaussian Random Variables

Because of their instrumental importance throughout proofs in this paper, we briefly recall tail bounds of the normal distribution and the so-called sub-Gaussian random variables.

Proposition 2 (Gaussian tail bounds)**.** *Let $X \sim \mathcal{N}\left(\mu, \sigma^2\right)$ be a normally distributed variable with expected value μ and variance σ^2. Its tail bounds satisfy*

$$\mathbf{Pr}\left[X \geq \mu + \varepsilon\right] \leq e^{\frac{-\varepsilon^2}{2\sigma^2}}, \qquad \textit{and} \qquad \mathbf{Pr}\left[X \leq \mu - \varepsilon\right] \leq e^{\frac{-\varepsilon^2}{2\sigma^2}}. \tag{7}$$

Proof. See [Rig15, Example 2.1] or [VH14, Example 3.4] for the derivation. □

Definition 2 (Sub-Gaussian random variable)**.** A random variable X with finite expectation $\mu = \mathbf{E}\left[X\right]$ is said to be sub-Gaussian if there is a positive number σ^2 such that for all $\lambda \in \mathbb{R}$ it satisfies

$$\mathbf{E}\left[e^{\lambda(X-\mu)}\right] \leq e^{\frac{\sigma^2\lambda^2}{2}}. \tag{8}$$

We denote this with $X \lesssim \mathcal{N}\left(\mu, \sigma^2\right)$ and call σ^2 the *variance proxy.*

Showing that a random variable is sub-Gaussian is desirable because they obey the same Gaussian tail bounds from Proposition 2. In fact, other useful properties of normally distributed random variables also apply. For example, scaling a sub-Gaussian $X \lesssim \mathcal{N}\left(\mu, \sigma\right)$ by $a \in \mathbb{R}$, yields another sub-Gaussian random variable $aX \lesssim \mathcal{N}\left(a\mu, a^2\sigma^2\right)$. Similarly, adding independent sub-Gaussian random variables $X_1 \lesssim \mathcal{N}\left(\mu_1, \sigma_1^2\right)$ and $X_2 \lesssim \mathcal{N}\left(\mu_2, \sigma_2^2\right)$ produces a new sub-Gaussian $X_1 + X_2 \lesssim \mathcal{N}\left(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2\right)$ [Rig15].

Finally, we recall a method for showing that a functions of random variables, where the influence of any one random variable cannot be arbitrarily large, are sub-Gaussian.

Proposition 3 (McDiarmid's inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables with respective domains $\mathcal{X}_1, \ldots, \mathcal{X}_n$, and $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_n \to \mathbb{R}$ be a function. If there are constants $c_1, \ldots, c_n$ such that for all $x_1 \in \mathcal{X}_1, \ldots, x_n \in \mathcal{X}_n$, we have*

$$\sup_{x_i' \in \mathcal{X}_i} |f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, x_i', \ldots, x_n)| \le c_i \ , \tag{9}$$

then $Y = f(X_1, \ldots, X_n) \lesssim \mathcal{N}(\mu, \sigma^2)$ with $\mu = \mathbf{E}[Y]$ and $\sigma^2 = \frac{1}{4}\sum_{i=1}^n c_i^2$.

Proof. See [VH14, Theorem 3.11] for the derivation. □

A.2 Deviation of The Plug-in Entropy Estimator

Lemma 2 (The plug-in entropy estimator is sub-Gaussian)**.** *For a discrete random variable X, the plug-in entropy estimator $\widehat{\mathrm{H}}_n(X)$ from Definition 1 is sub-Gaussian, i.e., $\widehat{\mathrm{H}}_n(X) \lesssim \mathcal{N}(\mu, \sigma^2)$, with $\mu = \mathbf{E}\left[\widehat{\mathrm{H}}_n(X)\right]$ and $\sigma^2 = \frac{n}{4}\mathrm{H}_{bin}\left(n^{-1}\right)^2$.*

Proof. $\widehat{\mathrm{H}}_n(X)$ is a function of the random variables $\boldsymbol{X}_n = (X_i)_{i=1}^n$, which are all independent and distributed identically to X. Therefore, if we are able to show that changes in the outcome of X_i have a limited influence on $\widehat{\mathrm{H}}_n(X)$, we can apply McDiarmid's inequality from Proposition 3 to prove the sub-Gaussianity of $\widehat{\mathrm{H}}_n(X)$.

