A Survey of Constraint-Based Side-Channel Analysis: From Algebraic Attacks to Exact Probabilistic Inference

GAURAV KUMAR, Centre for Development of Advanced Computing, India

Over the last fifteen years, side-channel analysis (SCA) against implementations of both classical and post-quantum cryptography has undergone a quiet but fundamental change of paradigm: from treating leakage as evidence to be combined with an algorithm's structure into a system of *exact* equations to be *solved*, to treating it as evidence to be combined into a joint probability distribution to be *queried*. This survey gives a systematic account of that evolution through four paradigms: Algebraic SCA (ASCA), which encodes leakage as hard Boolean or algebraic constraints and solves the resulting system with SAT or Gröbner-basis solvers; Tolerant/weighted Algebraic SCA (TASCA), which relaxes hard constraints into pseudo-Boolean costs and recovers a maximum-a-posteriori key via combinatorial optimization; Soft Analytical SCA (SASCA), which retains leakage as full likelihood functions attached to a factor graph and approximates the posterior with loopy belief propagation; and Exact SASCA (ExSASCA), which replaces the uncertified approximation of loopy belief propagation with exact inference on a knowledge-compiled tractable probabilistic circuit. Our central contribution is a single Bayesian factor-graph formulation under which all four paradigms are shown to be different *representations* of the same leakage-augmented joint distribution combined with different *inference queries and solvers*: we prove, in a precise but lightweight sense, that ASCA is the zero-temperature limit of TASCA, that TASCA is a maximum-a-posteriori restriction of SASCA's model, and that ExSASCA is SASCA with its approximate solver replaced by an exact one on the identical graphical model. Building on this formulation, we develop a taxonomy that organizes the literature along a representation axis (hard, weighted, probabilistic) and a solver axis (satisfiability, combinatorial optimization, approximate message passing, exact tractable inference), survey the concrete attack literature in each paradigm on both AES and NTT-based post-quantum schemes (Kyber, Dilithium, Falcon/FN-DSA), and analyze the computational and security trade-offs of each approach, including its implications for what a security evaluator may soundly conclude from a failed attack. We close with open problems at the current frontier of tractable probabilistic inference for cryptographic implementation security.

CCS Concepts: • **Security and privacy** → **Side-channel analysis and countermeasures**; *Cryptanalysis and other attacks*; • **Mathematics of computing** → *Probabilistic reasoning algorithms*; • **Theory of computation** → *Constraint and logic programming*.

Additional Key Words and Phrases: side-channel analysis, algebraic cryptanalysis, belief propagation, factor graphs, probabilistic circuits, knowledge compilation, masking countermeasures, post-quantum cryptography, lattice-based cryptography

1 Introduction

A cryptographic algorithm can be unconditionally sound on paper and still fail in silicon. The gap between the two is the subject of side-channel analysis (SCA): an implementation of a cipher does not merely compute a function, it also consumes power, radiates electromagnetically, and takes time to do so, and each of these physical observables is statistically dependent on the secret data flowing through the circuit [8, 16, 17]. Since the introduction of differential power analysis [17] and template attacks [2], the standard adversarial recipe has been *divide and conquer*: guess a small chunk of the secret (a key byte, a small group of coefficients), score each hypothesis against one or a few leaking intermediate variables that depend on it, and repeat independently for every chunk. This recipe is simple, well understood, and devastating against unprotected implementations – but it has a structural blind spot. It only ever looks at one small piece of the computation at a time,

Author's Contact Information: Gaurav Kumar, Centre for Development of Advanced Computing, Patna, India, gauravkumar.mnit@gmail.com.

and it therefore leaves on the table almost everything that a leaking device reveals about *how* its intermediate values relate to one another.

That blind spot becomes decisive as soon as an implementation is protected. Masking countermeasures [12, 29] split every sensitive variable into d+1 random shares such that no single share – and hence no single leakage point – carries any first-order information about the secret in isolation; a divide-and-conquer adversary who scores hypotheses one leakage sample at a time is, by the countermeasure's own design goal, left with nothing. Defeating masking requires *combining* many leakage points at once, in a way that respects the algebraic relations the algorithm imposes between them (a share and its complement, an S-box input and output, the two halves of a butterfly in a Number-Theoretic Transform). The same need for combination arises, for different reasons, whenever an adversary has only a single trace to work with – a realistic constraint for many public-key and post-quantum operations, where nonces, ephemeral randomness, or per-session freshness rule out the many repeated measurements that divide-and-conquer attacks assume [28].

Over the past decade and a half, the response to this need has crystallized into a distinct research thread that this survey calls *constraint-based side-channel analysis*: rather than attacking one intermediate variable at a time, treat the entire target algorithm – every gate, every S-box, every modular addition and NTT butterfly – together with everything the leakage says about every intermediate value it touches, as a single system to be solved or queried as a whole. Four paradigms have emerged along this thread, in a sequence that this survey argues is not a sequence of unrelated ideas but a single, well-defined trajectory through one design space:

- **Algebraic SCA (ASCA)**, introduced by Renauld, Standaert, and Veyrat-Charvillon [31, 33], represents the target cipher and leakage-derived information as an overdefined system of *exact* Boolean or algebraic equations and solves it with off-the-shelf SAT or Gröbner-basis solvers.
- **Tolerant/weighted Algebraic SCA (TASCA)**, introduced by Oren, Kirschbaum, Popp, and Wool [21], replaces ASCA's brittle hard constraints with *weighted* (pseudo-Boolean) costs derived from leakage confidence, recasting key recovery as combinatorial optimization rather than satisfiability.
- **Soft Analytical SCA (SASCA)**, introduced by Veyrat-Charvillon, Gérard, and Standaert [39], goes further still and retains leakage as full probability distributions attached to a factor-graph representation of the algorithm, approximating the posterior over the secret with loopy belief propagation.
- **Exact SASCA (ExSASCA)**, introduced very recently by Wedenig, Nagpal, Cassiers, Mangard, and Peharz [41], keeps SASCA's probabilistic representation but replaces its uncertified, sometimes-catastrophically-wrong approximate solver with *exact* inference on a knowledge-compiled tractable probabilistic circuit.

Figure 1 summarizes this progression. Each paradigm is usually presented in its own paper, with its own notation, its own running example, and its own claim to superiority over what came before – which makes it easy to lose sight of the fact that all four are answering variations on one question. The question this survey asks, precisely, is:

> *How has side-channel analysis evolved from solving hard algebraic constraints to performing soft and eventually exact probabilistic inference, and what are the computational and security trade-offs of each paradigm?*

and the answer we develop rests on a single technical claim, argued formally in Section 3:

> *ASCA, TASCA, SASCA, and ExSASCA are not four different attacks. They are four different ways of representing, and four different algorithms for querying, exactly one*

latent-secret inference problem – the posterior of the secret key given leakage, defined over exactly the same factor graph induced by the target implementation.

1.1 Contributions

This survey makes the following contributions.

(1) **A unified mathematical formulation** (Section 3) that casts all four paradigms as instances of one Bayesian inference problem over a factor graph shared by all of them, differing only along a *representation axis* (how much of the leakage likelihood $p(\ell_i \mid X_i)$ survives into the model: a hard indicator, a scalar weight, or a full distribution) and a *solver axis* (what query is asked of the resulting model, and with what algorithm: satisfiability, combinatorial optimization, approximate message passing, or exact tractable inference). We state and justify three reductions – ASCA as the zero-temperature limit of TASCA, TASCA as a maximum-a-posteriori restriction of SASCA, and ExSASCA as an exact-solver refinement of SASCA – that make the relationship between paradigms precise rather than metaphorical.
(2) **A systematic taxonomy** (Sections 4–7) that organizes fifteen years of literature within each paradigm along these same two axes, tracing the lineage from the original AES-focused attacks through their noise-tolerant, coding-theoretic, and lattice-based descendants.
(3) **A cross-paradigm trade-off analysis** (Section 8) covering exactness guarantees, noise tolerance, single-trace capability, scalability with treewidth, and – importantly for practitioners – what each paradigm's *failure* does and does not certify about the security of an implementation.
(4) **Case studies** (Section 9) grounding the taxonomy in concrete results on AES and on NTT-centric post-quantum schemes (Kyber, Dilithium, Falcon/FN-DSA), where constraint-based SCA has become one of the primary tools for evaluating masked implementations.
(5) **Open problems** (Section 10) at the current frontier: scaling exact tractable inference to the treewidth of masked, multi-kilobit lattice constructions; combining fault-derived hard constraints with leakage-derived soft ones in a single framework; and what a treewidth-aware notion of "compile-resistance" might mean as a design target for masked circuits.

