SoK: Deep Learning-based Physical Side-channel Analysis

Sengim Karayalçin
Leiden University

Marina Krček
Radboud University

Stjepan Picek
University of Zagreb Faculty of Electrical Engineering and Computing & Radboud University

Abstract

Deep learning-based side-channel analysis (DLSCA) represents a powerful paradigm for running side-channel attacks. DLSCA, in a state-of-the-art setting, can break multiple targets with a single attack trace, requiring minimal feature engineering. As such, DLSCA is an extremely active research area for both industry and academia. At the same time, due to domain activity, it becomes more difficult to understand the current trends and challenges. In this systematization of knowledge, we provide a critical overview of developments in DLSCA over the past years, enabling us to offer concrete suggestions and taxonomies. Moreover, we examine the reproducibility perspective and show that achieving it in full is difficult and requires a careful experimental setup.

1 Introduction

Cryptographic algorithms represent a core part of modern communication systems. While such algorithms are designed to be theoretically secure, once implemented, they face the threat of implementation attacks. Implementation attacks represent attacks that do not target the weakness of the algorithm, but rather its implementation [75]. They can be categorized based on their level of invasiveness on the device, ranging from invasive, semi-invasive, to non-invasive, and their impact on the device's operation, classified as either active or passive attacks. Side-channel analysis (SCA) represents passive, non-invasive attacks. SCA exploits unintended side-channel leakage that occurs during the execution of cryptographic algorithms on targeted devices. This side-channel information includes, e.g., power consumption [62], electromagnetic emissions [101], and execution time [61]. By analyzing these side channels, attackers can extract sensitive information such as cryptographic keys. Depending on the attacker's ability and resources, SCA is commonly divided into direct (non-profiling) and profiling (two-stage) SCA. This work focuses on physical SCA (which we simply denote by SCA in the rest of the paper) and leaves fully software attacks (e.g., microarchitectural attacks or cache attacks) as outside of scope.

In 2016, Maghrebi et al. [74] conducted the first study on deep learning for SCA, marking a shift in the SCA community's focus from MLSCA to DLSCA. Since then, the SCA community has been investigating various deep learning techniques to improve the performance of DLSCA. Current state-of-the-art results demonstrate DLSCA to be extremely powerful, requiring only minimal feature engineering and breaking protected targets with small numbers of attack traces. Figure 1 depicts the number of papers concentrating on DLSCA and published per year. Clearly, the domain is very active, with the number of papers increasing each year since the first work in 2016. In total, there are ≈ 535 papers published on DLSCA since 2016.[1] Given the large number of publications in the field, providing an exact count is challenging. Exact steps and code used to collect these papers are provided in our public repository[2].

Motivation. From the practical (industrial) perspective, the interest (and need) for DLSCA is increasing. For instance, implementation attacks are explicitly addressed within the Common Criteria (CC) [1] for evaluating the security of cryptographic systems. More recently, the German Federal Office for Information Security (BSI) published an updated version of the Application Notes and Interpretation of the Scheme (AIS) 46, providing guidelines for evaluating machine learning-based side-channel attack resistance [36]. At the same time, it is difficult to expect the industry to evaluate hundreds of publications and recognize the most important directions to follow.

From an academic perspective, it is difficult to identify crucial developments in the field since almost all papers claim to improve the state-of-the-art, but most lack proper statistical evaluation and significance tests. At the same time, the constant developments in the deep learning domain also bring potential new approaches for DLSCA, making the DLSCA

[1] These numbers were obtained by first collecting all papers that cite at least one of [6, 18, 74, 100]. Then, we remove duplicates and manually filter whether the paper uses DLSCA. When available, we use the officially published work, but unpublished preprints are also included.

