IACR Communications in Cryptology
ISSN 3006-5496, Vol. 1, No. 4, 35 pages.

https://doi.org/10.62056/ayl8ksdja

Check for updates

On Loopy Belief Propagation for SASCAs

An Analysis and Empirical Study of the Inference Problem

Rishub Nagpal[1], Gaëtan Cassiers[a, 2,3], Robert Primas[4], Christian Knoll[b, 5], Franz Pernkopf[1] and Stefan Mangard[1]

[1] Graz University of Technology, Graz, Austria
[2] CryptoExperts, Paris, France
[3] UCLouvain, ICTEAM, Crypto Group, Louvain-la-Neuve, Belgium
[4] Intel Labs, Portland, USA
[5] Levata GmbH, Graz, Austria

Abstract. Profiled power analysis is one of the most powerful forms of passive side-channel attacks. Over the last two decades, many works have analyzed their impact on cryptographic implementations as well as corresponding countermeasure techniques. To date, the most advanced variants of profiled power analysis are based on Soft-analytical Side-Channel Attacks (SASCA). After the initial profiling phase, a SASCA adversary creates a probabilistic graphical model, called a factor graph, of the target implementation and encodes the results of the previous step as prior information. Then, an inference algorithm such as loopy Belief Propagation (BP) can be used to recover the distribution of a target variable in the graph, i.e., sensitive data/keys.

Designers of cryptographic implementations aim to reduce information leakage as much as possible and assess how much leakage can be allowed without compromising security requirements. Despite the existence of many works on profiled power analysis, it is still notoriously difficult to state under which conditions a cryptographic implementation provides sufficient protection against a profiling attacker with certain capabilities. In particular, it is unknown when a BP-based attack is optimal or whether tuning some heuristics in that algorithm may significantly strengthen the attack.

This knowledge gap led us to investigate the effectiveness of BP for SASCAs by studying the modes of failures of BP in the context of the SASCA, and systematically analyzing the behavior of BP on practically-relevant factor graphs. We use exact inference to gauge the quality of the approximation provided by BP. Through this assessment, we show that there exists a significant disparity between BP and exact inference in terms of guessing entropy when performing SASCAs on several classes of factor graphs. We further review and analyze various BP improvement heuristics from the literature.

Keywords: Power Analysis · SASCAs · Belief Propagation · Machine Learning

1 Introduction

Side-channel attacks (SCA) are an important threat to the security of embedded devices that perform cryptographic operations. Using physical leakage channels such as power

E-mail: rishub.nagpal@tugraz.at (Rishub Nagpal), gaetan.cassiers@uclouvain.be (Gaëtan Cassiers), robert.primas@intel.com (Robert Primas), knoll@levata.at (Christian Knoll), pernkopf@tugraz.at (Franz Pernkopf), stefan.mangard@tugraz.at (Stefan Mangard)

[a]Part of this research was done while with Graz University of Technology.
[b]Part of this research was done while with Graz University of Technology.

Received: 2024-10-09 Accepted: 2024-12-03

consumption of electromagnetic radiation [KJJ99; QS01], these attacks often enable key recovery.

Since the introduction of SCAs, multiple countermeasures have been introduced, with the goal of increasing the number of traces (i.e., number of leaky executions measured) needed to perform an attack. For example, a common algorithmic countermeasure is masking, which consists in applying a secret-sharing scheme to split every variable in the computation into $t + 1$ randomized shares such that the observation of up to t shares does not reveal any information about the sensitive values [ISW03; DDF19]. Another approach is the use of leakage-resilient cryptography, which essentially ensures that, for any secret key, at most a few different leakage traces can be measured [BBC+20; DEM+20]. As a result, it is important to study the feasibility of attacks that require very few traces (typically in the single-digit range) - known as simple power analysis (SPA) - as opposed to the better studied differential power analysis (DPA) where the number of attack traces is large. Single-trace attacks are also relevant for public key cryptography, be it RSA, ECC or, more recently, (mostly lattice-based) post-quantum cryptographic algorithms [PPM17; PP19; BBPS19; KPP20; NDGJ21; HHP+21; HMS+23].

Early side-channel attacks relied only on weak and generic assumption about the structure of the leakage, e.g., for the differential power analysis [KJJ99] (DPA), the only assumption is that the average value of the leakage depends on an intermediate value in the computation. While such a generic assumption is broadly applicable, it leads to a suboptimal attack regarding the required number of traces. On the other hand, profiled attacks build a model of the leakage of the target using a set of profiling traces for which the secret is known. This class of attacks was first introduced with the Gaussian templates attack [CRR02], which have been extended in various ways [SLP05; CK13; CK14; CDSU23]. More recently, deep learning has been proposed as a powerful class of models. Using a good model, a profiled attack exploits more information content [RSV+11; BHM+19; MCHS23] from the traces than non-profiled attacks, reducing the number of traces required for a successful attack [MDP20]. Both profiled and non-profiled side-channel attacks typically follow a divide-and-conquer approach: they target an intermediate state in a computation that depends on a small part of the secret, such that the possible secret values can be enumerated. This approach is however suboptimal: the constraint on the target intermediate variables restricts them to a small fraction of all intermediate variables in the computations, and as a result, only a small fraction of the leakage is used. Soft-analytical Side-Channel Attacks (SASCA) [VGS14] have been introduced as a way to solve this issue. SASCAs take advantage of the knowledge of the computations performed by the target to build a probabilistic graphical model, called a *factor graph*, that encodes the relationships between the intermediate variables in the computation. In these graphs, factors are added to represent the soft information (i.e., a probability distribution) derived from the profiled model for each variable. Then, an inference algorithm such as loopy Belief Propagation (BP)[1] can be used to recover the distribution of a target variable in the graph (i.e., secret data/keys) with fewer traces than a divide-and-conquer attack.

Belief propagation is an algorithm to perform Bayesian inference. That is, given probabilistic information about multiple variables and relationships between these variables, it computes the marginal probability distribution of a variable. BP operates over a factor graph, which is a bipartite graph with variable nodes that represent the variables of the inference problem and factor nodes that correspond to the relationships[2] between the variables. When the factor graph is acyclic, BP computes exact marginal distributions, while for cyclic graphs, which are very common when performing SASCAs, there is little guarantee about the accuracy of the result of the algorithm.

[1]In this work, "Belief Propagation" or BP generally refers to the loopy BP algorithm.

[2]In the context of SASCAs, these relationships are deterministic functions, however, in general the relationships can be arbitrary.

Despite the existence of a plethora of works on profiled attacks, it is still notoriously difficult to state under which conditions a cryptographic implementation provides sufficient protection against a profiling attacker with certain capabilities. While recent works improved the bounds for the information content of a trace about one intermediate variable [MCHS23; CRBO24], it is still unclear to which extent such information can be combined to infer the secrets: we do not have tight bounds for the Bayesian inference problem, nor on how well practical methods such as BP can behave. While works such as [KPP20; YK21; PPM17; PP19; KPP20; NDGJ21; HHP+21; HMS+23] do experiment with various tweaks of BP aimed at improving the accuracy of the inference, they are often not motivated, not thoroughly studied, or cover only one particular factor graph. For a security evaluator, it is hence not easy to draw more general conclusions about the performance of BP from these works.

Our Contributions. We present an analysis of exact and approximate algorithms to solve inference problems in the context of the side-channel analysis of a cryptographic computation. First, we present the application of exact inference techniques towards SASCAs and analyze its limitations. Then, turning to BP, we provide an analysis of failure cases of this algorithm when used in a SASCA. This analysis provides a basis for explainability which otherwise is lacking in literature regarding profiled attacks. Following, we analyze several heuristic tweaks to BP that have been proposed in the side-channel literature over the last decade. We perform numerical experiments using factor graphs induced by various parts of cryptographic algorithms, assuming that all intermediate variables leak the same amount of information. We apply the techniques of selected works on the most relevant targets for SASCAs, and compare their results to exact inference to provide the basis for a security bound. Our results indicate that most proposed BP heuristics have a minimal impact on the attack performance in general, with exceptions for specific combinations of heuristics, circuit and experimental conditions, or for a heuristic with high computational cost. Finally, we discuss the impact of our findings on security evaluations.

2 The Side-Channel Inference Problem

Performing a key recovery is the most common goal of a side-channel adversary. In this setting, the unknown key $\boldsymbol{K} = (K_1, \ldots, K_\kappa)$[3] (we denote random variables by capital letters and vectors in bold) is the target of the adversary who observes the leakages $\boldsymbol{L}$. If the adversary has the computational power to enumerate n key candidates, the side-channel attack problem can be expressed as a generalized *maximum a posteriori* (MAP) estimate from Bayesian statistics: find the n most probable keys given the leakage. In other words, given side-channel leakage traces $\boldsymbol{\ell}$, what are the n most probable secret keys in the key space $\mathcal{K}$?

$$\boldsymbol{k}^1, \ldots, \boldsymbol{k}^\eta = \underset{(\text{top } \eta)\ \boldsymbol{k}}{\arg\max} \Pr[\boldsymbol{K} = \boldsymbol{k} \mid \boldsymbol{L} = \boldsymbol{\ell}] \tag{1}$$

A common assumption in side-channel attacks and security analysis is the independence of leakages [BS21; DDF19; BCS21; BCM+23]: given a series of intermediate variables $\boldsymbol{V} = (V_1, \ldots, V_n)$ in the leaking computation (including the key itself: $V_i = K_i$ for $i = 1, \ldots, \kappa$), it is assumed that $\boldsymbol{L} = (L_1, \ldots, L_n)$, where each L_i is a noisy function of V_i: $L_i = l_i(V_i)$ whose randomness is fresh (in particular, it is independent of the other intermediate variables' leakage). This implies the following Markov chains:

$$\boldsymbol{K} \to V_i \to L_i \quad \text{for } i = 1, \ldots, n$$

[3] $\boldsymbol{K}$ may also be any value of interest to break the cryptographic scheme, not necessarily a long-term key.

such that we can write

$$\Pr[\boldsymbol{L} = \boldsymbol{\ell} \mid \boldsymbol{K} = \boldsymbol{k}] = \sum_{v} \prod_{i=1}^{n} \Pr[L_i = \ell_i \mid V_i = v_i] \cdot \Pr[\boldsymbol{V} = \boldsymbol{v} \mid \boldsymbol{K} = \boldsymbol{k}]. \tag{2}$$

In practice, profiling adversaries build models to estimate $\Pr[V_i = v_i \mid L_i = \ell_i]$ (e.g., using Gaussian templates). Further, $\Pr[\boldsymbol{V} = \boldsymbol{v} \mid \boldsymbol{K} = \boldsymbol{k}]$ only depends on the cryptographic algorithm implementation, which is supposed to be known to the adversary.

The *side-channel inference problem* consists in solving Equation 1 under the assumption that $\Pr[\boldsymbol{K} \mid \boldsymbol{L}]$ satisfies Equation 2. In this work, algorithms that solve this problem are named *exact inference methods*. We also investigate *approximate*, or *inexact* inference methods: heuristic algorithms that aim at producing results close to the answer to the inference problem, while being more efficient than exact inference methods.

Attacks that solve the side-channel inference problem are known as Soft-Analytical Side-Channel Attacks (SASCA) since they exploit soft information on the intermediate variables ($\Pr[V_i = v_i \mid L_i = \ell_i]$) and the analytical knowledge of the computations (which implies the relationships between the intermediate variables and the key, $\Pr[\boldsymbol{V} = \boldsymbol{v} \mid \boldsymbol{K} = \boldsymbol{k}]$).

2.1 Side-channel inference problem and probabilistic graphical models

We identify dependencies between the variables by computing the set of parents of a variable: for a variable V_i which is the result of an operation on other intermediate variables, we denote by $P(i)$ the set of indices j such that V_j is an operand of the operation, and by $V_{P(i)}$ the set $V_j : j \in P(i)$. For example, if V_3 is computed as $V_3 = V_1 + V_2$, then $P(3) = 1, 2$ and $V_{P(3)} = V_1, V_2$. Some variables are not the result of an operation: those are the fresh randomness and the parts of the key ($K_i = V_i$, for $i = 1, \ldots, \kappa$). For all these variables V_i, $P(i) = \emptyset$. These dependencies imply:

$$\Pr[\boldsymbol{V}] = \prod_{i=1}^{n} \Pr[V_i \mid V_{P(i)}]. \tag{3}$$

The relationships $V_i \mid V_{P(i)}$ induce a directed acyclic graph (DAG) structure: Since these dependencies come from the way the variables V_i are originally computed, there cannot be any cyclic dependency.

This structure of relationships fits probabilistic graphical models (PGMs) such as *Bayesian networks* and *factor graphs.*

Bayesian networks. Such a structure of dependencies between random variables is a Bayesian network. Formally, a Bayesian network $\mathcal{B}$ is a pair $\mathcal{B} = (\mathcal{G}, \Phi)$ over a set of nodes representing random variables $\mathcal{X}$ that models probabilistic relationships between the variables such that ([Dar09, Definition 4.1] and [KF09, Theorem 9.5])

- $\mathcal{G}$ is a Directed Acyclic Graph (DAG) with nodes $\mathcal{X}$.
- Φ is the set of conditional probability factors between variables in $\mathcal{X}$ and their parent nodes in the graph:

$$\Phi = \{\phi_{X_i}\}_{X_i \in \mathcal{X}}, \quad \phi_{X_i} = \Pr[X_i \mid X_{\mathsf{Parents}(i)}]$$

In a Bayesian network, the joint probability distribution can be factorized as

$$\Pr[\boldsymbol{X} = \boldsymbol{x}] = \prod_{X_i \in \mathcal{X}} \Pr[X_i = x_i \mid X_{\mathsf{Parents}(i)} = x_{\mathsf{Parents}(i)}].$$

Algorithm 1 The first two operations in the AES

```
Require: $K_0$, $P_0$                                  ▷ Key, Plaintext
    $X_0 \leftarrow K_0 \oplus P_0$                    ▷ AddRoundKey
    $Y_0 \leftarrow \textsc{Sbox}(X_0)$                ▷ SubBytes
```

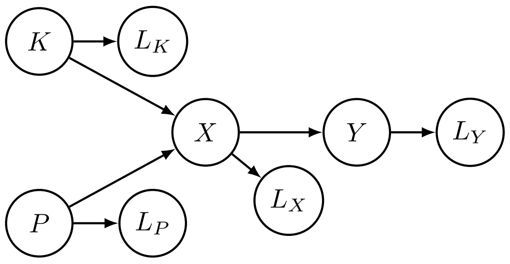


Figure 1: Bayesian network modeling the relationships between variables in Algorithm 1.