1.2 Scope and Related Surveys

We restrict attention to attacks that explicitly represent the target algorithm's data-flow structure and solve or query a model built jointly from that structure and from leakage; we do not cover single-point divide-and-conquer distinguishers (DPA, CPA, MIA) except as background, nor do we cover the substantial and largely orthogonal literature on deep-learning-based profiling, which is surveyed in depth elsewhere [26]. We likewise treat security-evaluation methodology (key rank and enumeration [27, 34, 37, 38]) only insofar as it bears on how constraint-based attacks should be interpreted by an evaluator, not as a survey topic in its own right. To our knowledge, no prior survey treats ASCA, TASCA, SASCA, and ExSASCA as a single design space with a shared mathematical skeleton; existing systematizations of side-channel analysis are organized either by physical channel and countermeasure [26] or by evaluation metric, rather than by inference paradigm.

2 Background and Threat Model

2.1 Physical Leakage and Leakage Models

A physical implementation of a cryptographic algorithm leaks because its instantaneous power draw, electromagnetic emanation, or execution time is statistically dependent on the data it manipulates [8, 16, 17]. The standard modeling abstraction treats an intermediate value v processed by the device as inducing a leakage sample $\ell \in \mathbb{R}^d$ drawn from a channel $p(\ell \mid v)$; the two workhorse choices are

a deterministic *leakage function* $\lambda(v)$ (commonly the Hamming weight or Hamming distance of v) corrupted by additive Gaussian noise, and a *profiled template* [2], in which an adversary with access to a clone device estimates $p(\ell \mid v)$ directly (typically as a multivariate Gaussian $\mathcal{N}(\ell; \mu_v, \Sigma_v)$ per value v) from a large number of labeled traces. Given such a model and a uniform prior on v, Bayes' rule turns an observed leakage sample into a *belief* over v,

$$p(v \mid \ell) \;=\; \frac{p(\ell \mid v)}{\sum_{v'} p(\ell \mid v')},$$

and it is this belief – not the raw leakage trace itself – that every paradigm surveyed in this article ultimately consumes. Constraint-based SCA is thus best understood as living one layer of abstraction above the signal-processing question of *how* to turn a trace into $p(\ell \mid v)$ or $p(v \mid \ell)$ (itself a rich topic, surveyed for deep-learning-based profiling in [26]), and squarely inside the question of what to do with such beliefs once obtained for many intermediate variables at once.

2.2 Divide-and-Conquer Attacks and Their Limits

Classical differential and correlation power analysis attack one small, key-dependent intermediate value $v = f(k_j, p)$ at a time: for every candidate value of a key chunk k_j, compute a hypothetical leakage prediction and score it against the observed traces, independently for every chunk j. This *divide-and-conquer* strategy is attractive because it turns an exponential key-recovery problem into a sequence of small, independent, embarrassingly parallel searches, and because standard security-evaluation tooling – key rank and key enumeration algorithms [27, 34, 37, 38] – was built to characterize exactly this kind of chunk-wise, score-then-combine attack: an evaluator who cannot outright break a device can still bound how many candidate keys remain to be tested after the divide step, and thereby bound the attack's real-world cost even when it falls short of full key recovery.

The strategy's weakness is structural, not statistical: it only ever conditions on the leakage of the single variable being scored, and so it cannot exploit anything that only becomes visible when *several* leakage points are considered jointly. Two situations make this a hard limit rather than a soft inefficiency.

2.3 Masking, the Combination Problem, and Single-Trace Attacks

Masking countermeasures. A d-th order Boolean masking scheme replaces every sensitive variable v with $d{+}1$ shares $v = v^{(0)} \oplus \cdots \oplus v^{(d)}$, generated so that any d of the shares are individually uniformly random and independent of v [12]. Under the d-probing security model and its formalizations [12, 29], no function of at most d leakage points carries first-order statistical information about v: a divide-and-conquer adversary who scores hypotheses against one leakage sample at a time is, by the countermeasure's explicit design goal, reduced to noise. Defeating masking therefore requires *combining* information across the leakage of several shares (and typically several other intermediate variables besides), while respecting the algebraic relation – e.g., that the shares XOR to the unmasked value – that ties them together. This is precisely the "combine/conquer" step that divide-and-conquer attacks skip and that every paradigm surveyed in this article was designed to perform.

Single-trace attacks. A second, independent motivation for combination is data scarcity. Divide-and-conquer attacks, and the statistical distinguishers underlying them, typically need many measurements under a fixed (or repeated) secret to average out noise. This assumption fails whenever the secret-dependent computation is re-randomized on every invocation – ephemeral nonces in signing, fresh randomness in key generation, one-shot key exchange – so that an adversary

who wants to recover anything must extract it from a *single* trace [28]. A single trace, however, typically *does* contain leakage on tens or hundreds of intermediate variables computed from the same secret; the question is again how to combine all of that information rather than how to average away noise on one variable at a time. Both motivations point in the same direction, and it is this direction – attack the whole computation graph jointly – that defines constraint-based SCA.

2.4 Formal Setup

We now fix notation used throughout the rest of this survey. Let $K \in \mathcal{K}$ denote the secret (a symmetric key, or the coefficients of a lattice-based secret key/signing key), and let P denote public or known input (a plaintext, nonce, or syndrome). The target algorithm is a deterministic circuit: it computes a sequence of intermediate variables $X = (X_1, \ldots, X_n)$, each over a finite domain $\mathcal{X}_i$ (a bit, a byte, a ring element), via local update functions

$$X_i \;=\; g_i\big(X_{\mathrm{pa}(i)}, K, P\big), \qquad i = 1, \ldots, n,$$

where $\mathrm{pa}(i)$ indexes the "parent" variables that X_i depends on directly. This is nothing more than the algorithm's data-flow graph made explicit: an S-box lookup, an XOR, a modular multiplication, an NTT butterfly, and a comparison inside a discrete Gaussian sampler are all instances of some g_i. The physical device leaks a channel L_i per intermediate variable (or per small group of variables), $L_i \sim p(\ell_i \mid X_i)$, and the adversary observes a realization $\boldsymbol{\ell} = (\ell_1, \ldots, \ell_n)$ together with P, with the goal of learning K.

3 A Unified Formulation of Constraint-Based Side-Channel Analysis

This section develops the survey's central technical claim: that ASCA, TASCA, SASCA, and ExSASCA are four points in one two-dimensional design space, obtained by making two independent choices – how much of the leakage likelihood to retain (*representation*), and what query to ask of the resulting model and with what algorithm (*solver*) – rather than four unrelated attacks that happen to share a target.

3.1 The Latent-Secret Inference Problem

By Bayes' rule, and using the fact that X is a deterministic function of (K, P) under the circuit $\{g_i\}$, the joint posterior over the secret and every intermediate variable, given leakage and public input, factors as

$$p(K, X \mid \boldsymbol{\ell}, P) \;\propto\; p(K) \underbrace{\left[\prod_{i=1}^{n} \mathbb{1}\{X_i = g_i(X_{\mathrm{pa}(i)}, K, P)\}\right]}_{\Phi_C(K, X; P)} \underbrace{\left[\prod_{i=1}^{n} \phi_i(X_i; \ell_i)\right]}_{\Phi_L(X; \boldsymbol{\ell})}. \tag{1}$$

The first bracket, Φ_C, is a product of *hard* indicator factors that simply re-state the algorithm: it is 1 on the unique (K, X) consistent with the circuit and P, and 0 everywhere else, for every choice of K. The second bracket, Φ_L, is a product of *leakage factors* $\phi_i(X_i; \ell_i)$, one per instrumented intermediate variable, that encode what the observed leakage says about that variable. Equation (1) is exactly a *factor graph* [18, 19]: a bipartite graph $G = (V, F)$ with a variable node for K and for every X_i, and a factor node for every term of Φ_C and Φ_L, connected to the variables it involves. Crucially, G's variable-and-circuit-factor skeleton is determined entirely by the target *algorithm* (and which variables are instrumented); it is identical across all four paradigms surveyed in this article. What differs between paradigms is (i) how $\phi_i(X_i; \ell_i)$ represents $p(\ell_i \mid X_i)$, and (ii) which inference query is asked of Equation (1), and with what solver.