[2] https://zenodo.org/records/20340778

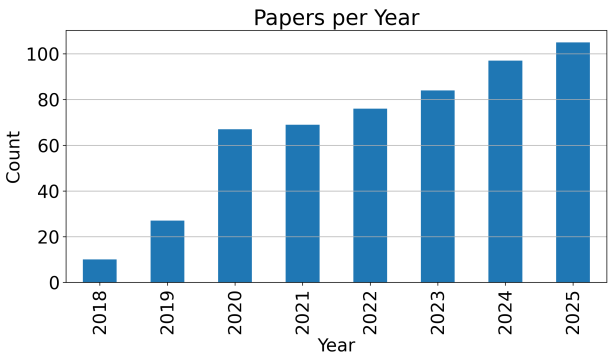


Figure 1: Overview of papers dealing with deep learning-based side-channel analysis per year from 2018. We consider [74] as the first paper focusing on DLSCA. One paper from 2016 and three papers from 2017 are excluded from the figure for readability reasons.

community an “implementer" of deep learning paradigms.

Contributions. This Systematization of Knowledge (SoK) has two goals: (i) to synthesize and structure the rapidly growing literature on DLSCA, and (ii) to provide a small set of reusable artifacts that make future work easier to compare and reproduce. While building upon the existing SoK on this topic from Picek et al. [100], this work introduces entirely new taxonomies and artifacts to systemize the DLSCA field. Concretely, our contributions are:

- *Synthesis/systematization.* We curate and organize prior DLSCA work into a unified view of (a) attacker goals across the SCA pipeline (e.g., preprocessing, feature learning, leakage assessment, and key recovery), (b) supervision regimes used in practice, (c) common model-selection and evaluation methodologies, and (d) emerging directions including explainability/interpretability and generative methods in SCA.

- *Proposed artifacts (introduced in this paper).* We introduce three community-facing artifacts intended to support comparability and reproducibility: (1) an effect-based dataset difficulty description based on a (P, I, S) score capturing leakage strength, temporal instability, and scope; (2) a two-axis threat-model taxonomy that classifies DLSCA methods by attacker objective and supervision state; and (3) a reproducibility assessment rubric and checklist derived from a survey of published papers, highlighting common reporting gaps and offering minimum reporting guidance.

2 Preliminaries

2.1 Side-channel Analysis

Direct Attacks. Direct SCA analyzes traces from a target device directly, relying on statistical methods (e.g., Pearson correlation, difference-of-means, or mutual information) to reveal correlations between observed side-channel information and secret data. Commonly used techniques are Simple Power Analysis (SPA) [61] and Differential Power Analysis (DPA) [62]. For differential attacks, an attacker generally targets some intermediate operation that depends on a known input variable and (part of) the key. To extract the key, the attacker then enumerates the hypothetical intermediate values generated for each possible key candidate. The key candidate that best matches the measured leakage is then assumed to be the correct key.

Profiling Attacks. Profiling SCA constitutes a powerful SCA category used for worst-case evaluations of implementations. It consists of two stages: a profiling stage where an attacker uses a (fully) controlled copy of the device to model the secret-dependent leakage; and an attack stage where this model is leveraged to extract the secret key from the target [23]. During the profiling stage, the attacker uses the controlled device with known key(s) to measure a set of N traces $\mathbf{X} = \{\mathbf{x}_i | i \in \mathbf{Z}_N\}$ where each $\mathbf{x}_i$ is a vector consisting of M features during a single execution of the algorithm. Since for this set of traces the keys are known, the attacker can determine the used values for any chosen intermediate operation. A standard example of a profiling attack is the template attack (TA) [23]. Template attacks create a statistical model or ‘template’ that describes the side-channel leakage and noise of the open device under control. During the attack phase, the adversary uses the created template to analyze the measurements collected from the target device, which should be similar to the clone device, and determines the most likely key used on the target device. Besides the template attack, common examples of profiling attacks are attacks based on machine (deep) learning. The corresponding approaches are commonly denoted by machine learning-based SCA (MLSCA) and deep learning-based SCA (DLSCA).