Note that the Bayesian network is not limited to modeling the relationships between the variables in V, it can also contain the variables L_i, where each L_i is only connected to its parent V_i.

Example 1. Consider the first two operations of the AES round (Algorithm 1). The intermediate variables are $\boldsymbol{V} = (K, P, X, Y)$, with the corresponding leakages being $\boldsymbol{L} = (l_k(K), l_p(P), l_x(X), l_y(Y))$ (in practice, the plaintext P is often known to the adversary, which is modeled with $l_p(P) = P$). The relationships between the variables are

$$X = P \oplus K$$
$$Y = \textsc{Sbox}(X)$$

which, in addition to the leakage, constitute the Bayesian network shown in Figure 1.

Factor graphs. A factor graph is a model that generalizes Bayesian networks by removing the acyclic structure of dependencies. It represents an arbitrary factorization of a joint probability with a bipartite graph. Given a function P^*[4] defined over a set of n variables, $\mathbf{x} \equiv \{x_i\}_{i=1,\dots,n}$, which is the product of m factors:

$$P^*(\mathbf{x}) = \prod_{j=1}^{m} f_j(\mathbf{x}_{I(j)}) \tag{4}$$

The corresponding factor graph $\mathcal{G} = (\mathbf{x}, \mathbf{f}, \mathbf{e})$ consists of variable nodes $\mathbf{x}$, factor nodes $\mathbf{f} \equiv \{f_j\}_{j=1}^{m}$ and undirected edges $\mathbf{e}$ which join all variables in $\mathbf{x}_{I(j)}$ ($\mathbf{x}_{I(j)}$ is a subset of the variables in $\mathbf{x}$) with f_j. Each factor is a function $f_j : [0,1]^{n_j} \to [0,1]$. Any Bayesian network can therefore be re-written as a factor graph by turning each conditional probability into a factor.

Remark 1. In the context of PGMs, factors represent the actual probabilistic relationship between conditionally dependent variables. Intuitively, factors represent the likelihood of each combination of values for a subset of variables. A particular case is the *deterministic factor* (by opposition to probabilistic factors), which always evaluates to 0 or 1, representing *impossible* or *certain* variable combinations. In Bayesian networks, a conditional probability

[4] P^* is not necessarily a probability distribution since it may not sum to 1. This issue can be solved by considering $P'(x) = P^*(x)/\sum_{x'} P^*(x')$.

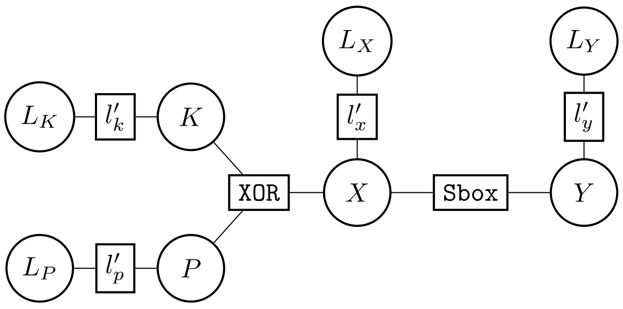


Figure 2: Factor graph representation of Algorithm 1. Variables and factors are represented with circle and rectangular-shaped nodes, respectively.

distribution ϕ_{X_i} is deterministic when there exists a (deterministic) function f_i such that $X_i = f_i(X_{\textsc{Parents(i)}})$. For factor graph, a factor f_j is deterministic if its image is a subset of $\{0,1\}$. Moreover, when converting a Bayesian network to a factor graph, a deterministic probability distribution translates into a deterministic factor which is sparse: if the distribution of the variable depends on n_i variables, then the factor description contains $|\mathbb{K}|^{n_i+1}$ entries, of which only $|\mathbb{K}|^{n_i}$ are non-null (where $\mathbb{K}$ is the field for the variables).

The AES scenario from Example 1 can be modeled via a factor graph (Figure 2). The `XOR` and `Sbox` factors are

$$\texttt{XOR(k,p,x)} = \begin{cases} 1 & x = k \oplus p \\ 0 & \text{otherwise} \end{cases}$$

$$\texttt{SBOX(y,x)} = \begin{cases} 1 & y = \texttt{Sbox}(x) \\ 0 & \text{otherwise.} \end{cases}$$

while the l'_x factor corresponds to the joint (X, L_x) distribution (and likewise for l'_y, l'_k and l'_p).

The interest of using a factor graph over a Bayesian network does not appear in our example, and in fact Bayesian network may seem sufficient to model all side-channel inference problems, since the intrinsic ordering in a computation implies a DAG structure. However, it is sometimes handy to insert additional variables and knowledge in the problem. For example let us consider a masked circuit where the input is a sharing $(K_1, \ldots, K_d)$ such that the secret K does not appear in the computations, but satisfies the relationship $K = K_1 \oplus \cdots \oplus K_d$. This can be easily represented in a factor graph, while encoding this in a Bayesian network, while feasible, is less natural. Factor graphs may become the only option in more involved situations e.g., if we want to incorporate knowledge of a sharing $X = X_1 \oplus \cdots \oplus X_d$ for intermediate variables X_i which already have conditional probability factors.

2.2 Inference Methods

In this section, we discuss exact and approximate inference methods for solving SASCA problems.

2.2.1 Exact inference methods

Brute-force The simplest approach to solving the inference problem is to enumerate all possible values for $\boldsymbol{V}$. Indeed, in order to solve Equation 1 for a given $\boldsymbol{L} = \boldsymbol{\ell}$, it suffices to compute $\Pr[\boldsymbol{L} = \boldsymbol{\ell} \mid \boldsymbol{K} = \boldsymbol{k}]$ for all $\boldsymbol{k} \in \mathcal{K}$, thanks to Bayes rule:

$$\Pr[\boldsymbol{K} = \boldsymbol{k} \mid \boldsymbol{L} = \boldsymbol{\ell}] = \frac{\Pr[\boldsymbol{L} = \boldsymbol{\ell} \mid \boldsymbol{K} = \boldsymbol{k}] \Pr[\boldsymbol{K} = \boldsymbol{k}]}{\sum_{\boldsymbol{k}'} \Pr[\boldsymbol{L} = \boldsymbol{\ell} \mid \boldsymbol{K} = \boldsymbol{k}'] \Pr[\boldsymbol{K} = \boldsymbol{k}']}.$$

Next, using Equation 2 and Equation 3, we get

$$\Pr[\boldsymbol{L} = \boldsymbol{\ell} \mid \boldsymbol{K} = \boldsymbol{k}] = \sum_{v \text{ s.t.} (v_1,\ldots,v_\kappa)=k} \prod_{i=1}^{n} \Pr[L_i = \ell_i \mid V_i = v_i] \prod_{i=\kappa+1}^{n} \Pr[V_i = v_i \mid V_{P(i)} = v_{P(i)}].$$

Finally, let us assume that there are ρ fresh uniform random variables in the circuit: $(V_{\kappa+1}, \ldots, V_{\kappa+\rho}) = R$, then all other variables in $\boldsymbol{V}$ are fully determined by $\boldsymbol{K}$ and $\boldsymbol{R}$: $\boldsymbol{V} = c(\boldsymbol{K}, \boldsymbol{R})$, which means:

$$\Pr[\boldsymbol{L} = \boldsymbol{\ell} \mid \boldsymbol{K} = \boldsymbol{k}] = \sum_{\boldsymbol{r}} \prod_{i=1}^{n} \Pr[L_i = \ell_i \mid V_i = c_i(\boldsymbol{k}, \boldsymbol{r})] \Pr[\boldsymbol{R} = \boldsymbol{r}].$$

The computational cost of running the brute-force is thus $O(n\,|\mathbb{K}|^{\kappa+\rho})$, where $\mathbb{K}$ is the field in which the inference is computed.

Here, we assumed that the inference problem can be written as a Bayesian network. The brute-force method also works with a general factor graph, with conditional probabilities replaced by factors, although the last step (eliminating all variables except $\boldsymbol{K}$ and $\boldsymbol{R}$ from the enumeration) cannot be systematically performed.

Variable elimination. We give a brief informal overview of variable elimination and refer the reader to [KF09] (or other reference works on probabilistic inference) for in-depth explanations.

The variable elimination (VE) algorithm works on factor graphs (therefore, it also applies to Bayesian networks). Given a partition $(\mathcal{Y}, \mathcal{Z})$ of its set of variables $\mathcal{X}$ and a value $\mathbf{x}_\mathcal{Y}$ for the variables in $\mathcal{Y}$, it computes *marginalization* queries of the form $\sum_{\mathbf{x}_\mathcal{Z}} P^*(\mathbf{x}_\mathcal{Y}, \mathbf{x}_\mathcal{Z})$, next denoted $P^*(X_\mathcal{Y} = \mathbf{x}_\mathcal{Y})$.

In order to solve Equation 1, we can compute

$$\Pr[\boldsymbol{K} = \boldsymbol{k} \mid \boldsymbol{L} = \boldsymbol{\ell}] = \frac{\Pr[(\boldsymbol{K}, \boldsymbol{L}) = (\boldsymbol{k}, \boldsymbol{\ell})]}{\Pr[\boldsymbol{L} = \boldsymbol{\ell}]}, \tag{5}$$

but, since $\Pr[\boldsymbol{L} = \boldsymbol{\ell}]$ does not depend on $\boldsymbol{k}$, we only need $\Pr[(\boldsymbol{K}, \boldsymbol{L}) = (\boldsymbol{k}, \boldsymbol{\ell})]$. Let us assume that we have a factor graph whose variables are $(\boldsymbol{V}, \boldsymbol{L})$ and whose function P^* corresponds to the joint probability of $(\boldsymbol{V}, \boldsymbol{L})$. Further, let $\boldsymbol{U}$ be the non-key intermediate variables such that $V = (\boldsymbol{K}, \boldsymbol{U})$. We then have

$$\Pr[(\boldsymbol{K}, \boldsymbol{L}) = (\boldsymbol{k}, \boldsymbol{\ell})] = \sum_{\boldsymbol{u}} \Pr[(\boldsymbol{K}, \boldsymbol{U}, \boldsymbol{L}) = (\boldsymbol{k}, \boldsymbol{u}, \boldsymbol{\ell})] = \sum_{\boldsymbol{u}} \prod_{j=1}^{m} f_j((\boldsymbol{k}, \boldsymbol{u}, \boldsymbol{\ell})_{I(j)}).$$

Naively computing this sum corresponds to the brute-force method, but the variable elimination algorithm can be more efficient. Let us assume without loss of generality that the variable u_1 appears only in a subset of the factors $\boldsymbol{J}$, we can rewrite:

$$\Pr[(\boldsymbol{K}, \boldsymbol{L}) = (\boldsymbol{k}, \boldsymbol{\ell})] = \sum_{u_2, u_3, \ldots} \prod_{j \in \bar{\boldsymbol{J}}} f_j((\boldsymbol{k}, \boldsymbol{u}, \boldsymbol{\ell})_{I(j)}) \left(\sum_{u_1} \prod_{j \in \boldsymbol{J}} f_j((\boldsymbol{k}, \boldsymbol{u}, \boldsymbol{\ell})_{I(j)}) \right)$$

with $\bar{\boldsymbol{J}} = \{1, \ldots, m\} \setminus \boldsymbol{J}$. This transformation, named “eliminating u_1” removes one variable from the left-most sum, merging all factors adjacent to u_1 into a single factor by multiplying them and summing-out u_1. The computational cost is now the total of (i) computing that merged factor and (ii) computing the leftmost sum. The latter cost is reduced by a factor $|\mathbb{K}|$, while the cost of computing the merged factor is exponential in the number of variables involved in that factor (there is a gain as long as the merged factor is not connected to all u_i variables).

In general, the variable elimination algorithm iteratively eliminates all the variable, in a well-chosen order. It therefore essentially amounts to using distributivity of the multiplication to reduce the number of computations to perform, by multiplying factors and summing-out variables. The computational cost depends exponentially on the maximum scope size, i.e. the maximum number of variable adjacent to any single merged factor that gets computed. This important value is lower-bounded by the treewidth of the factor graph[5], and depends crucially on the variable elimination order (finding a good elimination order is a difficult problem).

Going back to our AES example, VE may perform the following computation:

$$\begin{aligned}\Pr[(\boldsymbol{K},\boldsymbol{L}) = (\boldsymbol{k},\boldsymbol{\ell})] = \Pr[(K, L_k) = (k, \ell_k)] \cdot \sum_x \Bigg(& \\ & \Pr[(X, L_x) = (x, \ell_x)] \\ & \cdot \left(\sum_p \Pr[(P, L_p) = (p, \ell_p)] \cdot \mathtt{XOR}(k, p, x)\right) \\ & \cdot \left(\sum_y \mathtt{Sbox}(y, x) \cdot \Pr[(Y, L_y) = (y, \ell_y)]\right) \\ \Bigg) &\end{aligned}$$

where y, p and x have been successively eliminated.