Definition 1 (Latent-secret inference problem). *Given a factor graph G built from a target algorithm as above, a choice of leakage-factor representation $\{\phi_i(\cdot;\ell_i)\}_{i=1}^n$, and a query $Q \in \{\textsc{Sat}, \textsc{Map}, \textsc{Mar}\}$, the* latent-secret inference problem *(G, ϕ, Q) is the problem of computing Q against $p(K, X \mid \boldsymbol{\ell}, P) \propto \Phi_C(K, X; P) \cdot \Phi_L(X; \boldsymbol{\ell})$ from Equation (1), where Sat asks for a (the) satisfying assignment, Map asks for $\arg\max_{K,X} p(K, X \mid \boldsymbol{\ell}, P)$, and Mar asks for the marginal $p(K_j \mid \boldsymbol{\ell}, P)$ of each secret sub-variable K_j.*

Every constraint-based side-channel attack in the literature is an instantiation of Definition 1. Table 1 and Figure 1 summarize the four instantiations that this survey treats as paradigms; we describe each in turn before formalizing how they relate to one another.

Algebraic SCA (ASCA). The leakage factor collapses $p(\ell_i \mid X_i)$ to a *hard* indicator over a set $S(\ell_i)$ of values judged consistent with the observed leakage under some decision rule (often a threshold on Hamming weight): $\phi_i(X_i; \ell_i) = \mathbb{1}\{X_i \in S(\ell_i)\}$. The query is Sat: find any (K, X) satisfying every circuit and leakage factor – equivalently, solve an overdefined system of Boolean or algebraic equations, one sub-system from Φ_C (the cipher) and one from Φ_L (the leakage). The canonical solvers are conflict-driven-clause-learning SAT solvers over a Boolean encoding, or Gröbner-basis computation over a polynomial encoding in GF(2) or GF(2^8) [31, 33].

Tolerant/weighted Algebraic SCA (TASCA). The leakage factor is retained as a *scalar weight* rather than collapsed to a 0/1 indicator: writing $w_i(X_i) = -\log \phi_i(X_i; \ell_i)$ for a cost derived from (an often quantized approximation of) $-\log p(\ell_i \mid X_i)$, the circuit factors Φ_C remain exactly as hard as in ASCA, but each leakage factor becomes $\phi_i(X_i; \ell_i) = \exp(-w_i(X_i))$. The query is Map: minimize $\sum_i w_i(X_i)$ subject to $\Phi_C(K, X; P) = 1$. This is solved with pseudo-Boolean, MaxSAT, integer-linear-programming, or constraint-programming solvers [21, 23, 24].

Soft Analytical SCA (SASCA). The leakage factor is retained in full: $\phi_i(X_i; \ell_i) = p(\ell_i \mid X_i)$, a properly normalized distribution over X_i's entire domain, with no collapse to a point decision or a single scalar. The query is Mar: compute the marginal posterior of each secret sub-variable. This is solved by sum-product message passing (belief propagation) directly on G [19, 39] – exact if G is a tree, and an uncertified approximation ("loopy BP") whenever G contains cycles, which it does for essentially every cipher of practical interest.

Exact SASCA (ExSASCA). The representation is identical to SASCA – full likelihoods, no collapse. The query is Mar and/or Map (in the tractable-inference literature, a Map query over X is usually called *most probable explanation*, MPE). What changes is the solver: rather than running approximate message passing directly on the loopy graph G, ExSASCA first *compiles* the loopy portion of G – via knowledge compilation [6, 7] into a tractable probabilistic circuit such as a probabilistic sentential decision diagram (PSDD) [5, 14] – into an equivalent representation on which sum-product and max-product operations are *exact* and run in time polynomial in the compiled circuit's size, regardless of how many cycles the original graph had [41].

3.2 Three Reductions

Table 1 makes the family resemblance visible; the following three remarks make it precise. Each states that one paradigm is recovered from another by a specific, well-defined operation on Equation (1) – not merely that the two are "similar in spirit."

Remark 1 (ASCA is the zero-temperature limit of TASCA). *Introduce an inverse-temperature parameter $\beta \geq 0$ into TASCA's leakage factor, $\phi_i^\beta(X_i; \ell_i) = \exp(-\beta\, w_i(X_i))$, so that $\beta = 1$ recovers TASCA as defined above. As $\beta \to \infty$, ϕ_i^β concentrates all of its mass on $\arg\min_{X_i} w_i(X_i)$ and converges*

Table 1. The representation and solver axes of constraint-based SCA. All four paradigms solve Definition 1 on the same factor graph G; they differ in how much of $p(\ell_i \mid X_i)$ the leakage factor ϕ_i retains, and in the query and algorithm used to answer it.

Paradigm	**Leakage factor** $\phi_i(X_i; \ell_i)$	**Query**	**Solver**	**Exactness**
Algebraic SCA (§4)	hard indicator $\mathbb{1}\{X_i \in S(\ell_i)\}$	SAT	SAT / Gröbner basis	exact when it terminates, but brittle: a single misclassified sample makes the whole system UNSAT
TASCA (§5)	scalar weight $\exp(-w_i(X_i))$	MAP	pseudo-Boolean / MaxSAT / ILP / CP	exact MAP on the weighted model, but only a point estimate – no posterior or ranking
SASCA (§6)	full likelihood $p(\ell_i \mid X_i)$	MAR	loopy belief propagation	approximate; convergence and correctness are both uncertified on cyclic graphs
ExSASCA (§7)	full likelihood $p(\ell_i \mid X_i)$	MAR / MAP	knowledge-compiled tractable circuit	exact, subject to compiled-circuit size (worst case exponential in treewidth)

(after renormalization) to a hard indicator on that single value; if instead w_i is itself defined as a hard decision rule (finite cost on a consistency set $S(\ell_i)$, infinite cost outside it, as is typical when TASCA weights are derived from a thresholded leakage classifier), then $\phi_i^{\beta} \to \mathbb{1}\{X_i \in S(\ell_i)\}$ for any $\beta > 0$. *Either way, ASCA's hard-constraint representation is recovered from TASCA's weighted representation in the limit of full confidence in the leakage-derived decision, i.e.,* ASCA = TASCA in the zero-measurement-noise limit. *This is also why ASCA is so brittle in practice: it is the special case of the model that has* no *mechanism left to express doubt about a single misclassified leakage sample once $\beta \to \infty$ has been taken.*

Remark 2 (TASCA is a MAP-restriction of SASCA). *Writing $w_i = -\log \phi_i$, TASCA's objective* $\arg\min_{K,X} \sum_i w_i(X_i)$ *subject to $\Phi_C(K, X; P) = 1$ is, term for term,* $\arg\max_{K,X} \log\big[\Phi_C(K, X; P) \cdot \Phi_L(X; \boldsymbol{\ell})\big] = \arg\max_{K,X} p(K, X \mid \boldsymbol{\ell}, P)$*:* exactly *SASCA's MAP query, on exactly SASCA's joint distribution (Equation (1)), restricted to weights w_i that approximate $-\log p(\ell_i \mid X_i)$ rather than using the full likelihood at every step of the search. TASCA and SASCA are therefore not two different models of the attack surface; they are the same model, queried differently – a single point estimate obtained by combinatorial search, versus a full marginal obtained by message passing. This also explains a recurring empirical observation in the literature: TASCA-style solvers can succeed on instances where naive SASCA message passing struggles to converge, because MAP on a weighted CSP is a well-posed combinatorial search with mature, complete solvers, whereas MAR via loopy BP inherits no such completeness guarantee (Remark 3).*

Remark 3 (ExSASCA is SASCA with its solver made exact). *SASCA and ExSASCA use the identical representation ($\phi_i = p(\ell_i \mid X_i)$, full likelihoods) and pose the identical query (MAR, optionally MAP/MPE) against the identical joint distribution (Equation (1)). They differ* only *in the algorithm used*