In the following, we analyze what happens to $\widehat{\mathrm{H}}_n$ when one of the random variable, without loss of generality we choose the i-th one, has a different outcome. Let function $h : p \mapsto -p\log_2(p)$ represent entropy summands. Furthermore, let $(x_i)_{i=1}^n$ be the outcomes of sampling $\boldsymbol{X}_n$, and let x_i' be a different outcome of the i-th random variable. The greatest change in $\widehat{\mathrm{H}}_n$ is given by

$$\sup_{x_i' \in \mathcal{X}} \left|\widehat{\mathrm{H}}_n(x_1, \ldots, x_i, \ldots, x_n) - \widehat{\mathrm{H}}_n(x_1, \ldots, x_i', \ldots, x_n)\right| =$$

$$\sup_{x_i' \in \mathcal{X}} \left| \sum_{x \in \mathcal{X}} h\left(\sum_{j=1}^n \frac{\mathbb{1}_{\{x\}}(x_j)}{n}\right) - \sum_{x \in \mathcal{X}} h\left(\sum_{j=1}^{i-1} \frac{\mathbb{1}_{\{x\}}(x_j)}{n} + \frac{\mathbb{1}_{\{x\}}(x_i')}{n} + \sum_{j=i+1}^{n} \frac{\mathbb{1}_{\{x\}}(x_j)}{n}\right)\right|.$$

For all values $x \in \mathcal{X} \setminus \{x_i, x_i'\}$, the term $h\left(\sum_{j=1}^n \frac{\mathbb{1}_{\{x\}}(x_j)}{n}\right)$ appears both positively and negatively inside the absolute value, canceling them. As for x_i (and x_i'), the argument to the negative terms decreases (increases) by $\frac{1}{n}$. Let $c_i = \sum_{j=1}^n \mathbb{1}_{\{x_i\}}(x_j)$ and let $c_i' = \sum_{j=1}^n \mathbb{1}_{\{x_i'\}}(x_j)$. The above supremum simplifies to

$$\sup_{x_i' \in \mathcal{X}} \left| h\left(\frac{c_i}{n}\right) + h\left(\frac{c_i'}{n}\right) - h\left(\frac{c_i - 1}{n}\right) - h\left(\frac{c_i' + 1}{n}\right)\right|. \tag{10}$$

We now analyze the parts of the absolute value depending on c_i and c_i' separately. We see that $h\left(\frac{c_i}{n}\right) - h\left(\frac{c_i-1}{n}\right)$ is monotonically decreasing because it has the strictly negative derivative $\frac{1}{n}\left(\log_2\left(\frac{c_i-1}{n}\right) - \log_2\left(\frac{c_i}{n}\right)\right)$. Therefore $h\left(\frac{c_i}{n}\right) - h\left(\frac{c_i-1}{n}\right)$ has its supremum and infimum at the interval ends $c_i = 1$, respectively $c_i = n$, *i.e.*,

$$-h\left(\frac{n-1}{n}\right) \le h\left(\frac{c_i}{n}\right) - h\left(\frac{c_i - 1}{n}\right) \le h\left(\frac{1}{n}\right).$$

Doing a similar analysis for parts of the absolute value depending on c_i' shows that

$$-h\left(\frac{1}{n}\right) \le h\left(\frac{c_i'}{n}\right) - h\left(\frac{c_i' + 1}{n}\right) \le h\left(\frac{n-1}{n}\right).$$