While VE can be more efficient than brute-force, many factor graphs that are relevant for SASCAs have high treewidth and therefore make VE impractical. Given the high treewidth of cryptographic circuits relative to the number of key (and randomness) variables, VE is often not more efficient than brute-force, and further, given the exponential computation cost of VE, it is actually practical to run VE in few of those cases. There exists other inference methods (e.g., probabilistic circuits [WNC+24]), but they suffer from similar issues: there are very few circuits for which they are practical to run while performing better than brute-force.

2.2.2 Approximate inference: belief propagation

The belief propagation (BP) algorithm of Pearl [Pea82] efficiently computes the marginalization of a function given its factorization. The algorithm takes a message-passing approach in which a node's state is iteratively updated based on "messages" (often referred to as "beliefs") from adjacent nodes until reaching convergence. BP can be applied to any PGM, like Bayesian networks, by first converting it into a factor graph representation. For tree-like factor graphs, BP gives a correct solution and is therefore an exact inference algorithm equivalent to VE with a good ordering (which is easy to find in this case). In contrast, if the factor graph contains cycles, then the heuristically guided *loopy* variant of the algorithm must be used and the result is an approximation of the marginalization query. We give an overview of applying BP to SASCA problems and refer the reader to the works [Mac03, Chapter 26] and [VGS14] for a more detailed description.

[5]Informally, the treewidth is an integer which quantifies how "tree-like" a graph is. The only graphs having a treewidth of 1 are trees (while graphs with a single cycle have treewidth 2).

In more detail, given a factor graph and one of its variable nodes X_i, the BP algorithm computes or approximates the marginalization $P^*(X_i = x_i)$, i.e., it computes the marginal for a single variable. Therefore, in order to compute $\Pr[\boldsymbol{K} = \boldsymbol{k} \mid \boldsymbol{L} = \boldsymbol{\ell}]$ in Equation 1, we first condition the factor graph by adding new factor to the graph to model the (hard) evidence $\boldsymbol{L} = \boldsymbol{\ell}$ (we connect a new factor to each variable node L_i, encoding the fact $L_i = \ell_i$). Then, we only need to query $\Pr[\boldsymbol{K} = \boldsymbol{k}]$ on this modified graph. If the key contains multiple variables, then we actually query $\Pr[K_i = k_i]$ for all key variables, then assume that the distribution of $\boldsymbol{K}$ is the product of independent distributions K_i.[6]

BP works by exchanging beliefs along the edges of the graph, where each belief is a distribution over the value of the adjacent variable node. Beliefs from variables to factors (V2F) are denoted as $\mu^{X_i \to f_j}$ and beliefs from factors to variables (F2V) are denoted as $\mu^{f_j \to X_i}$). The beliefs are iteratively updated, according to the following rules. The belief from a variable to a factor is the product of the beliefs from the other factors to the variable, which corresponds to combining independent probability distributions:

$$\mu^{X_i \to f_j} = \prod_{\{j' \mid i \in I(j') \wedge j \neq j'\}} \mu^{f_{j'} \to X_i}. \tag{6}$$

Regarding the F2V beliefs, for a factor f_j and a variable X_i, $\mu^{f_j \to X_i}$ is computed as the marginalization of the factor f_j taking into account independent marginals from $\mu^{X_{i'} \to f_j}$ for variables $X_{i'}$ adjacent to f_j (but distinct from X_i):

$$\mu_\alpha^{f_j \to X_i} = \sum_{\{\mathbf{x}_{I(j)} \mid \mathbf{x}_i = \alpha\}} f_j(\mathbf{x}_{I(j)}) \prod_{i' \in I(j) \setminus \{i\}} \mu_{\mathbf{x}_i}^{X_{i'} \to f_j}. \tag{7}$$

Finally, the marginal for a variable $P^*(X_i = x_i)$ is computed as the product of all incoming messages:

$$P^*(X_i = x_i) = \prod_{\{j \mid i \in I(j)\}} \mu_{x_i}^{f_j \to X_i}. \tag{8}$$

An important consideration in the usage of the BP algorithm is the *scheduling*, i.e., the order in which Equation 6 and Equation 7 are applied to the different factors and variables. For tree-like structures, all[7] schedulings converge to the same solution, resulting in an exact inference. However, for loopy factor graphs, the scheduling impacts the final result, as well as whether the algorithm converges. As a baseline, we consider the simple *parallel update* scheduling: an iterative process in which each iteration is made of two steps: first, all F2V beliefs in the graph are updated using Equation 7, then all V2F beliefs are updated with Equation 6. The V2F beliefs are initialized with uniform distributions.

Let us illustrate the BP algorithm by applying it to the AES example (Example 1 and Figure 2). Since the graph is a tree, we use an optimal scheduling (thanks to a well-chosen "leaf-to-root" order, we compute each belief twice):

1. Observed leakage values ℓ_v are propagated from the leakage variables $L_v \in (L_k, L_p, L_x, L_p)$ to the respective factors:
$$\mu^{L_v \to l'_v} = \delta_{\ell_v}$$
where δ_x is a discrete Dirac distribution: it has probability 1 at x and 0 elsewhere.

2. Beliefs from l'_v are propagated to v:
$$\mu_\alpha^{l'_v \to v} = \sum_\ell l'_v(\alpha, \ell) \mu_\ell^{L_v \to l'_v} = l'_v(\alpha, \ell_v)$$

[6]This technique is also often used with exact inference methods, when it is infeasible to compute the key distribution due to a too large key space $\mathcal{K}$.

[7]As long as they satisfy some minimal constraints to not ignore some variables or factors.

3. The belief from Y to the Sbox factor, is computed, then the belief from the Sbox to X:

$$\begin{aligned}\mu^{Y\to\textsc{Sbox}} &= \mu^{l'_Y\to Y}\\ \mu_x^{\textsc{Sbox}\to X} &= \sum_y \textsc{Sbox}(x,y)\cdot\mu_y^{Y\to\textsc{Sbox}}\end{aligned}$$

4. Next, the beliefs from X and P to the Xor factor are computed, which allows computing the belief from that factor to K:

$$\begin{aligned}\mu^{P\to\textsc{Xor}} &= \mu^{l'_P\to P}\\ \mu^{X\to\textsc{Xor}} &= \mu^{l'_X\to X}\cdot\mu^{\textsc{Sbox}\to X}\\ \mu_k^{\textsc{Xor}\to K} &= \sum_{x,p}\textsc{Xor}(k,p,x)\cdot\mu_p^{P\to\textsc{Xor}}\cdot\mu_x^{X\to\textsc{Xor}}\end{aligned}$$

5. Finally, the marginal for K is

$$P^*(K=k) = \mu_k^{\textsc{Xor}\to K}\cdot\mu_k^{l'_K\to K}.$$

While it is not guaranteed that the loopy BP algorithm will return correct values or even converge, it gives sufficiently precise approximations of the marginals in many real-world applications, such as the decoding of error correcting codes [KFL01]. For cryptographic circuits, BP can be efficiently executed, with complexity $O(m\,|\mathbb{K}|^{n_i}+n\,|\mathbb{K}|)$ per iteration when working in a field $\mathbb{K}$ with deterministic operations having at most n_i inputs (the factors have n_i+1 adjacent variables, but we can exploit their sparsity to optimize computations).[8] For many SASCA factor graphs, BP is the only practical algorithm from a performance point of view, this algorithm is therefore the de-facto standard when for conducting SASCAs.

2.3 Metrics for Comparing SASCAs

The stochastic nature of side-channel analysis leads to a zoo of metrics which attempt to characterize the success (or failure) of a given process. Particularly in the case of SASCAs, it is a challenging task to find an adequate metric to describe the results of many experiments. We briefly describe the popular metrics used in the literature and motivate our choices for the metrics we report in the next sections.

Rank and Guessing Entropy. The rank of a key candidate is its position in the solution generated by Equation 1 over the key space, K. In our semantics, $\mathsf{rank}_k=1$ means that the correct key is at the first member in the list of solutions. Often times, we are interested if the rank of the correct key is less than n, where a brute-force test of the solution becomes feasible (roughly 2^{64} for a determined adversary). For large key spaces, a rank estimation algorithm [PSG16; CDS23] can be used. It is often convenient to report the log of the rank e.g., the *Guessing Entropy*: $\mathsf{GE}=\log_2(\mathsf{rank}_k)$.

Success rate. The success rate (SR) is the proportion of key ranks which are equal to 1 in the distribution of results. The generalized SR at rank r is the proportion of key ranks less than or equal to an order r in the distribution of results.

[8] We can even achieve better complexity for some particular factors, e.g. for the addition, an algorithm based on a Fourier transform achieves complexity in $|\mathbb{K}|\log|\mathbb{K}|$ [CDSU23].

Mutual Information and Perceived Information. The Mutual Information (MI) quantifies the information which can be extracted between a variable X and the leakage L:

$$\mathsf{MI}(X;L) = \mathsf{H}[X] + \sum_{x\in\mathcal{X}} \Pr(x) \sum_{l\in\mathcal{L}} \Pr(l \mid x) \cdot \log_2 \Pr(x \mid l)$$

In practice, the leakage distribution $\mathcal{L}$ is unknown, therefore the quantity $\Pr(l \mid x)$ is replaced by one generated by a leakage model, $\mathcal{L}_m$. If the chosen model differs from the actual leakage distribution, the MI cannot be computed. Instead, the PI can be captured:

$$\mathsf{PI}(X;L) = \mathsf{H}[X] + \sum_{x\in\mathcal{X}} \Pr(x) \sum_{l\in\mathcal{L}_m} \Pr(l \mid x) \cdot \log_2 \Pr(x \mid l)$$

The PI is a lower bound of the MI [BHM+19] that converges to the MI if the model can match the true leakage distribution [MCHS23].

Choice of reported metrics. The distribution of multiple SASCA experiments can be multimodal, especially for larger bit widths: the attack may sometimes succeed and sometimes completely fail, with a low probability of an intermediate outcome. Therefore, extreme care must be taken when selecting the metrics for comparison as not to obfuscate any meaningful observations. In our experiments in the next sections, we chose to report only the average guessing entropy, as it appears to be the most relevant and reliable metric in our case. The SR for $\mathsf{rank} = 1$ was determined to be too conservative, with a very sharp drop off as the signal-to-noise ratio (SNR) decreases. For $\mathsf{rank} > 1$, it is difficult to determine a suitable rank such that the SR is meaningful for our factor graphs, which vary in bit width for different cases. On the other hand, the PI was omitted due to its difficult interpretation: even when an attack succeeds (i.e., low guessing entropy), the PI can be low if the key probability distribution estimated by the attack does not correspond to the correct one, as discussed in Section 3.4.

3 How Belief Propagation Fails in SASCAs

As discussed in Section 2, loopy belief propagation is the most widespread inference method for SASCA. Unfortunately, while BP may work well, it also sometimes fails spectacularly, e.g. by not converging, or converging to a value that is clearly wrong. In general, the behavior of BP is not well understood and general guarantees on convergence or accuracy are unavailable. Although studying the message dynamics [MK05; KP17; LKSP21] has significantly improved the understanding the convergence properties, sufficient conditions for convergence remain unattainable except for specific assumptions (e.g., for Gaussian models [SW15], models with a single loop [Wei00], or models with a unique fixed point [MK07; IIW05]). Research on BP with abstracted factor graphs (e.g., binary Markov Random Fields (MRFs) with pairwise factors) captures a wide variety of problems and use-cases. However, these graphs are not typically a good match for SASCA problems: cryptographic factor graphs in particular have several features which do not lend itself to be modeled by e.g., binary MRFs. For instance, the SASCA typically exploits leakages from real-world devices, whose variable domains are determined by the word size of the processor linked by deterministic functions involving two or more inputs. Thus, it is difficult to leverage existing theoretical results on BP for the SASCA.

In this section, we discuss concrete situations on relevant cryptographic factor graphs where BP either fails or produces suboptimal results. In particular and in order of severity, we study when BP converges to the wrong marginals with high confidence, when BP can oscillate between two fixed states, when BP is “blinded” by certain functions and when BP is overconfident. These examples illustrate some pitfalls we should care about when using SASCA.

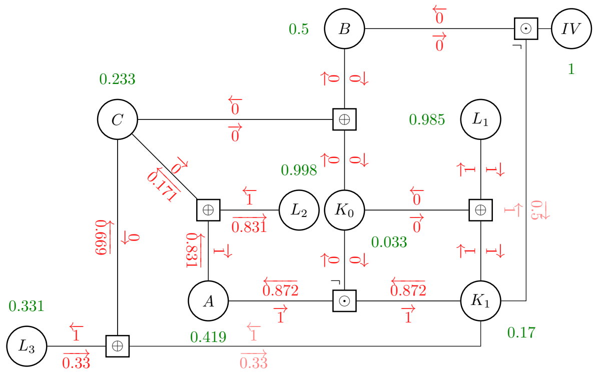


Figure 3: ISAP Factor Graph with annotated expectations of beliefs. Green indicates evidence on a variable and red indicates amortized beliefs along an edge. The numbers indicate the probability of the value being equal to 1.

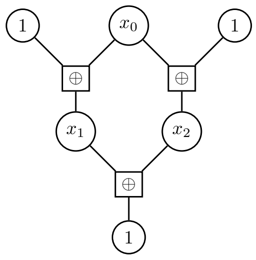


Figure 4: Frustrated XOR factor graph example.