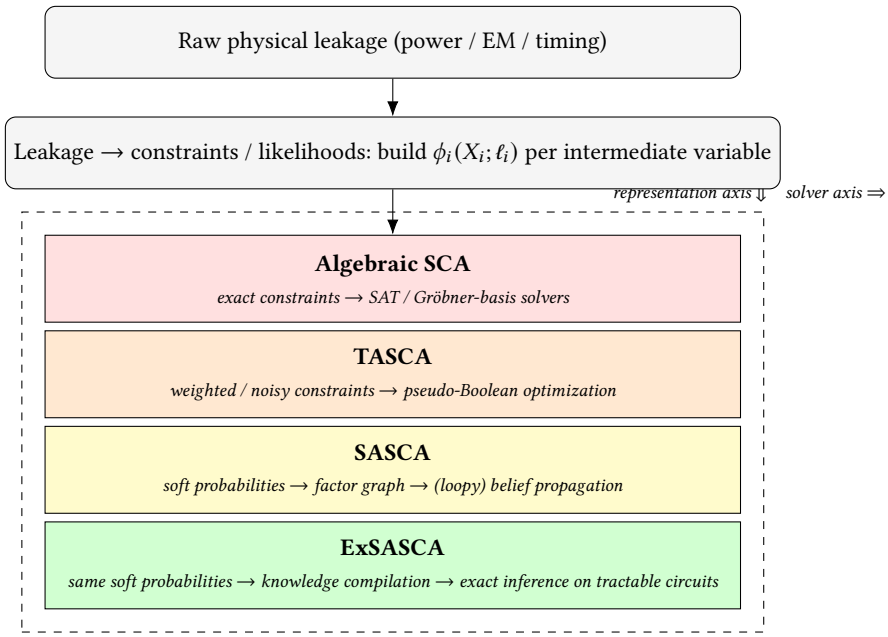

Fig. 1. The taxonomy of constraint-based SCA as a pipeline. Descending the stack retains progressively more of the leakage likelihood (representation axis: hard ⟶ weighted ⟶ probabilistic) and moves from combinatorial satisfiability/optimization to probabilistic inference, approximate and then exact (solver axis). All four bands operate on the same circuit-induced factor graph; see Table 1 and the reductions of Section 3.2.

to answer that query on a graph G that is not a tree: SASCA runs sum-product message passing directly on G, which is exact if G is a tree and an approximation with no general convergence or correctness guarantee otherwise [15] – indeed it is straightforward to construct instances on which loopy BP fails outright despite the underlying inference problem being easy [41]. ExSASCA instead first compiles the loopy sub-graph of G into a tractable circuit (Section 7) and then runs the same *sum-product operations – factor multiplication and marginalization – on that circuit, where they are exact by construction because the compiled representation is decomposable/tree-shaped regardless of how loopy the original G was. A useful corollary, exploited explicitly by ExSASCA's authors, is that the compiled circuit's* structure *depends only on the target algorithm g_i (fixed, public, leakage-independent), while only the numerical values plugged into it – the $\phi_i(X_i; \ell_i)$ – depend on a given trace; compilation is therefore a one-time cost amortized over every subsequent trace or key hypothesis, rather than a per-trace cost as in ASCA's or TASCA's per-instance solve.*

Chained together, Remarks 1–3 give exactly the claim advertised in Section 1: descending Figure 1's stack corresponds to (i) un-collapsing the leakage factor from a hard indicator to a weight to a full distribution, while (ii) moving the solver from combinatorial satisfiability, through combinatorial optimization, through approximate probabilistic inference, to exact probabilistic inference – all four stages acting on one factor graph fixed by the target algorithm. The remainder of this survey treats each stage's literature in turn (Sections 4–7), then returns to the practical consequences of this unification for security evaluation (Section 8).

4 Algebraic Side-Channel Analysis

4.1 From Scoring to Solving

Renauld, Standaert, and Veyrat-Charvillon's original algebraic side-channel attack (ASCA) [31, 33] is best understood as a change of question. A divide-and-conquer adversary asks, for each key chunk independently, "which value best explains the leakage I see at the points that depend on this chunk?" ASCA instead asks, of the *whole* algorithm at once, "which assignment to every key bit and every intermediate bit is *consistent* with both the cipher's definition and everything the leakage rules out?" Concretely, ASCA encodes the target cipher as a system of equations over GF(2) (a Boolean circuit, typically fed to a SAT solver after Tseitin-style clausification) or directly over $\mathrm{GF}(2^8)$ (an algebraic system, amenable to Gröbner-basis elimination), and *augments* that system with additional equations asserting, for each instrumented intermediate variable X_i, that X_i takes a value consistent with the observed leakage – in our notation, $\phi_i(X_i; \ell_i) = \mathbb{1}\{X_i \in S(\ell_i)\}$, where $S(\ell_i)$ is a set of "possible" values obtained by thresholding or otherwise hard-deciding $p(\ell_i \mid X_i)$ (e.g., every byte whose Hamming weight matches an estimated Hamming weight from a power measurement).

Solving the combined system – cipher equations *and* leakage equations together – is what makes ASCA powerful: a leakage constraint on one byte can, propagated through the cipher's own algebraic relations, sharply narrow the feasible set for bytes many rounds away, in a way no single-point distinguisher could discover on its own. This is also precisely the Sat query of Definition 1 applied to $\phi_i(X_i; \ell_i) = \mathbb{1}\{X_i \in S(\ell_i)\}$: ASCA does not rank or score candidate keys at all, it merely asks a satisfiability solver to find the (ideally unique) assignment that violates nothing.

4.2 Strengths, Weaknesses, and Follow-on Work

The practical payoff, demonstrated on PRESENT and on AES [33], is that ASCA can succeed from very few traces – in the noiseless limit, sometimes a single one – because it exploits every algebraic relation the cipher provides rather than a handful of hand-picked points of interest. Renauld and Standaert's follow-up work formalized this appeal to structure through a notion of *algebraic immunity* that bounds the complexity of the Gröbner-basis computation as a function of both the cipher's algebraic structure and the leakage model [31], and separately studied how the choice of intermediate-variable representation, leakage function, and target cipher each affect attack cost [32]; Mohamed et al. pushed the Gröbner-basis side of the same idea further, giving improved elimination strategies for AES [20].

ASCA's weakness is the direct cost of the same design choice that gives it its strength: because ϕ_i is a hard indicator, a *single* incorrectly classified leakage sample – one byte whose true value falls outside $S(\ell_i)$ due to ordinary measurement noise – renders the entire combined system unsatisfiable, and the solver reports failure with no further information about which constraint was at fault or how close the true key was to being found. Renauld et al. already observed empirically that realistic noise levels break the naive formulation [33]; the natural fix – keep the algebraic structure of ASCA, but stop treating leakage-derived constraints as infallible – is exactly what TASCA does next.

5 TASCA: Tolerant / Weighted Algebraic Side-Channel Analysis

5.1 From Hard Constraints to Pseudo-Boolean Costs

Oren, Kirschbaum, Popp, and Wool's Tolerant Algebraic Side-Channel Analysis (TASCA) [21] keeps ASCA's central move – encode the cipher as an exact algebraic/Boolean system – but replaces its all-or-nothing leakage constraints with *weighted* ones. Instead of declaring a byte's value "possible" or "impossible," TASCA attaches a cost $w_i(X_i)$ to every candidate value of every instrumented variable, derived from (a quantized approximation of) $-\log p(\ell_i \mid X_i)$, and asks for

the assignment that minimizes total cost subject to the cipher's equations still holding exactly. In the notation of Definition 1, the circuit factors Φ_C remain hard, but the leakage factor becomes $\phi_i(X_i; \ell_i) = \exp(-w_i(X_i))$, and the query changes from SAT to MAP: find $\arg\max_{K,X} p(K, X \mid \boldsymbol{\ell}, P)$ rather than merely a feasible point. Because this is now a constrained combinatorial *optimization* problem rather than a satisfiability problem, one bad leakage sample no longer collapses the whole system – it simply costs a little, and the solver can still find the assignment that best balances that cost against every other constraint. This is exactly Remark 1's reduction run in reverse: TASCA *is* ASCA with the zero-temperature limit undone.

5.2 Solvers and Extensions

TASCA's original instantiation used pseudo-Boolean (PB) optimization solvers [21], and the same authors' journal-length treatment generalizes the approach and evaluates it on full-size AES [23, 24], framing side-channel key recovery in general as pseudo-Boolean optimization over a cost function assembled from leakage. Oren, Renauld, Standaert, and Wool subsequently extended TASCA-style weighting beyond the Hamming-weight leakage model to arbitrary profiled leakage functions [22], and a parallel line of work reformulated the same weighted-CSP problem for other solver back-ends and error models, including SAT-based error-tolerant variants [43] and an enhanced model (MDASCA) supporting alternative leakage models and improved noise handling [44]. Later work has also cast the identical optimization problem as constraint programming (CP), which brings a different, often more scalable, family of propagation-based solvers to bear on the same weighted system.

5.3 What TASCA Buys, and What It Still Cannot Give

TASCA closes ASCA's most damaging weakness – catastrophic, uninformative failure under realistic noise – while keeping ASCA's chief strength, the ability to exploit the cipher's full algebraic structure rather than a handful of independently scored points. What it does not provide is anything beyond a single point estimate: TASCA returns *a* most-probable key, with no accompanying notion of how much more probable it is than the runner-up, and hence no principled way to rank remaining candidates if that single guess is wrong. As Remark 2 makes precise, this is not an incidental limitation of any particular TASCA solver but a direct consequence of asking a MAP query rather than a MAR query of the underlying model – to get a posterior (and therefore a ranking, a confidence level, or a rate at which to safely stop trusting a single guess), the leakage factor itself must stop being collapsed into a scalar weight and be retained as a full distribution. That is precisely SASCA's move, taken up next.