The absolute value in (10) reaches its largest value when either both the c_i and c'_i-dependent parts reach their supremum or both reach their infimum. Therefore, finally

$$\sup_{x'_i \in \mathcal{X}} \left| \widehat{\mathrm{H}}_n\left(x_1, \ldots, x_i, \ldots, x_n\right) - \widehat{\mathrm{H}}_n\left(x_1, \ldots, x'_i, \ldots, x_n\right) \right| \leq h\left(\frac{1}{n}\right) + h\left(\frac{n-1}{n}\right) = \mathrm{H}_{\mathrm{bin}}\left(\frac{1}{n}\right).$$

Since the bound is independent of the X_i whose value changes, we can apply McDiarmid's inequality from Proposition 3 to get $\widehat{\mathrm{H}}_n(X) \lesssim \mathcal{N}\left(\mathbf{E}\left[\widehat{\mathrm{H}}_n(X)\right], \frac{n}{4}\mathrm{H}_{\mathrm{bin}}\left(n^{-1}\right)^2\right)$. □

A.3 Deviation of Conditional Entropy

Lemma 3 (Conditional entropy is sub-Gaussian)**.** *For discrete random variables X and Y taking values in domains $\mathcal{X}$ and $\mathcal{Y}$ respectively, the entropy $f(y) = \mathrm{H}(X|Y=y)$ is a sub-Gaussian random variable, with $f(Y) \lesssim \mathcal{N}\left(\mathrm{H}(X|Y), \frac{1}{4}\log_2^2|\mathcal{X}|\right)$.*

Proof. Function f is a bounded, with $0 \leq f(y)$ because entropy is non-negative and $f(y) \leq \mathrm{H}(X) \leq \log_2|\mathcal{X}|$. Since $\mathbf{sup}_{y' \in \mathcal{Y}} |f(y) - f(y')| \leq \log_2|\mathcal{X}|$, we can apply McDiarmid's inequality to show $f(Y)$ is sub-Gaussian with mean $\mu = \mathbf{E}[\mathrm{H}(X|Y=y)] = \mathrm{H}(X|Y)$ and variance proxy $\sigma^2 = \frac{1}{4}\log_2^2|\mathcal{X}|$. □

A.4 Proof of Mutual Information Bounds

Theorem 1. *Let X, D and S be discrete random variables with domains $\mathcal{X}$, $\mathcal{D}$ and $\mathcal{S}$, and let D and S be distributed uniformly. Let n_D, $n_{X|D}$, $n_{S,D}$, and $n_{X|S,D}$ be positive integers and $\delta > 0$ be a real number. Let $\boldsymbol{D} = (D_i)_{i=1}^{n_D}$, $\boldsymbol{D}' = (D'_i)_{i=1}^{n_{S,D}}$, $\boldsymbol{S} = (S_i)_{i=1}^{n_{S,D}}$ be vectors of independent random variables distributed identically to D and S respectively. Furthermore, let $\boldsymbol{X}_i = (X_{i,j})_{j=1}^{n_{X|D}}$ and $\boldsymbol{X}'_i = \left(X'_{i,j}\right)_{j=1}^{n_{X|S,D}}$ be vectors of independent random variables distributed identically to $X|D=D_i$ and $X|S=S_i, D=D'_i$ respectively. Finally, let*

$$\widehat{\mathrm{H}}_{n_D, n_{X|D}}(X|D) = \frac{1}{n_D}\sum_{i=1}^{n_D} \widehat{\mathrm{H}}_{n_{X|D}}(X|D=D_i),$$

$$\widehat{\mathrm{H}}_{n_{S,D}, n_{X|S,D}}(X|S,D) = \frac{1}{n_{S,D}}\sum_{i=1}^{n_{S,D}} \widehat{\mathrm{H}}_{n_{X|S,D}}(X|S=S_i, D=D'_i)$$