3.1 Converging to the Wrong Value

It is well understood that, in general, BP can converge to incorrect marginals with high confidence. In the context of SASCA, an overconfident BP algorithm would converge towards assigning a high likelihood to the wrong key. As an example of this situation, let us consider the ISAP-RK [DEM+20] factor graph in Figure 3. With the given evidence, BP is tasked with finding the marginals of K_0 and K_1, whose respective true values are 1 and 0. Due to the (highly noisy) evidence on K_0, the posterior marginal on $\Pr[K_0]$ converges to the incorrect value, 0. With more BP iterations, the uncertainty on the joint distribution of K_0 and K_1 converges to 0 as well, however, the guessing entropy remains high. If the correct values for K_0 and K_1 were unknown, one would conclude that $K_0 = 0$ incorrectly. In contrast, an enumeration solution (i.e., exact inference) on the same setting results in a lower guessing entropy, but with higher uncertainty at 0.36-bits. In other words, the exact method is able to solve the SASCA, but does recognize the uncertainty in the solution, whereas BP converges to a wrong solution, and assigns it a high confidence.

3.2 Oscillations in Marginals

Consider the “frustrated XOR” example in Figure 4 which depicts a factor graph over three binary variables, x_0, x_1 and x_2. All nodes marked with 1 are variables with observations assumed to be certain ($\Pr[x_n = 1] = 1$), and $\oplus$ refers to the XOR operator. While this graph does not admit any solution (the bits x_0, x_1 and x_2 would have to be all distinct from each other), it is a simple illustration of the source of oscillations in the BP algorithm.

The dynamics of BP for this graph is fairly simple: starting from a belief $(p, 1-p)$ from x_0 to the XOR that connects it to x_1, we get a belief of $(1-p, p)$ to x_1, which is

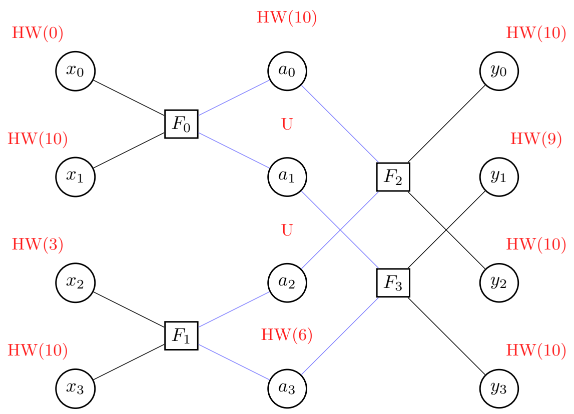


Figure 5: Frustrated NTT example with butterfly nodes. The cycle within this graph is highlighted in blue.

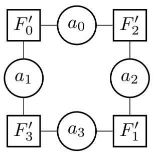


Figure 6: Simplified version of the factor graph given in Figure 5

then forwarded as a belief of $(p, 1-p)$ to x_2 (for the sake of simplicity, we assume that there is no evidence for the variables), which in turn leads to a belief $(1-p, p)$ towards x_0. Since this is the inverse of the belief we started with, and since the belief circles through the graph, with an inversion each time it goes through a factor, it can be the source of oscillation. While we considered only one belief, the dynamics of BP for the graph is actually composed of three beliefs going through this circle independently, and three beliefs circling in the other direction.

In the next example, we show that dynamics similar to the "frustrated XOR" can appear in realistic factor graphs that admit a valid solution. Consider the two layer simplified "butterfly" circuit depicted in Figure 5, where each function F_n is given the same definition:

$$F_n(x_0, x_1, y_0, y_1) = \begin{cases} 1 & \begin{array}{l} y_0 = x_0 + \omega x_1 \bmod q \wedge \\ y_1 = x_0 - \omega x_1 \bmod q \end{array} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

This circuit corresponds to a very small number-theoretic transform (NTT) circuit, and we take all twiddle factors $\omega = 1$ for the sake of simplicity. It is a downsized case study for practically-relevant circuits, since NTTs are often used in lattice-based cryptography [BDK+18]. Again for illustration, we use a small field size of $q = 13$.

As a starting point, evaluate the circuit for input $(x_0, x_1, x_2, x_3) = (0, 10, 3, 10)$ and initialize each variable with evidence corresponding to noise-free leakage of the Hamming weight of the variable's value, denoted $HW(x)$. For two variables (a_1 and a_2), we assume that there is no leakage: we set the leakage to the uniform distribution (denoted by U). For example, $HW(1)$ generates a distribution where all values from $[0, 12]$ with a Hamming weight of 1 are equally likely i.e., $\Pr[X = 1] = \Pr[X = 2] = \Pr[X = 4] = \Pr[X = 8] = \frac{1}{4}$.

Although the butterfly layout of Figure 5 does not make it obvious, the factor graph actually has a fairly simple structure. Indeed, it is equivalent to the graph of Figure 6,

Table 1: Beliefs in the NTT example. For compactness, as multi-sets: the probability of a value is proportional to the number of times it appears in the list (e.g., $0, 0, 1$ indicates $\Pr[x = 0] = \frac{2}{3}$ and $\Pr[x = 1] = \frac{1}{3}$).

Iteration	$a_1 \to F_0$	$a_0 \to F_2$	$a_2 \to F_1$	$a_3 \to F_3$
0	3, 5, 6, 9, 10, 12			
1		3,10		
2			0, 0, 6, 7	
3				6, 12
4	3, 4, 11, 12			
5		9, 10		
6			0, 0, 3, 10	
7				6, 9
8	1, 3, 12			
9		10, 12		
10			0, 0, 6, 7	
11				6, 12

where each F_i' factor corresponds to the F_i factor of the original graph, multiplied with the evidence factors of the adjacent x_j or y_j variables (this simplification does not change the results BP computations on the graph). Therefore, the "circling beliefs" analysis of the frustrated XOR example also apply to the NTT example, this time with 4 variables in the loop instead of 3.

Now, there is evidence in the system, hence the circling beliefs are updated at each step, which leads to multiple possible situations:

- All beliefs converge to the same state, hence BP converges (to a correct state: the "happy" case, or incorrect value: the wrong convergence described in the previous section).
- The beliefs in the two circling directions converge to different states. If these states are incompatible (i.e., there is no value that has non-zero probability in both beliefs), then BP cannot produce a marginalization.[9] This may happen, e.g., if a part (or "circle direction") of a graph converges to the correct value, while another part of the graph converges to a wrong value.
- The circling beliefs do not converge, and instead oscillate. This case happens for the NTT graph with the values given above, as shown in Table 1.

More generally, deterministic factors (such as the XOR, AND or butterfly factors) remove possible solutions from beliefs (i.e., set more values to $\Pr[X = x] = 0$), which has been shown to cause oscillations in other settings [KF09, Chapter 11.3]. Cryptographic factor graphs are particularly susceptible to oscillations since they typically contain many deterministic factors.

3.3 Blinding

BP uses local propagation rules to perform global inference on the graph. While this may work well in some cases (e.g., when the information is contained in the marginals of individual variables), it tends to not work well in more complex cases, where information

[9]In this case, concrete implementations of BP may produce surprising results, depending on how they perform the computations, and there may be a high sensitivity to numerical errors.

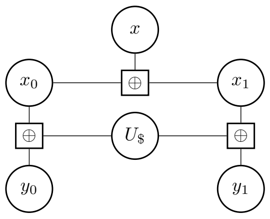


Figure 7: Simplified example where BP is blinded.

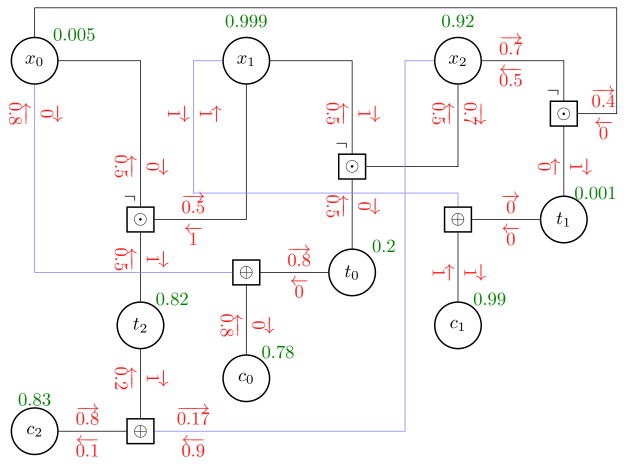


Figure 8: Blinded Keccak example. Green labels indicate evidence and red indicates the amortized beliefs along an edge. Blue edges are used to increase readability. Beliefs are (for space constraints) denoted only by a single number x which is their value on the "1" case (the "0" case can be computed by taking $1 - x$).

lies in the joint distribution of multiple variables. We coin the term "blinding" to designate the situations where this limitation leads to the inability of a SASCA to exploit part of the evidence in the factor graph.

An example of blinding is given in Figure 7, which is a simplified masked circuit. The two shares (x_0, x_1) of a variable x are refreshed with a uniform random bit $U_\$$ to create a new sharing of x: (y_0, y_1). Let us assume that evidence is only supplied for y_0 and y_1, and that this evidence reveals the exact values of these shares. This evidence is enough to recover the value of x since $x = y_0 \oplus y_1$, however BP completely fails and returns a uniform distribution. Indeed, for both XOR factors, two of the incoming beliefs are initially the uniform distribution, making the belief from these factors to the variables uniform. The BP algorithm is therefore stuck in a "uniform beliefs" state.

The blinding issue is not limited to "uniform beliefs" and does not require variables without evidence. We illustrate this in Figure 8, a 3 input Keccak-like S-box. In the example, the evidence is sampled from a normal distribution with $\sigma = 0.29$. The concrete values for the inputs and outputs to the S-box are: $(x_0, x_1, x_2) = (c_0, c_1, c_2) = (1, 1, 1)$. From the noisy sampling, the evidence on x_0 and t_2 are biased incorrectly (i.e., the wrong value is more likely than the correct one) but the rest of the variables are biased correctly.[10] We run BP, then compute a joint distribution for (x_0, x_1, x_2) by multiplying the marginals. This distribution can then be compared to the joint distribution of these variables obtained

[10]While a confidently wrong evidence such as the one we have for x_0 may seem unlikely, such things happen frequently in practice due to the large number of variables in factor graphs, therefore many evidence factors are extracted from the leakage, making "unlucky" events appear frequently (e.g., even our tiny graph has already 9 evidence factors).

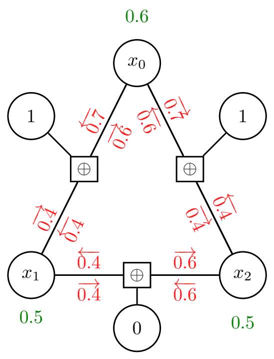


Figure 9: Overconfidence in a simple graph. Green indicates evidence on a variable and red indicates beliefs at the beginning of the fourth iteration.

via exact inference. The guessing entropy of the BP solution is 2 bits, while the guessing entropy of the exact solution is only 1 bit.

3.4 Overconfidence

Overconfidence occurs when BP reaches a solution with low uncertainty when it should not be able to do so, even if this solution is correct. We motivate how this can occur with a simple example.

As an example, consider the graph in Figure 9, which is similar to the graph in Figure 4 except that it has a solution. We indicate the beliefs at the beginning of the fourth iteration in red. Initially, x_0 is the only variable with evidence, $\Pr[x_0 = 1] = 0.6$, with the rest being set to uniform. As the messages iterate over the loop, x_0 ends up receiving information from itself, reinforcing the belief that $x_0 = 1$ incorrectly. Indeed, at the beginning of the fourth iteration, all outgoing beliefs from have $\Pr[x_0 = 1] = 0.7$ which eventually are amortized to $x_0 = 1$. In turn, this causes the values of x_2 and x_3 to also converge. For reference, an inference query using an exact method would not be able to decrease the uncertainty on the marginal distribution $\Pr[x_0 = x]$ beyond what is given in the evidence.

While overconfidence is not a problem when using SASCAs to perform an attack, it can heavily skew the key distribution, leading to an improper estimation of the amount of leakage when computing metrics such as PI. This can be a problem when using SASCAs in leakage assessment tasks. The overconfidence problem appears most strongly with graphs that have many small loops. As discussed in Section 4.2, limiting the number of iterations can mitigate overconfidence and reduce the positive reinforcement.

4 An Empirical Study of the SASCA

Works on SASCAs typically include some attempt to improve the success rate of the attack e.g., by optimizing the factor graph or by applying additional heuristics to the BP algorithm. Over the years, the research community explored several techniques to improve the SASCA with varying degrees of success, however, there is a lack of transparency on why certain heuristics perform better than others, or any generalization of these techniques.

In this section, we study the SASCA in detail and contextualize how well it performs when instantiated with exact inference methods and BP instantiated with different heuristics. First, we introduce the factor graphs we analyze based on relevant cryptographic circuits and the experimental conditions we study. Then, we compare how belief propagation's performance compares to the bound given by exact inference in Section 4.1. Finally,

Table 2: Factor graph parameters

Factor Graph	Treewidth	Diameter	#Factors	Total GE
AES	25	6	128	$8 \log_2 q$
2-Share ISW AES	57	9	468	$8 \log_2 q$
3-Share ISW AES	92	11	1004	$8 \log_2 q$
Ascon	5	4	22	$5 \log_2 q$
2-Share ISW Ascon	10	5	68	$5 \log_2 q$
3-Share ISW Ascon	15	6	149	$5 \log_2 q$
ISAP-RK	3	2	8	$2 \log_2 q$
Kyber NTT 4 layers	14	8	128	$16 \log_2 q$

we conclude the section with a deep dive into several BP heuristics proposed by existing literature and present some novel ones in Section 4.2.