6 SASCA: Soft Analytical Side-Channel Analysis

6.1 From a Point Estimate to a Posterior

Veyrat-Charvillon, Gérard, and Standaert's Soft Analytical Side-Channel Attack (SASCA) [39] takes the step that Section 5 closed with: stop collapsing $p(\ell_i \mid X_i)$ into a scalar cost, and instead attach the *entire* likelihood function – exactly the belief $p(v \mid \ell)$ that a profiled template attack already produces (Section 2) – to every instrumented variable in the circuit's factor graph. The original paper frames this explicitly as a change of representation: rather than a system of equations, express the cipher and its leakage as a *code*, in the spirit of low-density parity-check codes, and decode it with belief propagation rather than solve it with a SAT or optimization solver [39]. Guo, Grosso, Standaert, and Bronchain later made this coding-theoretic reading precise and systematic, formalizing SASCA's factor graph as a code over the algorithm's data flow and analyzing its decoding behavior accordingly [9]. In the language of Definition 1, this is $\phi_i(X_i; \ell_i) = p(\ell_i \mid X_i)$

with no collapse whatsoever, and a MAR query: recover the full marginal posterior $p(K_j \mid \boldsymbol{\ell}, P)$ of each secret sub-variable, which can then be ranked, thresholded, or fed to a key-enumeration algorithm [27, 37] exactly as a divide-and-conquer attack's per-byte scores would be, but now informed by every leaking point in the computation at once.

6.2 Belief Propagation on the Circuit's Factor Graph

Mechanically, SASCA builds the factor graph of Equation (1) directly out of the target algorithm's operations – an XOR becomes a ternary indicator factor, an S-box or an NTT butterfly becomes a lookup-table factor, a modular reduction becomes an arithmetic factor – attaches a template-derived likelihood factor ϕ_i to every node the adversary has leakage for, and runs the sum-product algorithm [19]: messages are passed along every edge, each factor multiplies its incoming messages and marginalizes out every variable but the one the outgoing message targets, and each variable node multiplies its incoming factor messages to form a current belief. On a tree-shaped factor graph this converges in one forward-backward sweep to the *exact* marginals; the trouble is that essentially no cipher of practical interest induces a tree. AES's MIXCOLUMNS step alone couples many bytes through a small dense linear system, and lattice-based NTTs are built from repeated butterfly units whose shared inputs and outputs create short cycles throughout the graph. On such *loopy* graphs, belief propagation is only ever an approximation, and a well-documented one: its fixed points correspond to stationary points of the Bethe free energy rather than to the true marginals, it is not guaranteed to converge at all, and – most inconveniently for a security evaluator – convergence does not imply correctness, so it is possible to construct instances where an easy inference problem causes loopy BP to fail outright [15, 41]. We return to the consequences of this fact for security evaluation in Section 8.

6.3 Extensions to Masked and Lattice-Based Implementations

SASCA's ability to combine many leakage points coherently makes it a natural fit for exactly the two settings identified in Section 2.3: masked implementations, where no single point carries first-order information, and single-trace public-key settings, where there is no averaging to fall back on. Both concerns converge in NTT-based lattice cryptography. Primas, Pessl, and Mangard gave the first single-trace attack on masked lattice-based encryption, combining template-derived beliefs over an unprotected NTT's intermediate coefficients with belief propagation and a final lattice-reduction step, and explicitly noting that the attack survives masking because it never needs to isolate a single share [28]. Pessl and Primas refined the underlying factor-graph construction, introducing specialized butterfly-node factors designed to avoid the short cycles that most degrade belief-propagation convergence in an NTT graph [25]. Hamburg et al. combined a similar belief-propagation attack on Kyber's decryption-side NTT with a chosen-ciphertext strategy to improve noise tolerance against masked implementations [10], and Hermelink, Streit, Strieder, and Thieme further adapted the belief-propagation construction to remain effective when the NTT's operand order is additionally randomized by shuffling, a countermeasure intended to disrupt exactly this kind of combination attack [11]. Carrera Rodriguez, Bruguier, Valea, and Benoit evaluated SASCA directly against *local masking* of NTT twiddle factors – a countermeasure proposed specifically to defeat SASCA – by extending the factor graph with explicit masking nodes, and found that the countermeasure's security scales only gradually (and expensively) with the number of masks used per layer [1]. Kannwischer, Pessl, and Primas showed the same style of attack applies beyond the NTT itself, to the Keccak permutation used pervasively as a PRF/PRNG inside lattice-based schemes [13]. Most recently, Wang et al.'s Neur-SASCA replaces SASCA's hand-built templates with an end-to-end neural model that predicts intermediate-value distributions directly from raw NTT traces, reporting substantially higher noise tolerance and full recovery from far fewer

profiling traces than classical template-based SASCA on physical hardware [40], illustrating that the representation axis of Section 3 (how ϕ_i is built) and the profiling methodology that feeds it are improving in step with the solver axis.

6.4 The Scalability Ceiling

SASCA's combination of tolerance to noise, applicability under masking, and single-trace capability makes it, alongside its algebraic and weighted predecessors, one of the standard tools for evaluating lattice-based implementations today. It inherits two costs for this generality. First, as already noted, loopy belief propagation's approximation is uncertified: an unsuccessful SASCA attack is not evidence that an implementation is secure, only that this particular approximate solver did not succeed on it (Section 8). Second, the size of the messages passed during belief propagation grows with the domain of the variables involved, and full-precision, multi-layer NTTs push this cost far past what a naive implementation can hold in memory: Sun et al. report that standard SASCA with FFT-accelerated message passing already requires on the order of 122 GB of memory for a single unprotected six-layer NTT sub-component of ML-DSA, before masking further inflates the graph [35]. Both costs point toward the same remedy pursued by the next paradigm and, from a different angle, by scalable-inference work such as Sun et al.'s region-wise pruning [35]: replace or restructure the solver, while keeping SASCA's representation exactly as it is.

7 ExSASCA: Exact Soft Analytical Side-Channel Analysis

7.1 The Certificate Problem

SASCA's central practical weakness is epistemic rather than statistical: because loopy belief propagation carries no general convergence or correctness guarantee, a security evaluator who runs SASCA against a candidate implementation and fails to recover the key learns very little. The failure might mean the implementation is genuinely hard to break; it might equally mean that the true posterior was easy to compute and loopy BP simply failed to compute it [15, 41]. Wedenig, Nagpal, Cassiers, Mangard, and Peharz's Exact Soft Analytical Side-Channel Attack (ExSASCA) [41] directly targets this gap: keep SASCA's representation exactly as is – full likelihood factors, the same factor graph, the same MAR/MAP queries – but replace the uncertified approximate solver with one that is exact by construction, so that a failure to recover the key is, for the first time in this lineage, an actual certificate about the leakage rather than an artifact of the solver.

7.2 Knowledge Compilation and Tractable Probabilistic Circuits

The tool ExSASCA borrows for this purpose comes from the knowledge-compilation literature [7], whose central idea is to trade an expensive *compilation* step, performed once, for cheap, repeated *querying* afterward. A Boolean formula – in ExSASCA's case, the sub-circuit of the target cipher spanned by a loopy portion of its factor graph – is compiled into a different target representation on which the queries of interest (satisfiability, model counting, and, in the probabilistic extension used here, marginalization and most-probable-explanation) can be answered in time polynomial in the *compiled representation's* size, however large that size turns out to be. ExSASCA uses the sentential decision diagram (SDD) [6] as its compilation target, and its probabilistic extension, the probabilistic SDD (PSDD) [14] – a special case of the broader family of tractable probabilistic circuits [5, 36], which support factor multiplication, marginalization, and most-probable-explanation queries in time linear or (for multiplication) worst-case quadratic in circuit size, provided the circuits involved are *compatible* (respect a common variable order, or *vtree*). Reducing exact probabilistic inference to compilation followed by cheap querying in this way is closely related to weighted model counting [3]

and to dynamic SDD minimization techniques for keeping the compiled representation small during construction [4].