$$\widehat{\mathrm{I}}(X;S|D) = \widehat{\mathrm{H}}_{n_D, n_{X|D}}(X|D) - \widehat{\mathrm{H}}_{n_{S,D}, n_{X|S,D}}(X|S,D)$$

be estimates for $\mathrm{H}(X|D)$, $\mathrm{H}(X|D,S)$, *and* $\mathrm{I}(X;S|D)$ *respectively. Then*

$$\mathrm{I}_{LB}(X;S|D) = \widehat{\mathrm{I}}(X;S|D) - \log_2\left(1 + \frac{|\mathcal{X}|-1}{n_{X|S,D}}\right) - \sigma\sqrt{2\log\left(\delta^{-1}\right)} \quad \textit{and}$$

$$\mathrm{I}_{UB}(X;S|D) = \widehat{\mathrm{I}}(X;S|D) + \log_2\left(1 + \frac{|\mathcal{X}|-1}{n_{X|D}}\right) + \sigma\sqrt{2\log\left(\delta^{-1}\right)}, \quad \textit{with}$$

$$\sigma^2 = \frac{\log_2^2|\mathcal{X}|}{4n_D} + \frac{n_{X|D}}{4n_D}\mathrm{H}_{bin}\left({n_{X|D}}^{-1}\right)^2 + \frac{\log_2^2|\mathcal{X}|}{4n_{S,D}} + \frac{n_{X|S,D}}{4n_{S,D}}\mathrm{H}_{bin}\left({n_{X|S,D}}^{-1}\right)^2, \quad \textit{satisfy}$$

$$\mathbf{Pr}\left[\mathrm{I}_{LB}(X;S|D) < \mathrm{I}(X;S|D)\right] > 1-\delta \quad \textit{and} \quad \mathbf{Pr}\left[\mathrm{I}_{UB}(X;S|D) > \mathrm{I}(X;S|D)\right] > 1-\delta. \tag{11}$$

Proof. We break down the difference between the estimate and real mutual information as

$$\widehat{\mathrm{I}}(X;S|D) - \mathrm{I}(X;S|D) = P + Q + R - U - V - W, \tag{12}$$

where

$$P = \frac{1}{n_D}\sum_{i=1}^{n_D} p(D_i), \qquad Q = \frac{1}{n_D}\sum_{i=1}^{n_D} q(D_i), \qquad R = \frac{1}{n_D}\sum_{i=1}^{n_D} r(D_i),$$

$$U = \frac{1}{n_{S,D}}\sum_{i=1}^{n_{S,D}} u(S_i, D_i'), \quad V = \frac{1}{n_{S,D}}\sum_{i=1}^{n_{S,D}} v(S_i, D_i'), \quad W = \frac{1}{n_{S,D}}\sum_{i=1}^{n_{S,D}} w(S_i, D_i'),$$

$$\begin{aligned}
p(d) &= \widehat{\mathrm{H}}_{n_{X|D}}(X|D=d) - \mathbf{E}\left[\widehat{\mathrm{H}}_{n_{X|D}}(X|D=d)\right],\\
q(d) &= \mathbf{E}\left[\widehat{\mathrm{H}}_{n_{X|D}}(X|D=d)\right] - \mathrm{H}(X|D=d),\\
r(d) &= \mathrm{H}(X|D=d) - \mathrm{H}(X|D),\\
u(s,d) &= \widehat{\mathrm{H}}_{n_{X|S,D}}(X|S=s,D=d) - \mathbf{E}\left[\widehat{\mathrm{H}}_{n_{X|S,D}}(X|S=s,D=d)\right],\\
v(s,d) &= \mathbf{E}\left[\widehat{\mathrm{H}}_{n_{X|S,D}}(X|S=s,D=d)\right] - \mathrm{H}(X|S=s,D=d),\\
w(s,d) &= \mathrm{H}(X|S=s,D=d) - \mathrm{H}(X|S,D).
\end{aligned}$$