Choice of factor graphs. We reviewed several cryptographic algorithms and narrowed our final selection to four relevant cases for SASCAs: the AES S-box based on Boyar-Peralta's design [BP12], the Ascon S-box [DEMS21], the initialization of the ISAP-RK function [DEM+20] (factor graph from [CDSU23, Figure 1]) and a 4-layer version of the Kyber Number Theoretic Transform (NTT)[BDK+18]. We derived factor graphs from the descriptions of each algorithm, which correlates to how these algorithms are typically implemented. In the base case, we restrict our factor graphs to use only the leakage factors and deterministic factors modeling the Boolean operators: XOR, AND and NOT, and the modular arithmetic operators: ADD, SUB and MUL. Furthermore, each factor graph is assumed to be invariant across each experiment, unless it is modified by an examined heuristic e.g., butterfly factors. Given these graphs, our SASCA aims at recovering the input of the corresponding cryptographic (sub-)circuit. This value recovery is our focus since it is an intermediate step in most side-channel attacks, whether data-assisted (e.g., known plaintext or ciphertext) or not.

In Table 2, we list parameters for each factor graph to characterize their structure. In particular, we report the *treewidth* of a graph, the *diameter*[11] which characterizes the number of BP iterations needed for beliefs to propagate between the two furthest factors, the number of factors within the factor graph and the total GE refers to the number of bits a naive attack would need to guess to solve the inference query.

Due to limitations in the implementation of the rank estimation algorithm used, we were unable to report results for the GE of the full Kyber NTT. Therefore, we report results for a factor graph with only 4 layers (16 inputs). In the context of our experiments, the reduced version serves as a good proxy for how the full NTT would behave and does not change our conclusions. For heuristics on masked circuits, we implemented 2 and 3-share versions of the AES and Ascon factor graphs using ISW multipliers.

Experimental setup and leakage model. The adversarial model in our experiments assumes a worst-case scenario in which a single-trace attacker obtains prior information on every intermediate within a factor graph, including the inputs (e.g., the secret key) and, in masked settings, also on the random bits. For each heuristic, we created an experiment set using the relevant factor graphs and report the guessing entropy (GE) over the SNR, following the discussion from Section 2.3. The prior information for the SASCAs is generated from single-trace leakage simulations. We opted for simulations to increase reproducibility of our results and to allow for more fine-grained control over the

[11]The diameter is the maximum distance between all pairs of factors within a graph. The distance is defined as the number of variable nodes along the shortest path between two factors.

experimental setup. Precisely, we generated leakage traces from the well-known noisy Hamming weight model, where a simulated leakage trace ℓ is generated from the addition of normally distributed noise with zero mean and standard deviation σ to the Hamming weight of an intermediate value $x \in \mathbb{Z}_q$:

$$\ell = \mathrm{HW}(x) + \mathcal{N}(0, \sigma)$$

In the following figures, we report the guessing entropy averaged over 100 independent experiments for each SNR level, assuming for the sake of simplicity that all variables leak with the same SNR. The SNR is defined as:

$$\mathrm{SNR} = \frac{\mathrm{Var}(\{\mathrm{HW}(x) \mid x \in \mathbb{Z}_q\})}{\sigma^2}$$

We perform a logarithmic sweep of σ from 10^{-1} to 10^2 where possible. We report the field size of $\mathbb{Z}_q$ for each experiment as the value varies depending on the computational complexity of the inference algorithm under test. Finally, we denote “baseline” as the results after 50 iterations of an unmodified BP implementation [CB23] in accordance to the description given in Section 2.2.2.

Let us note that the absolute value of the GE is not very important, and we focus on the relative performance between the curves. Indeed, these differences show how the inference method influence the attack result, while the absolute value of the Guessing Entropy can be improved by parameters out of the scope of this study, such as the number of attack traces (we do exclusively single-trace attacks, which maximizes the range of SNR for which interesting results are observed).

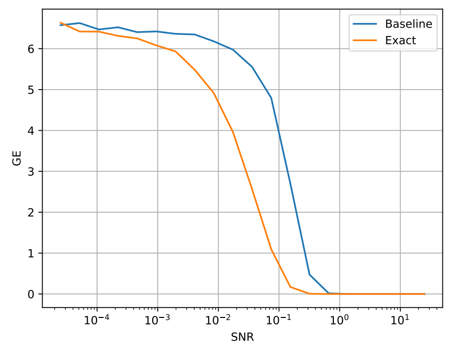
4.1 Comparing Exact and Approximate Inference for SASCAs

We introduce our experiments by first comparing exact vs approximate inference methods for implementing a SASCA to better understand how well BP approximates the correct solution.

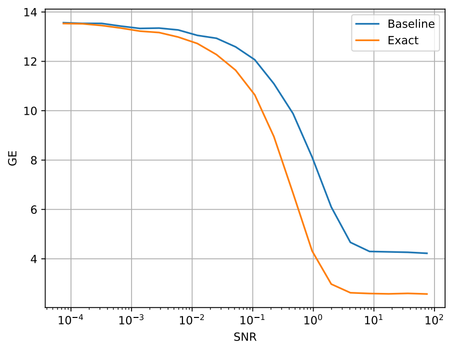
Exact inference vs belief propagation. We compare the results of a SASCA attack using BP and with exact inference methods. The exact SASCA can be understood as the optimal attack given the prior information. In Figure 10, we utilize the brute-force method for exact inference e.g., we exhaustively enumerate all possible states for a given factor graph to find the most likely key, and in Figure 11 we utilize the variable elimination method, both described in Section 2.2.1. For masked circuits, the inclusion of ρ fresh uniform random bits greatly increases the cost of brute-forcing. Since the variables representing these bits are connected to relatively few factors, the variable elimination method is able to more efficiently execute the SASCA by successively eliminating them.

The exact inference curves show that BP is still far from optimal. For high SNR cases, BP and exact inference give very similar results, implying that, for our leakage model, BP is optimal in low noise settings. As the SNR decreases, the results of BP degrade much faster than exact inference. This is consistent with the observation by [MWJ99] and the formalization by [KP17] that increased entropy on the prior information decreases the stability of BP. Exact inference performs better here since it does not suffer from the issues described in Section 3.

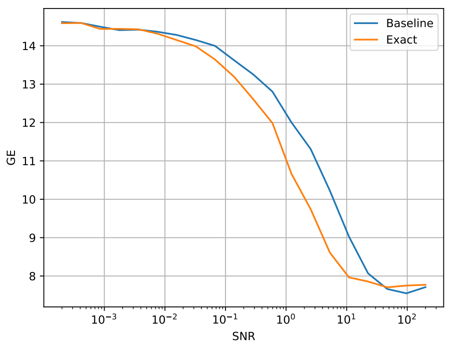
Although the “gap” between exact and approximate inference is not unexpected, from a side-channel perspective it leads to interesting observations: (1) for high and low SNR settings, BP performs about the same as exact inference, and (2) the “middle” SNR range where exact inference performs much better indicates that BP (or other approximate inference algorithms) can be tuned to further improve performance. As we discuss in the next sections, this improvement likely comes with additional computational cost.



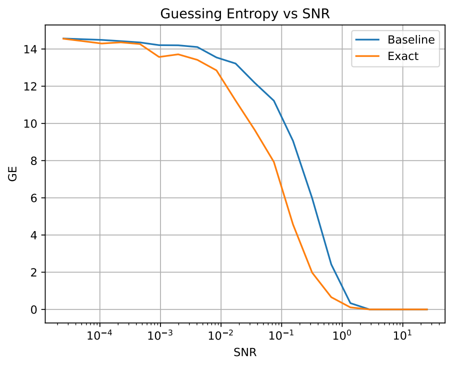
(a) AES - $q = 2$



(b) Ascon - $q = 8$



(c) ISAP-RK - $q = 256$



(d) Kyber NTT - $q = 2$

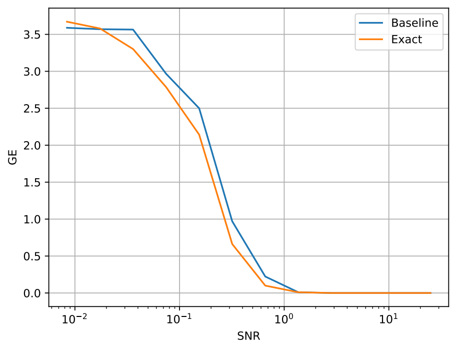
Figure 10: Exact inference vs. belief propagation implementations of the SASCA

4.2 Do Heuristics Improve the SASCA?

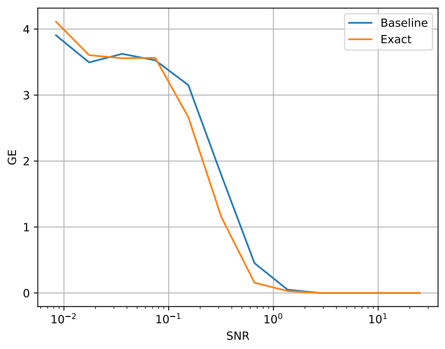
In this section, we review the most relevant BP heuristics suggested in the literature to characterize their effectiveness in carrying out a SASCA. Specifically, we look at varying the number of BP iterations, the message damping technique of [MWJ99], techniques for scheduling BP messages, merging of factor nodes, simplifying the factor graphs of masked circuits and pruning factor graphs under different conditions.

Iteration count. Tweaking the number of BP iterations is a natural first step when optimizing the SASCA. A good baseline is typically chosen to be a multiple of the factor graph's diameter, such that the information from a node passes along all edges. As discussed in Section 3.4, multiple iterations along a loop can compound evidence and lead to overconfidence. In Figure 12, we show the results of running up to 10,000 BP iterations on our graph selection. We see two behaviors: Either convergence to a fixed stable point, such as in Ascon and Kyber, or divergence, such as in AES and ISAP. Note, BP converges within a few iterations – if it does so, running BP for more iterations is not beneficial and one should consequently limit the maximum number of iterations.

Message damping [MWJ99]. Message damping, or simply damping, is a heuristic intended to decrease the likelihood of oscillations (described in Section 3.2) by preserving the "momentum" of message passages. This is achieved by computing a weighted average between the message sent at iteration t and the previously computed message at iteration



(a) 2-share Ascon - $q = 2$



(b) 3-share Ascon - $q = 2$

Figure 11: Belief propagation vs. exact inference for masked Ascon

$t - 1$. The weight for dampening is parameterized by $0.0 < \alpha \leq 1.0$. Concretely, we replace Equation 6 with a dampened version:

$$\mu^{X_i \to f_j(t)} = \alpha \left(\prod_{\{j' | i \in I(j') \wedge j \neq j'\}} \mu^{f_{j'} \to X_i(t)} \right) + (1 - \alpha)\mu^{X_i \to f_j(t-1)}. \tag{10}$$

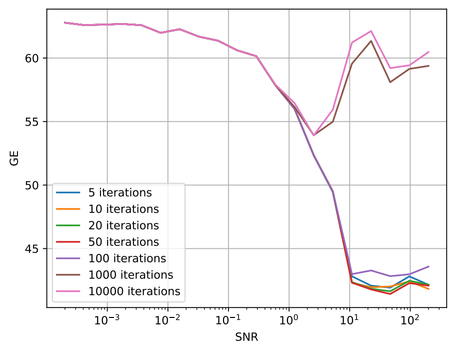
Figure 13 shows the effect of damping for different $\alpha \in [0.01, 0.1, 0.5, 0.9]$. For all factor graphs, damping has no tangible impact on the result of BP.

At first glance, this may appear somewhat surprising, however upon closer inspection, entirely consistent with observations made in SASCA literature. Damping was first utilized in [PP19] among several other optimizations. Though the results of the work yielded an improvement over their previous attack on the Kyber NTT [PPM17], it is unclear if damping had any meaningful effect, especially with their relatively conservative damping factor of $\alpha = 0.9$. It is more likely that the structural changes made to the NTT factor graph via butterfly factors better explain their results, as discussed later in this section. In [KPP20], a damping factor of $\alpha = 0.75$ is used in conjunction with other BP optimizations.

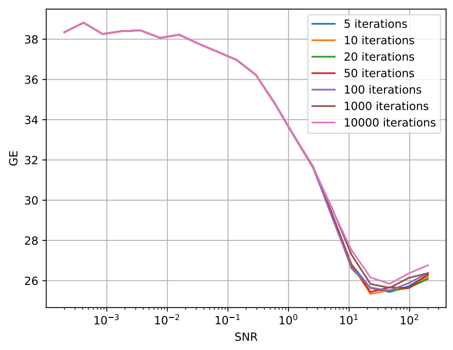
Later works corroborate our findings. In [YK21], the authors found that damping did not provide any consistent improvement to their results: "Finally, after not finding consistent improvements when trying different damping rates, we present our results without damping". In [HHP$^+$21], the authors explicitly mention that damping is omitted, however do not give a reason as to why.

We argue that damping is not advantageous for cryptographic models because of their inherent structure, a stance supported by the empirical observations for highly structured problems (see [PI17]). More specifically, BP exhibits two primary failure modes: the existence of fixed points with subpar approximation quality, and a fragile optimization landscape characterized by poor convergence behavior. Although damping can mitigate convergence problems, theoretical evidence suggests that it does not improve convergence in symmetric models (similar to cryptographic models) that have a bipartite structure [KP17].