7.3 The Construction, on AES

Wedenig et al. instantiate ExSASCA on AES-128's first round, whose loopy sub-graph is entirely contained within MixColumns – a 4×4-byte, GF(2^8)-linear operation coupling 21 byte-valued variables (168 bits) with many short cycles. Rather than running belief propagation on this loopy sub-graph directly, as SASCA does, they compile MixColumns's Boolean encoding (a CNF of 648 clauses) into a single SDD of roughly 19,000 sum/product nodes, effectively replacing the entire loopy region with one very large – but tractable – factor, on which the surrounding factor graph is again a tree and belief propagation is exact [41]. Two compilation strategies are given for combining this compiled MixColumns factor with the per-trace leakage likelihoods ϕ_i: an "out-of-the-box" approach that multiplies the likelihoods into the circuit directly, exploiting the fact that unprotected AES's per-byte beliefs are typically concentrated on a handful of Hamming-weight-equivalent values (a sparsity assumption that keeps the product circuit small); and a more general *weighted model counting* formulation, which introduces auxiliary byte-indicator variables and answers each belief-propagation message as a WMC query, at higher compilation cost but without any sparsity assumption – and hence still usable when countermeasures inflate the entropy of every intermediate byte and make the sparse approach ineffective.

7.4 Results and What They Show

On real power traces from an AES-128 implementation, ExSASCA improves top-1 key-recovery success rate by more than 31 percentage points (absolute) over conventional SASCA, at roughly the same computational cost as SASCA under the sparsity assumption, and using six times fewer operations than exhaustive (and therefore certainly exact) enumeration even without any sparsity assumption at all [41]. Two further findings are especially relevant to the unification argued for in Section 3. First, computing the exact Map solution directly over the *joint* key posterior (ExSASCA+MPE) outperforms computing exact per-byte marginals and combining them under an independence assumption (ExSASCA+MAR) – both are legitimate exact queries against the same model, and the choice between a Mar-style and a Map-style read-out remains meaningful even once the underlying inference is exact, mirroring the SASCA/TASCA distinction of Remark 2 one level up the exactness axis. Second, and most tellingly for the loopy-BP certificate problem motivating this section, as the leakage is made progressively less informative (simulating stronger protection), SASCA's success rate collapses toward the baseline while both ExSASCA variants continue to succeed at a substantial rate well past the point where SASCA has effectively given up [41] – direct empirical confirmation that at least some of SASCA's apparent resistance on noisier or better-protected targets is an artifact of the approximate solver rather than a property of the leakage itself.

7.5 Where Exactness Runs Out

ExSASCA's guarantee is exact inference on the compiled circuit, not unconditional tractability: compiled-circuit size can, in the worst case, grow exponentially in the treewidth of the loopy sub-graph being compiled, and Wedenig et al. note this explicitly as an opening for future work connecting circuit-complexity lower bounds [41] to the design of cryptographic circuits that provably resist compilation to a small tractable representation. This is precisely the tension already visible in SASCA's scalability ceiling on large lattice-based constructions (Section 6.3): the same treewidth that makes belief propagation loopy and unreliable is what makes exact compilation

Table 2. Cross-paradigm trade-offs. "Traces" indicates the typical minimum number of measurements the paradigm is designed to work from; "masking robustness" indicates whether the paradigm can, in principle, combine shares of a masked implementation rather than being blocked by masking outright; "failure certifies security" indicates whether an unsuccessful run is sound evidence that the modeled leakage does not suffice for key recovery.

Paradigm	**Dominant cost driver**	**Noise tolerance**	**Masking robustness**	**Failure certifies security?**
Algebraic SCA	size/difficulty of the SAT or Gröbner instance; blows up on any misclassification	very low – a single wrong hard constraint makes the system UNSAT	yes in principle (shares are just more variables), but hard-decision leakage on noisy shares is exactly what breaks first	no – UNSAT may just mean one constraint was wrong
TASCA	size/difficulty of the weighted CSP / PB instance	moderate – costs absorb noise, but only through a scalar, often quantized, weight	yes, more robustly than ASCA, since share-level costs can be small without being fatal	no – optimum may simply not have been found in the time given
SASCA	message size and iteration count of loopy BP; grows with variable domains and graph density	good – operates directly on full likelihoods	yes – explicitly designed for masked, single-trace settings (Section 6.3)	no – non-convergence or a bad fixed point look identical to genuine security
ExSASCA	compiled-circuit size, worst case exponential in treewidth of the loopy region	good – inherits SASCA's representation exactly	yes, subject to the compiled circuit remaining tractable at the countermeasure's induced treewidth	yes, whenever compilation succeeds: an exact MAR/MAP result that fails to recover the key is a sound statement about the modeled leakage

expensive. We return to this tension, and to what a treewidth-aware notion of implementation security might look like, in Sections 8 and 10.

8 Cross-Paradigm Trade-offs and Security Evaluation

Sections 4–7 traced one axis of comparison in detail: how each paradigm's choice of representation and solver trades exactness for robustness to noise. This section draws the trade-offs together (Table 2) and turns to a question that matters as much to a defender evaluating an implementation as it does to an attacker mounting one: given that a particular constraint-based attack *failed* to recover the key, what – if anything – has been shown about the implementation's security?

8.1 What a Failed Attack Is Allowed to Prove

Security evaluation has long had to grapple with the gap between "no attack I ran succeeded" and "no attack could succeed." For divide-and-conquer attacks, this gap was closed constructively: Standaert, Malkin, and Yung's unified framework for side-channel key-recovery evaluation [34],

together with the key-rank and key-enumeration algorithms that followed [27, 37, 38], let an evaluator who cannot fully break a device still compute a sound bound on the residual work an attacker would need – turning "my attack didn't recover the key" into a quantitative, defensible statement, precisely because the underlying per-chunk scoring step is well understood and its output (a list of per-chunk scores) is exactly what a rank-estimation algorithm consumes. Constraint-based SCA does not yet have an equally settled answer to the same question, and Table 2's last column shows why the answer differs so sharply across paradigms.

For ASCA and TASCA, a failed run (UNSAT, or an optimum that does not match the true key) is uninformative by construction: both reduce to a single combinatorial search whose failure could reflect a genuinely hard instance or nothing more than one noisy measurement. For SASCA, the situation is subtler and, in a sense, worse, because loopy belief propagation *usually* works well enough to be useful, which makes its occasional and unpredictable failures easy to mistake for a security margin the implementation does not actually have – exactly the failure mode Section 7 documented empirically, where SASCA's success rate collapsed well before ExSASCA's did on identical, less-informative leakage [41]. ExSASCA is, to date, the only paradigm in this lineage for which a negative result is sound: because its query is answered exactly against the declared model (Equation (1)), a failure to recover the key genuinely means that model's leakage does not determine the key, *conditioned on compilation having succeeded at all.* That final caveat is important and is where Table 2's first column bites hardest – a compilation that runs out of memory or time before finishing has, once again, told the evaluator nothing, which is exactly why Section 10 treats scaling exact compilation to larger, more heavily protected circuits as the central open problem of the whole design space, and why we suggest below that exact, certified inference of the kind ExSASCA provides is the natural target for a rank-estimation-style evaluation methodology purpose-built for constraint-based (rather than purely divide-and-conquer) attacks.

8.2 Implications for Countermeasure Design

The same trajectory that sharpens the attacker's tools also sharpens what a defender should assume about them. Noise alone – the implicit countermeasure ASCA already breaks under only very light realistic conditions – was never a serious defense once TASCA existed to absorb it. Masking alone stops *divide-and-conquer* attacks by design (Section 2.3) but was shown, repeatedly, not to stop SASCA-style combination once the masking scheme's own structure is added back into the factor graph as extra variables and factors [1, 10, 28]. The natural next question for a countermeasure designer, in light of Section 7, is whether a masked circuit can be deliberately structured to have *large treewidth* in its induced factor graph – resisting not merely today's loopy-BP-based attacks but tomorrow's exact, compiled ones – a design goal with no settled name or methodology yet, which we return to as an open problem in Section 10.

9 Case Studies

9.1 AES: One Target, Four Paradigms

AES is the only target attacked in the literature by all four paradigms surveyed here under broadly comparable conditions, which makes it a useful lens on Table 2 in concrete terms rather than in the abstract. Algebraic SCA and its Gröbner-basis variants recover AES keys from very few traces in low-noise settings but degrade sharply once realistic measurement error is introduced [20, 33]; TASCA and its pseudo-Boolean, SAT-based, and constraint-programming descendants restore robustness to that same noise while keeping the cipher's algebraic structure intact, at the cost of returning only a single best guess [21, 23, 43, 44]; SASCA turns the single AES round targeted by essentially all of this literature into a full posterior over each key byte via belief propagation on a

factor graph built directly from SubBytes, ShiftRows, MixColumns, and AddRoundKey [9, 39]; and ExSASCA, attacking the identical single-round factor graph and the identical MixColumns bottleneck, recovers the key exactly where SASCA's loopy approximation had been leaving more than thirty percentage points of achievable success rate on the table [41]. The four results are not four separate case studies so much as one case study viewed through Table 1's four lenses: the same circuit, the same class of leakage, and – per Section 3 – the same underlying joint distribution, answered with progressively more of its leakage evidence retained and progressively more of the resulting query answered exactly.