Applying Lemma 2 to $p(D_i)$ and $u(S_i, D_i')$, and Lemma 3 to $r(D_i)$ and $w(S_i, D_i')$, we get

$$p(D_i) \lesssim \mathcal{N}\left(0, \frac{n_{X|D}}{4}\mathrm{H}_{\mathrm{bin}}\left({n_{X|D}}^{-1}\right)^2\right), \qquad r(D_i) \lesssim \mathcal{N}\left(0, \frac{\log_2^2|\mathcal{X}|}{4}\right),$$

$$u(S_i, D_i') \lesssim \mathcal{N}\left(0, \frac{n_{X|S,D}}{4}\mathrm{H}_{\mathrm{bin}}\left({n_{X|S,D}}^{-1}\right)^2\right), \qquad w(S_i, D_i') \lesssim \mathcal{N}\left(0, \frac{\log_2^2|\mathcal{X}|}{4}\right).$$

Crutially, Proposition 1 bounds the bias $q(D_i)$ to the range $\left[-\log_2\left(1+\frac{|\mathcal{X}|-1}{n_{X|D}}\right), 0\right]$ and $-v(S_i, D_i')$ to the range $\left[0, \log_2\left(1+\frac{|\mathcal{X}|-1}{n_{X|S,D}}\right)\right]$. Rearranging (12) and applying linearity properties of sub-Gaussians, we get

$$\widehat{\mathrm{I}}(X;S|D) - \mathrm{I}(X;S|D) - Q + V = P + R - U - W \lesssim \mathcal{N}\left(0, \sigma^2\right).$$

Setting δ as the tail bound probability in (7), thus $\varepsilon = \sigma\sqrt{2\log\left(\delta^{-1}\right)}$, we get

$$\mathbf{Pr}\left[\mathrm{I}_{\mathrm{LB}}(X;S|D) \geq \mathrm{I}(X;S|D)\right] \leq \mathbf{Pr}\left[\widehat{\mathrm{I}}(X;S|D) - \mathrm{H}(X|D) - Q + V \geq \varepsilon\right] \leq \delta \quad \text{and}$$

$$\mathbf{Pr}\left[\mathrm{I}_{\mathrm{UB}}(X;S|D) \leq \mathrm{I}(X;S|D)\right] \leq \mathbf{Pr}\left[\widehat{\mathrm{I}}(X;S|D) - \mathrm{H}(X|D) - Q + V \leq -\varepsilon\right] \leq \delta.$$

Inverting these probabilities gives the statements in (11), concluding the proof. □

Remark 1. The total number of samples, which is proportional to the computational cost, is $n_D n_{X|D} + n_{S,D} n_{X|S,D}$. To get symmetric upper and lower bounds, one should take $n_D = n_{S,D}$ and $n_X = n_{X|D} = n_{X|S,D}$, giving Corollary 1.

Remark 2. We expect that our bounds that use Lemma 3 are not be very tight in practice, given that we use worst-case interval bounds, while we expect the entropies to have low (or even no) variance with data D or secret S. If a bound on these variances is available, then using Bernstein's inequality will lead to improved bounds.

A.5 Upper Bound Over a Full Computation

Lemma 4. *Let D and S be uniformly distributed discrete random variables with domains $\mathcal{D}$ and $\mathcal{S}$, and let D and S be distributed uniformly. Furthermore, let $(X_k)_{k=1}^{n_{comp}}$ be (possibly* *dependent) random variables whose mutual information* $\mathrm{I}(X_k; S|D)$ *is bounded as in Theorem 1 with confidence* $1 - \delta/n_{comp} > 0$*. Then*

$$\mathbf{Pr}\left[\sup_k \mathrm{I}(X_k; S|D) < \sup_k \mathrm{I}_{UB}(X_k; S|D)\right] > 1 - \delta. \tag{13}$$