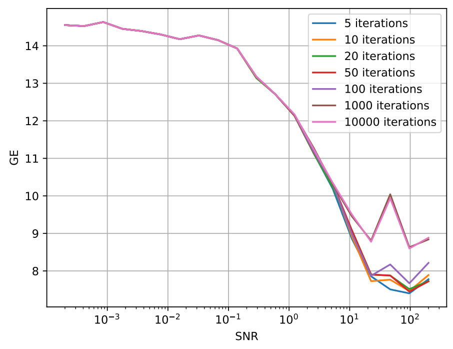
Message scheduling. It is well understood that loops in factor graphs can have an adverse effect on the outcome of BP. Intuitively, a node within a factor graph with cycles may receive partial information from itself after some number of iterations, which can be seen as a sort of positive feedback. Therefore, some works suggest controlling the scheduling of BP messages to minimize the impact of loops, especially small ones. [PP19]



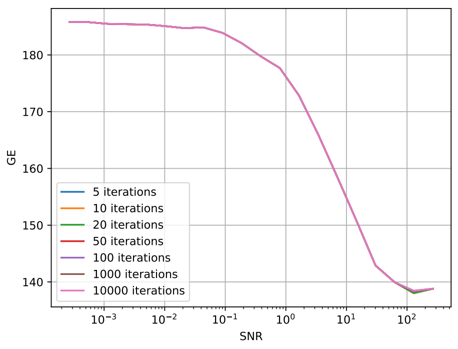
(a) AES - $q = 256$



(b) Ascon - $q = 256$



(c) ISAP-RK - $q = 256$



(d) Kyber NTT - $q = 3329$

Figure 12: Comparison of BP over many iterations

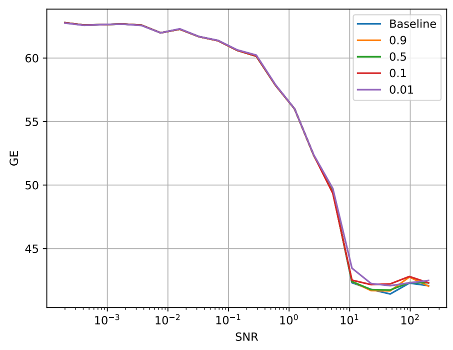
first suggested an alternative message scheduling for SASCA attacks on the Kyber NTT. Messages are propagated from the input layer to the output layer, then back to the input. This forward-backward cycle is then one iteration of BP. This scheduling was applied in subsequent works, such as [KPP20]. We refer to this method of message scheduling as *in-order scheduling*.

The results of BP with in-order scheduling and uncontrolled scheduling for different SNRs can be seen in Figure 14. For AES and Ascon, in-order scheduling performs as well as BP. However, for ISAP-RK and the Kyber NTT graphs, it performs worse. This result is an artifact of the scheduling algorithm; back and circular propagation is blocked; beliefs may never be propagated fully in the case of graphs with low diffusion (ISAP) or graphs with a highly regular structure (Kyber NTT). The diffusion properties of the AES and Ascon S-box allow for all beliefs to propagate fully in both settings.

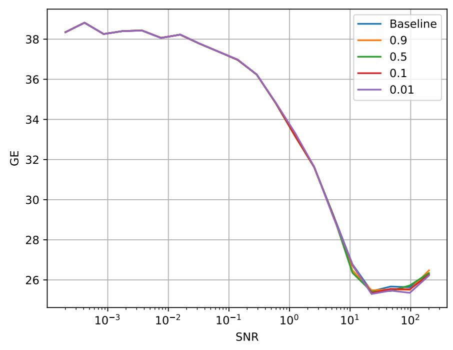
An alternative message scheduling scheme, which has not been applied to SASCAs, is *priority scheduling* [AAK20]. In this method, all F2V messages (Equation 7) are pre-computed at iteration t and stored in a priority queue, Q. Each message in Q is sorted according to a ranking function, $\mathrm{Q}_{\mathrm{rank}}(\cdot)$, which compares the messages at iteration t with the ones from the previous iteration $t-1$:

$$\mathrm{Q}_{\mathrm{rank}}(\mu^{f_j \to X_i(t)}) = \left\| \mu^{f_j \to X_i(t)} - \mu^{f_j \to X_i(t-1)} \right\|, \tag{11}$$

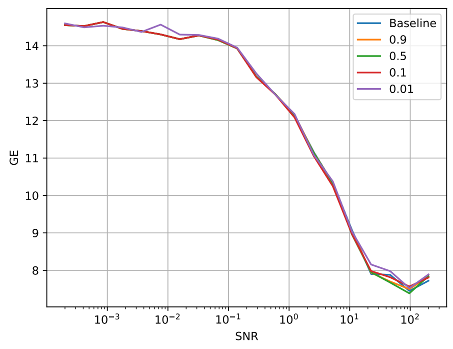
where $\|\cdot\|$ is an arbitrary norm function. After sorting, only the top p% of messages from Q are propagated, with the rest being discarded. The goal of priority scheduling is to only



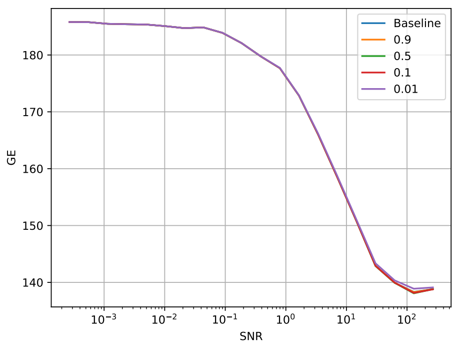
(a) AES - $q = 256$



(b) Ascon - $q = 256$



(c) ISAP-RK - $q = 256$



(d) Kyber NTT - $q = 3329$

Figure 13: Comparison of different damping factors, $\alpha \in [0.01, 0.1, 0.5, 0.9]$.

propagate messages which contain the largest amount of change, with the assumption that small messages cause divergence.

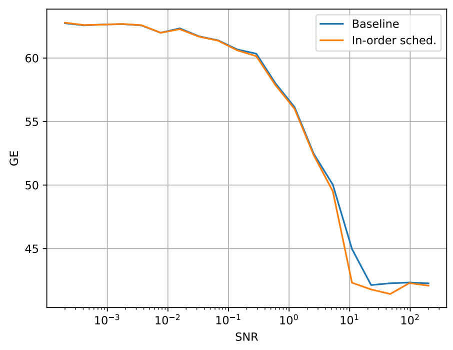
In our experiments, we studied the effect of the priority scheduling for $p \in [25, 50, 75]\%$. The $\mathrm{Q}_{\mathrm{rank}}(\cdot)$ function is defined as the Hellinger distance between the current and previous iterations of a F2V message:

$$\mathrm{Q}_{\mathrm{rank}}(\mu^{f_j \to X_i(t)}) = \frac{1}{\sqrt{2}} \left\| \sqrt{\mu^{f_j \to X_i(t)}} - \sqrt{\mu^{f_j \to X_i(t-1)}} \right\|_2 \tag{12}$$

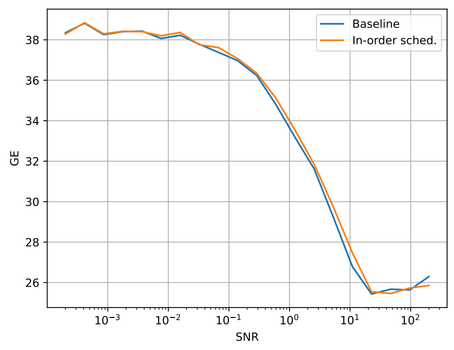
The Hellinger distance quantifies the similarity between two probability distributions and was previously used [GRO18] to improve SASCAs on AES.

Our results in Figure 15 indicate that this scheduling algorithm has no effect on the outcome of the SASCA. We observed that the convergence effects of this algorithm are similar to that of message damping. All messages in the graph are eventually propagated, but at a slower rate than that of the baseline. The parameter p determines the rate of convergence, but does not reach other solutions.

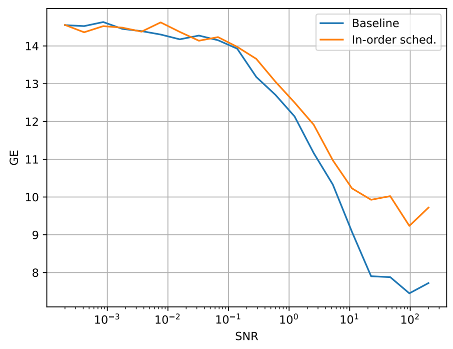
Merging of factor nodes. It is well known that the structure of the factor graph impacts the quality of results obtained from BP. In particular, short loops within the factor graph compound the positive feedback effect, causing divergence or oscillations. [Sto03] and [Yed04] first observed that BP on factor graphs of the FFT could be improved by merging factor nodes within the butterfly structure to a single node. [PP19] later applied this idea to the NTT to improve the SASCA attack on Kyber.



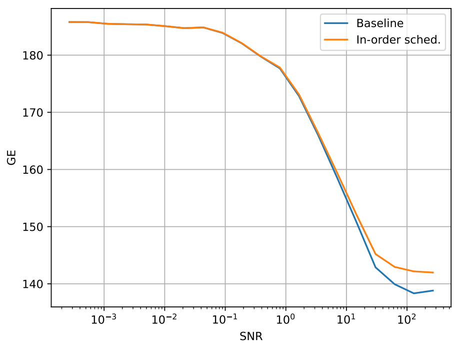
(a) AES - $q = 256$



(b) Ascon - $q = 256$



(c) ISAP-RK - $q = 256$



(d) Kyber NTT - $q = 3329$

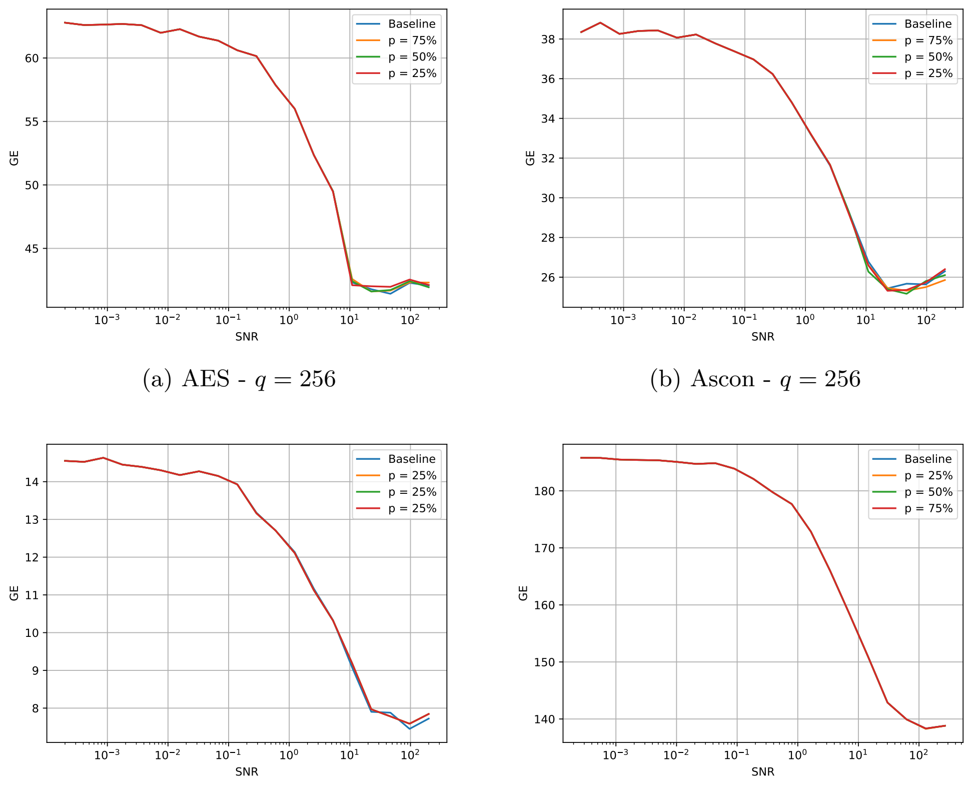
Figure 14: Comparison of in-order scheduling

We implemented the butterfly factor similar to that of [PP19], defined in Equation 9. In Figure 16, we show the result of the butterfly factor and the baseline case. For high SNRs, the butterfly factor case is a significant improvement. As explained in [PP19], the removal of small loops improves the quality of inference. The addition of this factor in their work most likely contributes the most meaningful improvement of results compared to [PPM17].

This improvement comes at increased computational cost. When merging two or more factors, the scope of the resulting factor increases; the cost of the summation to compute messages in Equation 7 therefore increases exponentially.

Simplified factor graphs for masked circuits [BS21]. Bronchain et al. describe modifying the factor graph of a masked circuit to remove the additional complexity of incorporating descriptions of masked gadgets (e.g., ISW multipliers). In this technique, they combine the unmasked version of the graph with an encoding graph, where the variables of an unmasked graph are estimated by their shares i.e., the marginal probability of x is estimated by the XOR of the marginals on the d-shares of x: $\Pr[X] = f_{\oplus_{i=0}^{d}}(\Pr[X_i])$.

In Figure 17, we compare the result of inference on the simplified graph versus the true masked graph. The results indicate that both techniques are roughly equivalent, except for the 2-share graphs in high SNR ranges. This is consistent with our observations on Figure 11.



(a) AES - $q = 256$ (b) Ascon - $q = 256$

(c) ISAP-RK - $q = 256$ (d) Kyber NTT - $q = 3329$

Figure 15: Comparison of priority scheduling for different p-values; $p \in [25, 50, 75]\%$.

Dropping high entropy factors. As discussed in Section 3, graph structure and the contents of individual beliefs can cause BP to perform poorly. Further, Knoll et al. showed that high entropy in prior evidence on variables (e.g., from noisy leakage measurements) can cause BP to destabilize [KP17]. These observations motivated us to explore eliminating factors which propagate beliefs with high entropy. For this investigation, we modify the numerical experiment setup: the "SNR" parameter σ is now a maximum, and for each variable node X in the factor graph, we select uniformly at random an actual SNR σ_X such that $0 \leq \sigma_X \leq \sigma$. The leakage is then generated as previously described.

In SASCAs, high entropy beliefs originate from factors connected to variables nodes whose prior evidence is derived from noisy measurements. In Figure 18, we sort factors based on the maximum entropy of incoming beliefs, from highest to lowest. Then, we eliminate factors from the graph according to this ordering, until the graph becomes acyclic. For all considered factor graphs, this heuristic performs better than baseline BP. We observe that high entropy beliefs tend to lead to situations similar to that described in Section 3.4, as higher overall entropy on priors introduces more possible incorrect steady states BP can reach.