9.2 NTT-Based Post-Quantum Cryptography

Lattice-based schemes standardized or advanced under NIST's post-quantum process – ML-KEM/Kyber, ML-DSA/Dilithium, and FN-DSA/Falcon – lean heavily on the Number-Theoretic Transform for efficient polynomial multiplication, and, as Section 6.3 detailed, the NTT's regular butterfly structure has made it the primary battleground for SASCA-family attacks on this generation of cryptography [1, 10, 11, 13, 25, 28, 40], precisely because single-trace, mask-penetrating combination is not a convenience here but a necessity: ephemeral randomness in encapsulation and signing rules out the repeated-measurement averaging that classical DPA relies on, and masking is the standard countermeasure these schemes deploy against exactly the kind of leakage SASCA is built to exploit. The countermeasure side of this arms race – local masking of NTT twiddle factors, operand shuffling – has itself been evaluated primarily by asking how much it degrades a SASCA-style attacker [1, 11], which is a direct, literal application of the "implications for countermeasure design" discussion of Section 8: the countermeasure's designers are, whether or not they use this survey's vocabulary, reasoning about the treewidth their masking gadget adds to the induced factor graph.

FN-DSA/Falcon's key-generation and signing procedures add a second high-value target beyond the NTT: discrete Gaussian sampling. Qiu and Aysu's SHIFT SNARE attack recovers Falcon's full secret key from a single power trace by exploiting a single leaking instruction – a 63-bit right-shift used to determine the sign of a sampled coefficient – reaching a per-coefficient success rate above 99.9999999% and full-key recovery above 99.9999% on an ARM Cortex-M4 running FALCON's NIST reference and optimized implementations [30]. It is worth being precise about where this attack sits relative to the taxonomy of Section 3: SHIFT SNARE is a classical, single-point template attack on one especially strong leakage source, not a factor-graph combination attack over the sampler's full data flow, and it is exactly the kind of target – a sequential pipeline of comparisons, shifts, and sign decisions with strong internal dependencies – for which a SASCA- or ExSASCA-style treatment of the *entire* sampler (rather than its single most-leaking instruction) remains, to our knowledge, an open extension rather than published fact. It is also, for the same reason, a natural stress test for the countermeasures now being proposed at the sampler level: recent masked discrete Gaussian samplers such as MAGNET build the sampler out of explicit masked gadgets (bitwise negation, increment, and squaring, denoted `SecNEG`, `SecINC`, and `SecSQU`) [42], and each such gadget is, in the language of this survey, additional circuit structure g_i that a future combination attack on the sampler would need to fold into its factor graph exactly as SASCA folded masking nodes into the NTT's [1]. Whether the resulting graph remains tractable for exact ExSASCA-style compilation, or is better attacked (or defended) by reasoning about its treewidth directly, is precisely the kind of question Section 10 takes up next.

10 Open Problems

The unification of Section 3 does more than tidy up terminology: it makes clear that several open problems which look, on the surface, specific to one paradigm are really open problems about the shared factor graph, visible from every paradigm at once. We highlight six.

Scaling exact compilation past small treewidth. ExSASCA's guarantee is exact inference in time polynomial in compiled-circuit size, and that size is, in the worst case, exponential in the treewidth of the loopy region being compiled [41]. AES's single-round MixColumns bottleneck compiles to a manageable circuit; a masked, multi-layer lattice-based NTT does not yet, and even SASCA's approximate solver already requires on the order of a hundred gigabytes of memory on a single unprotected sub-component of ML-DSA [35]. Whether region-wise decomposition strategies of the kind Sun et al. propose for loopy BP [35] can be adapted to *exact* compilation – compiling and gluing together many small tractable regions rather than one large one – is, in our view, the single highest-value open problem raised by this survey's formulation.

Convergence guarantees where exact compilation is not yet feasible. Wherever exact compilation remains out of reach, loopy belief propagation is not going away, and the field still lacks general, structural conditions under which it is guaranteed to converge to the correct marginals on the specific families of factor graphs induced by masked NTTs and similar constructions – as opposed to the case-by-case, empirically-validated confidence the current literature offers [11, 15, 25, 28]. A convergence theory tailored to this graph family, even a conditional or SNR-threshold-style one, would let evaluators use SASCA's cheaper solver with something closer to ExSASCA's certificate.

Treewidth as a countermeasure design target. Section 8 suggested that a masking scheme's real target, in light of ExSASCA, is not merely to reduce the signal-to-noise ratio available to an attacker but to keep the induced factor graph's treewidth high enough to resist tractable compilation altogether. Turning that suggestion into an actual design methodology – a treewidth (or compiled-circuit-size) budget that a masking gadget can be verified against, analogous to how the d-probing model [12] gives circuit designers a checkable target today – is, to our knowledge, unexplored.

Unifying fault-derived and leakage-derived constraints. This survey has treated leakage as the sole source of evidence, but fault-injection analysis contributes evidence of exactly the same two flavors this survey's representation axis already distinguishes: an observed faulty output is often a hard algebraic constraint (in the tradition of differential fault analysis), while a fault's physical parameters more realistically induce a distribution over which fault actually occurred. A framework that places fault-derived and leakage-derived factors on the same factor graph – hard or weighted constraints from faults, soft likelihoods from leakage, combined and queried exactly as in Definition 1 – would be a direct, natural extension of the formulation in Section 3 rather than a new paradigm, and would let a single inference engine exploit combined fault-and-leakage adversaries that current tooling treats as two separate attack surfaces.

Learned representations feeding exact solvers. Neur-SASCA already shows that replacing hand-built templates with a learned model that predicts ϕ_i directly from raw traces substantially improves SASCA's noise tolerance on the NTT [40], and Wedenig et al. explicitly flag learned, neuro-symbolic leakage models as future work for ExSASCA [41]. Whether learned representations can be built to keep the resulting factor graph *compilable* – rather than only more informative – so that the two lines of improvement (better ϕ_i, better solver) compose rather than trade off, is open.

A shared evaluation methodology. Divide-and-conquer attacks converged, over roughly a decade, on key rank and enumeration as a common yardstick for comparing attacks and bounding residual security [27, 34, 37, 38]. Constraint-based SCA has no analogous common yardstick yet: ASCA/TASCA results are typically reported as success/failure or time-to-solve, SASCA results as top-k success rate, and ExSASCA results as both success rate and compiled-circuit size, which makes the cross-paradigm comparisons this survey draws in Table 2 more laborious to assemble from the literature than they should be. A standard benchmark suite – fixed targets, fixed leakage models, and a

reporting format that records representation, query, solver, exactness, and (where applicable) compiled-circuit size side by side – would let the field track progress along the two axes of Section 3 directly, rather than through paradigm-specific proxies.

11 Conclusion

Fifteen years of work on constraint-based side-channel analysis have often been read as a sequence of increasingly sophisticated attacks, each superseding the last. This survey has argued for a different reading: ASCA, TASCA, SASCA, and ExSASCA are one attack, on one underlying latent-secret inference problem defined by a target algorithm's factor graph and the leakage attached to it, explored along exactly two axes – how much of the leakage likelihood is retained (hard, weighted, or fully probabilistic) and how the resulting query is answered (satisfiability, combinatorial optimization, approximate message passing, or exact tractable inference). Made precise, this reading yields three concrete reductions (Remarks 1–3) rather than a family resemblance, and it clarifies a question that matters well beyond taxonomy: what a given paradigm's *failure* is entitled to say about an implementation's security. Only the most recent paradigm, exact inference via knowledge-compiled tractable circuits, currently earns the right to say very much at all – and doing so at the scale of a modern masked, lattice-based implementation, rather than a single AES round, is the frontier this survey leaves open.

References

[1] Rafael Carrera Rodriguez, Florent Bruguier, Emanuele Valea, and Pascal Benoit. 2025. Cracking the Mask: SASCA Against Local-Masked NTT for CRYSTALS-Kyber. *Communications in Cryptology* 2, 2 (2025).

[2] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. 2002. Template Attacks. In *Cryptographic Hardware and Embedded Systems – CHES 2002 (LNCS, Vol. 2523)*. Springer, 13–28.

[3] Mark Chavira and Adnan Darwiche. 2008. On Probabilistic Inference by Weighted Model Counting. *Artificial Intelligence* 172, 6-7 (2008), 772–799.