Proof. We prove this by looking at the inverse probability, removing the supremum of $\mathrm{I}(X_k; S|D)$ by representing the inequality as an union of events, relaxing said events and applying an union bound for the probabilities.

$$\begin{aligned}
&\mathbf{Pr}\left[\sup_k \mathrm{I}(X_k; S|D) \geq \sup_k \mathrm{I}_{\mathrm{UB}}(X_k; S|D)\right] \\
=&\mathbf{Pr}\left[\bigcup_{l=1}^{n_{\mathrm{comp}}} \left\{\mathrm{I}(X_l; S|D) \geq \sup_k \mathrm{I}_{\mathrm{UB}}(X_k; S|D)\right\}\right] \\
\leq&\mathbf{Pr}\left[\bigcup_{l=1}^{n_{\mathrm{comp}}} \left\{\mathrm{I}(X_l; S|D) \geq \mathrm{I}_{\mathrm{UB}}(X_l; S|D)\right\}\right] \\
\leq& \sum_{k=1}^{n_{\mathrm{comp}}} \mathbf{Pr}\left[\mathrm{I}(X_l; S|D) \geq \mathrm{I}_{\mathrm{UB}}(X_l; S|D)\right] \leq n_{\mathrm{comp}} \cdot \frac{\delta}{n_{\mathrm{comp}}} = \delta.
\end{aligned}$$

Invering the probabilities again gives (13) concluding the proof. □

B Proof of Lemma 1

We first recall an instrumental result which bounds the conditional entropy of processed random variables.

Proposition 4 (Fano's inequality)**.** *Let* X*,* Y *and* X' *be random variables with domains* $\mathcal{X}$*,* $\mathcal{Y}$ *and* $\mathcal{X}$ *that obey the Markov chain* $X \to Y \to X'$*, with* $p = \mathbf{Pr}[X \neq X']$*. Then*

$$\mathrm{H}(X|Y) \leq \mathrm{H}(X|X') \leq \mathrm{H}_{bin}(p) + p \log_2 |\mathcal{X}|.$$

Proof. See [CT06, Theorem 2.10.1] for the derivation. □

Proposition 5. *Let* $\boldsymbol{X} = (X_i)_{i=1}^n$ *and* $\boldsymbol{Y} = (Y_i)_{i=1}^n$ *be two vectors of random variables. Then* $\mathrm{I}(\boldsymbol{X}; \boldsymbol{Y}) \leq n\,\mathrm{I}(X; Y)$*.*

Proof. See [dCGRP19, Lemma 3] for the derivation. □

Lemma 1. *Let* $n_{\mathsf{Adv}} \geq 0$ *be an integer,* S *be a random variable with domain* $\mathcal{S}$ *and* $\boldsymbol{D} = (D_i)_{i=1}^{n_{\mathsf{Adv}}}$ *be a vector of random variables independent of* S *with the same domain* $\mathcal{D}$*. Furthermore, let* $f : \mathcal{S} \times \mathcal{D} \to \mathcal{L}$ *be a randomized function modelling a memoryless channel, and let* $\boldsymbol{L} = (L_i)_{i=1}^{n_{\mathsf{Adv}}}$ *be a vector of random variables with* $L_i = f(S, D_i)$*. Let* $\mathsf{Adv} : \mathcal{D}^{n_{\mathsf{Adv}}} \times \mathcal{L}^{n_{\mathsf{Adv}}} \to \mathcal{S}$ *be a (potentially randomized) function attempting to recover the value of* S *as* $S' = \mathsf{Adv}(\boldsymbol{D}, \boldsymbol{L})$ *and let* $r_{\mathsf{Adv}} = \mathbf{Pr}[S' = S]$*. Then, assuming* $\mathrm{I}(L; S|D) \neq 0$*,*

$$n_{\mathsf{Adv}} \geq \frac{\mathrm{H}(S) - (1 - r_{\mathsf{Adv}}) \log_2(|\mathcal{S}| - 1) - \mathrm{H}_{bin}(r_{\mathsf{Adv}})}{\mathrm{I}(L; S|D)}. \tag{6}$$