As a control, we also perform a graph pruning BP for the "baseline" experimental case where all variable nodes have the same SNR (Figure 19). This results in a general degradation of the performance, leading us to conclude that the cycle pruning heuristic is not always beneficial.

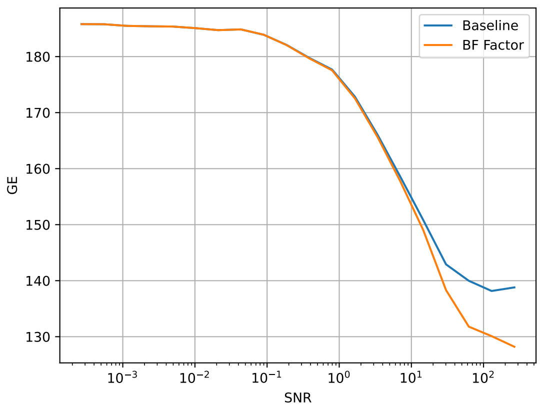
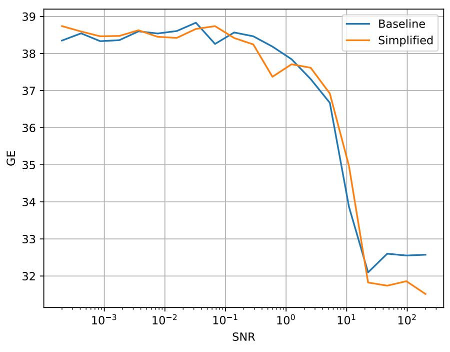


Figure 16: Comparison of the butterfly factor node (Equation 9) on the Kyber NTT - $q = 3329$

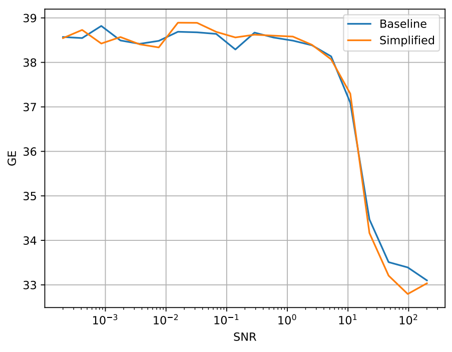
5 Discussion

In this paper, we reviewed various techniques to solve the side-channel inference problem and how belief propagation behaves in the context of the SASCA. First, we analyzed how BP can lead to misleading or incorrect results under several cases relevant to cryptographic attacks. Then, we compared SASCAs to exact inference methods such as variable elimination on Bayesian networks and brute-force enumeration. The main downside of these methods is their bad performance scaling when considering complex cryptographic circuits, making them often unusable, but they provide an interesting bound to compare SASCAs against. Next, we explored various tweaks to belief propagation. While some of these may improve the quality of the inference results, there is no “one size fits all” solution: the effectiveness of most tweaks depends on many parameters, including the considered problem (i.e., the structure of the factor graph), the size of the variables’ domain and the actual probability distributions (related to the noise level in our experiments).

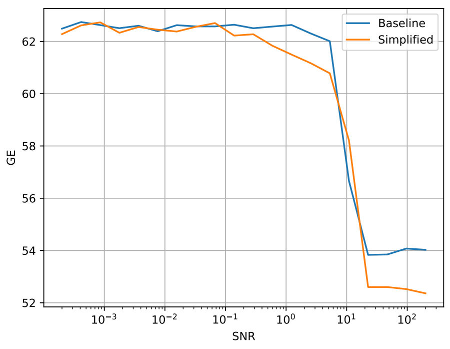
A consistent positive impact could only be achieved by merging multiple nodes into a single one, along with strategies such as removing variable nodes with high entropy on their prior evidence. Merging multiple nodes brings BP closer to exact inference; providing more accurate results at a higher computational cost. Consequently, our experiments highlight the important role of the graph structure in enhancing the quality of inference with BP. In the particular case where some variables are uninformative (low leakage SNR), though, removing the factors adjacent to these variables from the graph in order to eliminate cycles would generally lead to significant improvements. Further, we analyzed the gap between exact inference and BP to quantify the (in-)effectiveness of BP on SASCA problems. For high SNR cases, BP performs well, performing close to exact inference. With decreasing SNR, however, a notable performance gap between exact and approximate inference appears. One possible explanation for the poor performance in this region – given in [KF09, Chapter 11.3.4] – lies in the nature of the factors within cryptographic factor graphs: “graphs with factors closer to deterministic are more likely to have problems with convergence”. Indeed, deterministic factors (such as the ones used to model computations) are strongly influenced by their parents and a message from one parent “pulls” the belief in one direction, which may contradict the message from the other parent, which pulls in another direction, causing oscillations. Non-deterministic factors have a “smoothing” effect which reduces this effect.



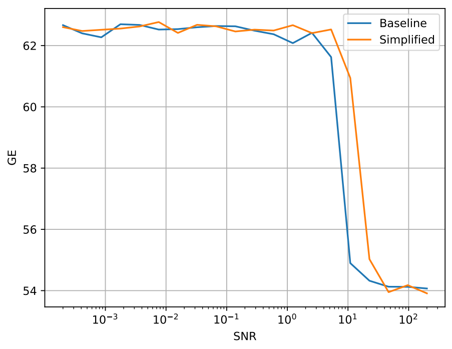
(a) 2-Share Ascon - $q = 256$



(b) 3-Share Ascon - $q = 256$



(c) 2-Share AES - $q = 256$



(d) 3-Share AES - $q = 256$

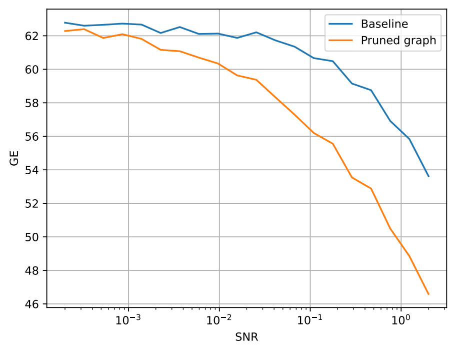
Figure 17: Comparison of the simplified masked factor graphs.

Let us now discuss the impacts of our findings on side-channel security evaluations.

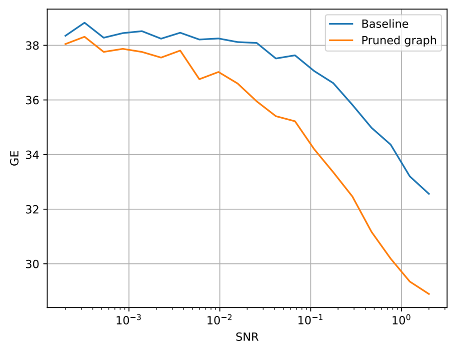
A first evaluation approach is to find the best possible attack, which in practice implies finding the best way to run BP. Our results indicate that this is a challenging task, since there are heuristics that can either improve or degrade the performance of simple BP, depending on the attack parameters (e.g., factor graph, information on the variables). A conceptually simple way out of this lack of a uniform behavior is to consider these heuristics (and their parameters) as hyperparameters of the attack to be automatically optimized. However, given the large design space, this may be computationally costly and lead to hard-to-explain results. Narrowing down the most promising heuristic candidates for particular circuits of interest is therefore a vast area for future investigations.

Another evaluation approach is based on finding upper-bounds to attacks. Here, our results show that running BP is unlikely to achieve performance near to the exact inference bound. Further, even considering only BP, it is likely not possible to find general error bounds [YFW05; PdGF+15][12]. To the best of our knowledge, the only attempt at bounding the capabilities of the SASCA inference has been the local random probing model (LRPM) [GGSB20; CS19], which is a heuristic technique that uses an information-based modeling of the BP algorithm. This approach is heuristic in nature, therefore it does not provide any formal bounds. However, in presence of cycles in the factor graph, it adopts

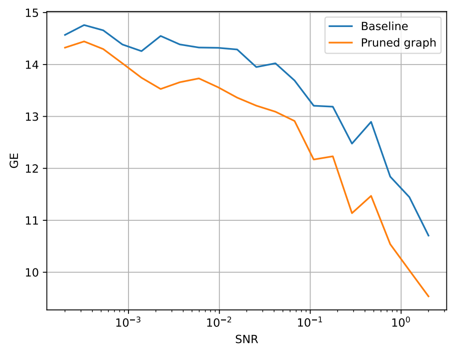
[12]The errors in BP have been analyzed for some classes of factor graphs (such as binary MRFs), which unfortunately do not map to the ones seen in SASCAs with a reasonable complexity. (All SASCA problems can be expressed as binary MRF graphs, however, the graph complexity scales poorly making the cost of inference prohibitive.)



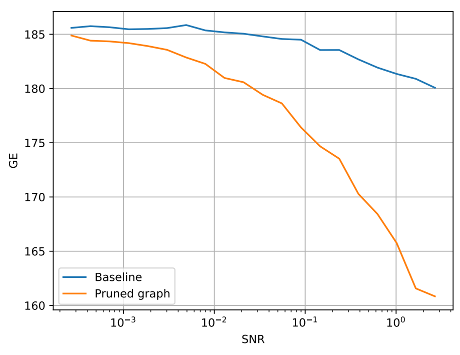
(a) AES - $q = 256$



(b) Ascon - $q = 256$



(c) ISAP-RK - $q = 256$



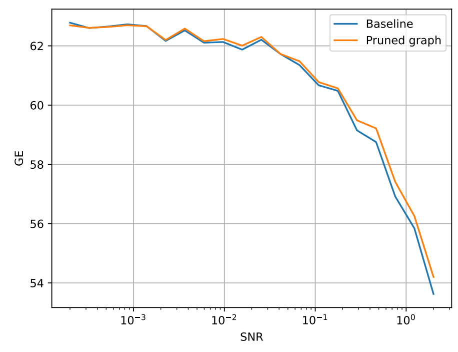
(d) Kyber NTT - $q = 3329$

Figure 18: Comparison of removing factors connected to variables with high entropy on their prior evidence.

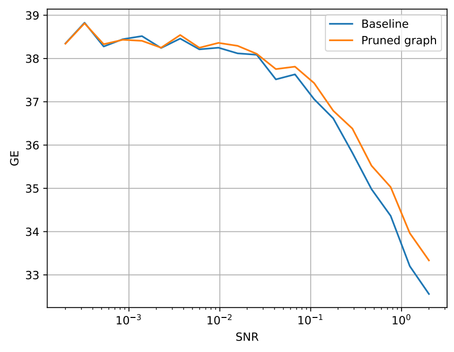
the conservative behavior of over-estimating the information. Nevertheless, the LRPM does not provide an upper bound on the exact inference capabilities, since it may suffer from the blinding problem, in the same way as BP.

At a higher level, it is still unclear which strategy an ideal evaluation should aim for: bounding the capabilities of an exact inference attack, or of the best possible BP variant? The former may be overly conservative, whereas the latter does not provide any concrete guarantees. Further, it is unclear how well BP performs compared to exact inference in general, requiring an evaluator to extrapolate based on simplified cases.

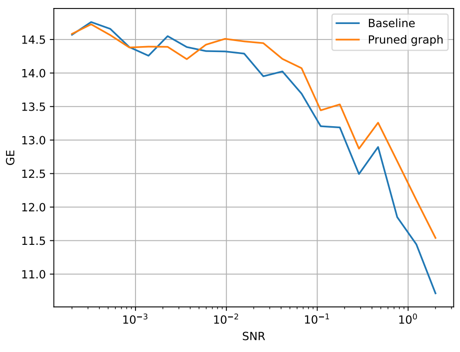
Acknowledgements Gaëtan Cassiers is a post-doctoral researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S.). This work was partially supported by the FWF SFB project SpyCoDe F8504, DESLab and by the European Union (ERC, AMAskZONE, 101077506). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. We thank our anonymous reviewers for their valuable comments in strengthening this work. We also would like to thank Christian Toth for their helpful discussions.



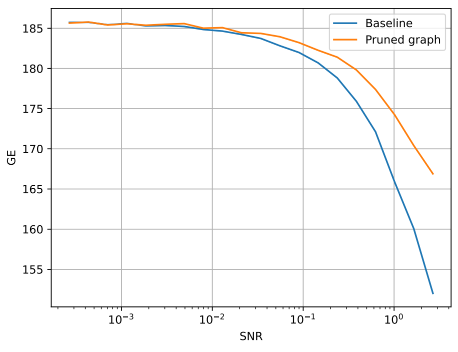
(a) AES - $q = 256$



(b) Ascon - $q = 256$



(c) ISAP-RK - $q = 256$



(d) Kyber NTT - $q = 3329$

Figure 19: Comparison of randomly eliminating factors until an acyclic factor graph is created.

References

[AAK20] Vitaly Aksenov, Dan Alistarh, and Janne H. Korhonen. Relaxed scheduling for scalable belief propagation. *CoRR*, abs/2002.11505, 2020. arXiv: 2002.11505. URL: https://arxiv.org/abs/2002.11505.

[BBC+20] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020. DOI: 10.1007/978-3-030-56784-2_13. URL: https://doi.org/10.1007/978-3-030-56784-2_13.

[BBPS19] Madalina Bolboceanu, Zvika Brakerski, Renen Perlman, and Devika Sharma. Order-lwe and the hardness of ring-lwe with entropic secrets. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019,* *Proceedings, Part II*, volume 11922 of *Lecture Notes in Computer Science*, pages 91–120. Springer, 2019. DOI: 10.1007/978-3-030-34621-8_4. URL: https://doi.org/10.1007/978-3-030-34621-8_4.