[4] Arthur Choi and Adnan Darwiche. 2013. Dynamic Minimization of Sentential Decision Diagrams. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*. 187–194.

[5] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. 2020. *Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Models.* Technical Report. UCLA.

[6] Adnan Darwiche. 2011. SDD: A New Canonical Representation of Propositional Knowledge Bases. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*. 819–826.

[7] Adnan Darwiche and Pierre Marquis. 2002. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research* 17 (2002), 229–264.

[8] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. 2001. Electromagnetic Analysis: Concrete Results. In *Cryptographic Hardware and Embedded Systems – CHES 2001 (LNCS, Vol. 2162)*. Springer, 251–261.

[9] Qian Guo, Vincent Grosso, François-Xavier Standaert, and Olivier Bronchain. 2020. Modeling Soft Analytical Side-Channel Attacks from a Coding Theory Viewpoint. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 4 (2020), 209–238.

[10] Mike Hamburg, Julius Hermelink, Robert Primas, Simona Samardjiska, Thomas Schamberger, Silvan Streit, Emanuele Strieder, and Christine Van Vredendaal. 2021. Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2021, 4 (2021), 88–113.

[11] Julius Hermelink, Silvan Streit, Emanuele Strieder, and Katharina Thieme. 2023. Adapting Belief Propagation to Counter Shuffling of NTTs. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2023, 1 (2023), 60–88.

[12] Yuval Ishai, Amit Sahai, and David Wagner. 2003. Private Circuits: Securing Hardware against Probing Attacks. In *Advances in Cryptology – CRYPTO 2003 (LNCS, Vol. 2729)*. Springer, 463–481.

[13] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. 2020. Single-Trace Attacks on Keccak. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020, 3 (2020), 243–268.

[14] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. 2014. Probabilistic Sentential Decision Diagrams. In *Principles of Knowledge Representation and Reasoning (KR 2014)*.

[15] Christian Knoll. 2022. Understanding the Behavior of Belief Propagation. arXiv:2209.05464.

[16] Paul C. Kocher. 1996. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology – CRYPTO 1996 (LNCS, Vol. 1109)*. Springer, 104–113.

[17] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Advances in Cryptology – CRYPTO 1999 (LNCS, Vol. 1666)*. Springer, 388–397.

[18] Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

[19] Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. 2001. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory* 47, 2 (2001), 498–519.

[20] Mohamed Saied Emam Mohamed, Stanislav Bulygin, Michael Zohner, Annelie Heuser, and Michael Walter. 2012. Improved Algebraic Side-Channel Attack on AES. Cryptology ePrint Archive, Report 2012/084.

[21] Yossef Oren, Mario Kirschbaum, Thomas Popp, and Avishai Wool. 2010. Algebraic Side-Channel Analysis in the Presence of Errors. In *Cryptographic Hardware and Embedded Systems – CHES 2010 (LNCS, Vol. 6225)*. Springer, 428–442.

[22] Yossef Oren, Mathieu Renauld, François-Xavier Standaert, and Avishai Wool. 2012. Algebraic Side-Channel Attacks Beyond the Hamming Weight Leakage Model. In *Cryptographic Hardware and Embedded Systems – CHES 2012 (LNCS, Vol. 7428)*. Springer, 140–154.

[23] Yossef Oren and Avishai Wool. 2012. Tolerant Algebraic Side-Channel Analysis of AES. Cryptology ePrint Archive, Report 2012/092.

[24] Yossef Oren and Avishai Wool. 2016. Side-Channel Cryptographic Attacks Using Pseudo-Boolean Optimization. *Constraints* 21, 4 (2016), 616–645.

[25] Peter Pessl and Robert Primas. 2019. More Practical Single-Trace Attacks on the Number Theoretic Transform. In *Progress in Cryptology – LATINCRYPT 2019 (LNCS)*. Springer.

[26] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. 2023. SoK: Deep Learning-based Physical Side-channel Analysis. *Comput. Surveys* 55, 11 (2023), 227.

[27] Romain Poussier, François-Xavier Standaert, and Vincent Grosso. 2016. Simple Key Enumeration (and Rank Estimation) Using Histograms: An Integrated Approach. In *Cryptographic Hardware and Embedded Systems – CHES 2016 (LNCS, Vol. 9813)*. Springer, 61–81.

[28] Robert Primas, Peter Pessl, and Stefan Mangard. 2017. Single-Trace Side-Channel Attacks on Masked Lattice-Based Encryption. In *Cryptographic Hardware and Embedded Systems – CHES 2017 (LNCS, Vol. 10529)*. Springer, 513–533.

[29] Emmanuel Prouff and Matthieu Rivain. 2013. Masking against Side-Channel Attacks: A Formal Security Proof. In *Advances in Cryptology – EUROCRYPT 2013 (LNCS, Vol. 7881)*. Springer, 142–159.

[30] Jinyi Qiu and Aydin Aysu. 2025. SHIFT SNARE: Uncovering Secret Keys in FALCON via Single-Trace Analysis. Cryptology ePrint Archive, Report 2025/146.

[31] Mathieu Renauld and François-Xavier Standaert. 2010. Algebraic Side-Channel Attacks. In *Information Security and Cryptology (Inscrypt 2009) (LNCS, Vol. 6151)*. Springer, 393–410.

[32] Mathieu Renauld and François-Xavier Standaert. 2010. Representation-, Leakage- and Cipher-Dependencies in Algebraic Side-Channel Attacks. In *Industrial Track of ACNS 2010*.

[33] Mathieu Renauld, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. 2009. Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In *Cryptographic Hardware and Embedded Systems – CHES 2009 (LNCS, Vol. 5747)*. Springer, 97–111.

[34] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. 2009. A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In *Advances in Cryptology – EUROCRYPT 2009 (LNCS, Vol. 5479)*. Springer, 443–461.

[35] Yiteng Sun, Zhuo Huang, Yan Zhuang, Shuo Sun, Xinyu Li, Yu Yu, and Weijia Wang. 2026. Efficient Soft Analytical Side-Channel Attacks on Large-Scale Cryptographic Computations. Cryptology ePrint Archive, Paper 2026/1811.

[36] Antonio Vergari, YooJung Choi, Robert Peharz, and Guy Van den Broeck. 2020. Probabilistic Circuits: Representations, Inference, Learning and Applications. AAAI Tutorial.

[37] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. 2013. An Optimal Key Enumeration Algorithm and its Application to Side-Channel Attacks. In *Selected Areas in Cryptography (SAC 2012) (LNCS, Vol. 7707)*. Springer, 390–406.

[38] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. 2013. Security Evaluations beyond Computing Power. In *Advances in Cryptology – EUROCRYPT 2013 (LNCS, Vol. 7881)*. Springer, 126–141.

[39] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. 2014. Soft Analytical Side-Channel Attacks. In *Advances in Cryptology – ASIACRYPT 2014 (LNCS, Vol. 8873)*. Springer, 282–296.

[40] Yiteng Wang, Jiayu Yu, Yifan Nie, Yuan Zhao, Xin Zhang, Yang Shi, Cheng Zhang, Zhe Wang, and Dawu Gu. 2026. NeurSASCA: High-Noise-Tolerant and Automated Single-Trace Attack against NTT. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2026, 3 (2026), 908–932.

[41] Thomas Wedenig, Rishub Nagpal, Gaëtan Cassiers, Stefan Mangard, and Robert Peharz. 2024. Exact Soft Analytical Side-Channel Attacks using Tractable Circuits. In *Proceedings of the 41st International Conference on Machine Learning (ICML) (PMLR, Vol. 235)*.

[42] Mohamed Yassi, Soundes Marzougui, Raymond K. Zhao, Muhammed F. Esgin, Amin Sakzad, and Ron Steinfeld. 2026. MAGNET: MAsked Gaussian Now Efficient and Table-less. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2026, 3 (2026), 769–804.

[43] Xinjie Zhao, Tao Wang, Shize Guo, Fan Zhang, Zhijie Shi, Huiying Liu, and Keke Wu. 2011. SAT Based Error Tolerant Algebraic Side-Channel Attacks. In *Conference on Cryptographic Algorithms and Cryptographic Chips (CASC)*.

[44] Xinjie Zhao, Fan Zhang, Shize Guo, Tao Wang, Zhijie Shi, Huiying Liu, and Keke Ji. 2012. MDASCA: An Enhanced Algebraic Side-Channel Attack for Error Tolerance and New Leakage Model Exploitation. In *Constructive Side-Channel Analysis and Secure Design (COSADE 2012) (LNCS, Vol. 7275)*. Springer, 231–248.