Proof. Since S and $\boldsymbol{D}$ are independent, we have

$$\begin{aligned}
\mathrm{I}((S, \boldsymbol{D}); (\boldsymbol{L}, \boldsymbol{D})) &= \mathrm{H}(S, \boldsymbol{D}) - \mathrm{H}(S, \boldsymbol{D}|\boldsymbol{L}, \boldsymbol{D}) \\
&= \mathrm{H}(S) + \mathrm{H}(\boldsymbol{D}) - \mathrm{H}(S|\boldsymbol{L}, \boldsymbol{D}).
\end{aligned} \tag{14}$$

The random variables S, $(\boldsymbol{L}, \boldsymbol{D})$ and S' rescribe a Markov chain, *i.e.*, $(\boldsymbol{L}, \boldsymbol{D})$ conditionally depends on S because $\boldsymbol{L}$ conditionally depends on S, whereas S' conditionally depends on $(\boldsymbol{L}, \boldsymbol{D})$ but not S. Therefore, Fano's inequality from Proposition 4 applies, giving

$$\mathrm{H}\left(S|\boldsymbol{L}, \boldsymbol{D}\right) \leq \mathrm{H}\left(S|S'\right) \leq \mathrm{H}_{\mathrm{bin}}\left(r_{\mathsf{Adv}}\right) + \left(1 - r_{\mathsf{Adv}}\right)\log_2|\mathcal{S}|\,. \tag{15}$$

Alternatively, the mutual information $\mathrm{I}\left((S, \boldsymbol{D})\,;(\boldsymbol{L}, \boldsymbol{D})\right)$ can be broken down as

$$\begin{aligned}\mathrm{I}\left((S, \boldsymbol{D})\,;(\boldsymbol{L}, \boldsymbol{D})\right) &= \mathrm{H}\left(\boldsymbol{L}, \boldsymbol{D}\right) - \mathrm{H}\left(\boldsymbol{L}, \boldsymbol{D}|S, \boldsymbol{D}\right) = \mathrm{H}\left(\boldsymbol{L}, \boldsymbol{D}\right) - \mathrm{H}\left(\boldsymbol{L}|S, \boldsymbol{D}\right)\\ &= \mathrm{H}\left(\boldsymbol{D}\right) + \mathrm{H}\left(\boldsymbol{L}|\boldsymbol{D}\right) - \mathrm{H}\left(\boldsymbol{L}|S, \boldsymbol{D}\right) = \mathrm{H}\left(\boldsymbol{D}\right) + \mathrm{I}\left(\boldsymbol{L}; S|\boldsymbol{D}\right).\end{aligned} \tag{16}$$

Combining equations (14) and (16), and subsequently applying inequality (15) gives

$$\begin{aligned}\mathrm{I}\left(\boldsymbol{L}; S|\boldsymbol{D}\right) &= \mathrm{H}\left(S\right) - \mathrm{H}\left(S|\boldsymbol{L}, D\right)\\ &\geq \mathrm{H}\left(S\right) - \mathrm{H}_{\mathrm{bin}}\left(r_{\mathsf{Adv}}\right) - \left(1 - r_{\mathsf{Adv}}\right)\log_2|\mathcal{S}|\,.\end{aligned} \tag{17}$$

Finally, since f models a memoryless channel, we can apply Proposition 5 to $(S, \ldots, S)$ and $\boldsymbol{L}$ (conditionning everything on $\boldsymbol{D}$), finding that $\mathrm{I}\left(\boldsymbol{L}; S|\boldsymbol{D}\right) \leq n_{\mathsf{Adv}}\,\mathrm{I}\left(L; S|D\right)$. Combining this with (17) gives (6) which concludes the proof.

□