[BCM+23] Sonia Belaïd, Gaëtan Cassiers, Camille Mutschler, Matthieu Rivain, Thomas Roche, François-Xavier Standaert, and Abdul Rahman Taleb. Towards achieving provable side-channel security in practice. *IACR Cryptol. ePrint Arch.*:1198, 2023. URL: https://eprint.iacr.org/2023/1198.

[BCS21] Olivier Bronchain, Gaëtan Cassiers, and François-Xavier Standaert. Give me 5 minutes: attacking ASCAD with a single side-channel trace. *IACR Cryptol. ePrint Arch.*:817, 2021. URL: https://eprint.iacr.org/2021/817.

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 353–367. IEEE, 2018. DOI: 10.1109/EUROSP.2018.00032. URL: https://doi.org/10.1109/EuroSP.2018.00032.

[BHM+19] Olivier Bronchain, Julien M. Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: bounding model errors in side-channel security evaluations. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 713–737. Springer, 2019. DOI: 10.1007/978-3-030-26948-7_25. URL: https://doi.org/10.1007/978-3-030-26948-7_25.

[BP12] Joan Boyar and René Peralta. A small depth-16 circuit for the AES s-box. In Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou, editors, *Information Security and Privacy Research - 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Heraklion, Crete, Greece, June 4-6, 2012. Proceedings*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 287–298. Springer, 2012. DOI: 10.1007/978-3-642-30436-1_24. URL: https://doi.org/10.1007/978-3-642-30436-1_24.

[BS21] Olivier Bronchain and François-Xavier Standaert. Breaking masked implementations with many shares on 32-bit software platforms or when the security order does not matter. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):202–234, 2021. DOI: 10.46586/TCHES.V2021.I3.202-234. URL: https://doi.org/10.46586/tches.v2021.i3.202-234.

[CB23] Gaëtan Cassiers and Olivier Bronchain. Scalib: A side-channel analysis library. *J. Open Source Softw.*, 8(86):5196, 2023. DOI: 10.21105/JOSS.05196. URL: https://doi.org/10.21105/joss.05196.

[CDS23] Giovanni Camurati, Matteo Dell'Amico, and François-Xavier Standaert. Mcrank: monte carlo key rank estimation for side-channel security evaluations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):277–300, 2023. DOI: 10.46586/TCHES.V2023.I1.277-300. URL: https://doi.org/10.46586/tches.v2023.i1.277-300.

[CDSU23] Gaëtan Cassiers, Henri Devillez, François-Xavier Standaert, and Balazs Udvarhelyi. Efficient regression-based linear discriminant analysis for side-channel security evaluations towards analytical attacks against 32-bit implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(3):270–293, 2023. DOI: 10.46586/TCHES.V2023.I3.270-293. URL: https://doi.org/10.46586/tches.v2023.i3.270-293.

[CK13] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2013. DOI: 10.1007/978-3-319-08302-5_17. URL: https://doi.org/10.1007/978-3-319-08302-5_17.

[CK14] Marios O. Choudary and Markus G. Kuhn. Efficient stochastic methods: profiled attacks beyond 8 bits. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 85–103. Springer, 2014. DOI: 10.1007/978-3-319-16763-3_6. URL: https://doi.org/10.1007/978-3-319-16763-3_6.

[CRBO24] Aakash Chowdhury, Arnab Roy, Carlo Brunetta, and Elisabeth Oswald. Leakage certification made simple. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VI*, volume 14925 of *Lecture Notes in Computer Science*, pages 427–460. Springer, 2024. DOI: 10.1007/978-3-031-68391-6_13. URL: https://doi.org/10.1007/978-3-031-68391-6_13.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002. DOI: 10.1007/3-540-36400-5_3. URL: https://doi.org/10.1007/3-540-36400-5_3.

[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.*, 2019(2):162–198, 2019. DOI: 10.13154/TCHES.V2019.I2.162-198. URL: https://doi.org/10.13154/tches.v2019.i2.162-198.

[Dar09] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009. ISBN: 978-0-521-88438-9. URL: http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=9780521884389.

[DDF19] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: from probing attacks to noisy leakage. *J. Cryptol.*, 32(1):151–177, 2019. DOI: 10.1007/S00145-018-9284-1. URL: https://doi.org/10.1007/s00145-018-9284-1.

[DEM+20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2.0. *IACR Trans. Symmetric Cryptol.*, 2020(S1):390–416, 2020. DOI: 10.13154/TOSC.V2020.IS1.390-416. URL: https://doi.org/10.13154/tosc.v2020.iS1.390-416.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021. DOI: 10.1007/S00145-021-09398-9. URL: https://doi.org/10.1007/s00145-021-09398-9.

[GGSB20] Qian Guo, Vincent Grosso, François-Xavier Standaert, and Olivier Bronchain. Modeling soft analytical side-channel attacks from a coding theory viewpoint. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):209–238, 2020. DOI: 10.13154/TCHES.V2020.I4.209-238. URL: https://doi.org/10.13154/tches.v2020.i4.209-238.

[GRO18] Joey Green, Arnab Roy, and Elisabeth Oswald. A systematic study of the impact of graphical models on inference-based attacks on AES. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications, 17th International Conference, CARDIS 2018, Montpellier, France, November 12-14, 2018, Revised Selected Papers*, volume 11389 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2018. DOI: 10.1007/978-3-030-15462-2_2. URL: https://doi.org/10.1007/978-3-030-15462-2_2.

[HHP+21] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine van Vredendaal. Chosen ciphertext k-trace attacks on masked CCA2 secure kyber. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):88–113, 2021. DOI: 10.46586/TCHES.V2021.I4.88-113. URL: https://doi.org/10.46586/tches.v2021.i4.88-113.

[HMS+23] Julius Hermelink, Erik Mårtensson, Simona Samardjiska, Peter Pessl, and Gabi Dreo Rodosek. Belief propagation meets lattice reduction: security estimates for error-tolerant key recovery from decryption errors. *IACR Cryptol. ePrint Arch.*:98, 2023. URL: https://eprint.iacr.org/2023/098.

[IIW05] Alexander T. Ihler, John W. Fisher III, and Alan S. Willsky. Loopy belief propagation: convergence and effects of message errors. *J. Mach. Learn. Res.*, 6:905–936, 2005. URL: https://jmlr.org/papers/v6/ihler05a.html.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003. DOI: 10.1007/978-3-540-45146-4_27. URL: https://doi.org/10.1007/978-3-540-45146-4_27.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009. ISBN: 978-0-262-01319-2. URL: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2%5C&tid=11886.

[KFL01] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inf. Theory*, 47(2):498–519, 2001. DOI: 10.1109/18.910572. URL: https://doi.org/10.1109/18.910572.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. DOI: 10.1007/3-540-48405-1_25. URL: https://doi.org/10.1007/3-540-48405-1_25.

[KP17] Christian Knoll and Franz Pernkopf. On loopy belief propagation - local stability analysis for non-vanishing fields. In Gal Elidan, Kristian Kersting, and Alexander Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017. URL: http://auai.org/uai2017/proceedings/papers/160.pdf.

[KPP20] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on keccak. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):243–268, 2020. DOI: 10.13154/TCHES.V2020.I3.243-268. URL: https://doi.org/10.13154/tches.v2020.i3.243-268.

[LKSP21] Harald Leisenberger, Christian Knoll, Richard Seeber, and Franz Pernkopf. Convergence behavior of belief propagation: estimating regions of attraction via lyapunov functions. In Cassio P. de Campos, Marloes H. Maathuis, and Erik Quaeghebeur, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021, Virtual Event, 27-30 July 2021*, volume 161 of *Proceedings of Machine Learning Research*, pages 1863–1873. AUAI Press, 2021. URL: https://proceedings.mlr.press/v161/leisenberger21a.html.

[Mac03] David J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003. ISBN: 978-0-521-64298-9.

[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.*, 2023(3):522–569, 2023. DOI: 10.46586/TCHES.V2023.I3.522-569. URL: https://doi.org/10.46586/tches.v2023.i3.522-569.

[MDP20] Loïc Masure, Cécile Dumas, and Emmanuel Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):348–375, 2020. DOI: 10.13154/TCHES.V2020.I1.348-375. URL: https://doi.org/10.13154/tches.v2020.i1.348-375.

[MK05] Joris M Mooij and Hilbert J Kappen. On the properties of the bethe approximation and loopy belief propagation on binary networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11):P11012, 2005. DOI: 10.1088/1742-5468/2005/11/P11012.

[MK07] Joris M. Mooij and Hilbert J. Kappen. Sufficient conditions for convergence of the sum-product algorithm. *IEEE Trans. Inf. Theory*, 53(12):4422–4437, 2007. DOI: 10.1109/TIT.2007.909166. URL: https://doi.org/10.1109/TIT.2007.909166.

[MWJ99] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: an empirical study. In Kathryn B. Laskey and Henri Prade, editors, *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 467–475. Morgan Kaufmann, 1999. URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1%5C&smnu=2%5C&article_id=199%5C&proceeding_id=15.

[NDGJ21] Kalle Ngo, Elena Dubrova, Qian Guo, and Thomas Johansson. A side-channel attack on a masked IND-CCA secure saber KEM implementation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):676–707, 2021. DOI: 10.46586/TCHES.V2021.I4.676-707. URL: https://doi.org/10.46586/tches.v2021.i4.676-707.

[PdGF+15] Nathalie Peyrard, Simon de Givry, Alain Franc, Stéphane Robin, Régis Sabbadin, Thomas Schiex, and Matthieu Vignes. Exact and approximate inference in graphical models: variable elimination and beyond. *CoRR*, abs/1506.08544, 2015. arXiv: 1506.08544. URL: http://arxiv.org/abs/1506.08544.

[Pea82] Judea Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In David L. Waltz, editor, *Proceedings of the National Conference on Artificial Intelligence, Pittsburgh, PA, USA, August 18-20, 1982*, pages 133–136. AAAI Press, 1982. URL: http://www.aaai.org/Library/AAAI/1982/aaai82-032.php.

[PI17] Wei Ping and Alexander Ihler. Belief propagation in conditional rbms for structured prediction. In Aarti Singh and Xiaojin (Jerry) Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pages 1141–1149. PMLR, 2017. URL: http://proceedings.mlr.press/v54/ping17a.html.

[PP19] Peter Pessl and Robert Primas. More practical single-trace attacks on the number theoretic transform. In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 130–149. Springer, 2019. DOI: 10.1007/978-3-030-30530-7_7. URL: https://doi.org/10.1007/978-3-030-30530-7_7.

[PPM17] Robert Primas, Peter Pessl, and Stefan Mangard. Single-trace side-channel attacks on masked lattice-based encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 513–533. Springer, 2017. DOI: 10.1007/978-3-319-66787-4_25. URL: https://doi.org/10.1007/978-3-319-66787-4_25.

[PSG16] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. Simple key enumeration (and rank estimation) using histograms: an integrated approach. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*, volume 9813 of *Lecture Notes in Computer Science*, pages 61–81. Springer, 2016. DOI: 10.1007/978-3-662-53140-2_4. URL: https://doi.org/10.1007/978-3-662-53140-2_4.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001. DOI: 10.1007/3-540-45418-7_17. URL: https://doi.org/10.1007/3-540-45418-7_17.

[RSV+11] Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre. A formal study of power variability issues and side-channel attacks for nanoscale devices. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2011. DOI: 10.1007/978-3-642-20465-4_8. URL: https://doi.org/10.1007/978-3-642-20465-4_8.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer, 2005. DOI: 10.1007/11545262_3. URL: https://doi.org/10.1007/11545262_3.

[Sto03] Amos J. Storkey. Generalised propagation for fast fourier transforms with partial or missing data. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16 [Neural Information Processing Systems, NIPS 2003, December 8-13, 2003, Vancouver and Whistler, British Columbia, Canada]*, pages 433–440. MIT Press, 2003. URL: https://proceedings.neurips.cc/paper/2003/hash/8c1b6fa97c4288a4514365198566c6fa-Abstract.html.

[SW15] Qinliang Su and Yik-Chung Wu. On convergence conditions of gaussian belief propagation. *IEEE Trans. Signal Process.*, 63(5):1144–1155, 2015. DOI: 10.1109/TSP.2015.2389755. URL: https://doi.org/10.1109/TSP.2015.2389755.

[VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014. DOI: 10.1007/978-3-662-45611-8_15. URL: https://doi.org/10.1007/978-3-662-45611-8_15.

[Wei00] Yair Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Comput.*, 12(1):1–41, 2000. DOI: 10.1162/089976600300015880. URL: https://doi.org/10.1162/089976600300015880.

[WNC+24] Thomas Wedenig, Rishub Nagpal, Gaëtan Cassiers, Stefan Mangard, and Robert Peharz. Exact soft analytical side-channel attacks using tractable circuits. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL: https://openreview.net/forum?id=0mklK4h0rX.

[Yed04] Jonathan S. Yedidia. Sparse factor graph representations of reed-solomon and related codes. In *Proceedings of the 2004 IEEE International Symposium on Information Theory, ISIT 2004, Chicago Downtown Marriott, Chicago, Illinois, USA, June 27 - July 2, 2004*, page 260. IEEE, 2004. DOI: 10.1109/ISIT.2004.1365296. URL: https://doi.org/10.1109/ISIT.2004.1365296.

[YFW05] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Trans. Inf. Theory*, 51(7):2282–2312, 2005. DOI: 10.1109/TIT.2005.850085. URL: https://doi.org/10.1109/TIT.2005.850085.

[YK21] Shih-Chun You and Markus G. Kuhn. Single-trace fragment template attack on a 32-bit implementation of keccak. In Vincent Grosso and Thomas Pöppelmann, editors, *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*, volume 13173 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2021. DOI: 10.1007/978-3-030-97348-3_1. URL: https://doi.org/10.1007/978-3-030-97348-3_1.