Assessing the attack performance. Common options are side-channel metrics like key rank, guessing entropy, and success rate. Key rank is simply the position of the correct key in the key guessing vector (the output of a neural network) that is ordered from the most likely to the least likely key guess. The smaller the key rank, the better the attack is. Guessing entropy is the average key rank, where we average performance over different sets of attack traces to reduce the variability stemming from selecting specific traces.

Countermeasures. Since side-channel attacks are very powerful, it is common to protect targets with countermeasures. A common division of countermeasures is into hiding and masking ones. Hiding countermeasures can occur in the

amplitude domain (e.g., adding Gaussian noise) and in the time domain (e.g., introducing local or global desynchronization). On the other hand, masking works by combining sensitive variables with some masks (random values), where the attacker cannot obtain the sensitive variable without obtaining the mask and then combining those to remove the mask.

2.2 Deep Learning-based SCA

Due to the large popularity of DLSCA, there are many approaches followed in the literature today that can be mapped to the machine learning terminology. Discriminative learning refers to models or objectives that learn decision functions from inputs (e.g., predicting a target from a trace), typically corresponding to learning conditional relationships such as $p(y \mid x)$. In contrast, generative models aim to model how the observed traces are produced by learning the input distribution, i.e., $p(x)$, or a conditional distribution $p(x \mid y)$ when generation is guided by auxiliary variables; such models can be used to synthesize traces, denoise, or learn structured latent representations [43]. Notably, template attacks [23] aim to model these joint distributions for leakage related to one (or more in the case of masked implementations) intermediate values for each possible label of the targeted operation. The modeled distributions are then matched to attack traces to predict how likely it is that any intermediate value generated the leakage.

We distinguish learning settings by the availability of targets (labels) for the task at hand. In supervised learning, training uses ground-truth targets (e.g., leakage model classes) to optimize predictive performance. A common approach is to use classification, but there are some works that also regress labels [42]. In unsupervised learning, no targets are provided, and training relies on alternative objectives, such as reconstruction or density modeling. Common unsupervised tasks include representation learning (learning embeddings that capture trace structure), clustering, and preprocessing tasks such as denoising, alignment, or segmentation.

Deep learning-based SCA then aims to learn parameters θ for a model $f_\theta : X \mapsto Y$ that extracts (a probability distribution over) the used intermediate value from the traces. These parameters are optimized for the training set, i.e., the traces $\mathbf{X}$ taken from the profiling device. To attack the target, the attacker measures traces from the target device and uses the trained model to extract the key by combining predictions across several traces. We refer readers to [57] for more practical information on DLSCA.

3 Targets and Datasets

Dataset acquisition represents the first step in mounting DLSCA. While many teams lack the expertise or equipment to measure new side-channel traces, this limitation becomes less critical when a good selection of publicly available datasets is available. Naturally, the effort to use such datasets should be as low as possible. That effort can be interpreted through several dimensions: 1) open-source permanent link, 2) unified format for storing datasets, and 3) detailed instructions on what is measured and how to use the dataset.

In Table 5, we provide information about the publicly available datasets. We can observe that a common argument that DLSCA lacks publicly available datasets does not hold. Indeed, the table contains more than 40 datasets. However, the fact that many datasets are available does not mean there are no gaps. We can divide the datasets based on the type of cryptographic algorithm implemented into 1) AES implementations, 2) public-key algorithms, 3) lightweight symmetric key algorithms, and 4) others.

3.1 Cryptographic Algorithm Categories

AES. Based on Table 5, there are more than 20 datasets with measurements from AES-128. While most of the targets are in software, there are also several hardware implementations and datasets containing diverse countermeasures.

ECC and PQC. The first datasets mostly provided elliptic curve cryptography (ECC) implementations, while more recent works are oriented toward post-quantum cryptography algorithms (PQC). With the standardization process initiated by the National Institute of Standards and Technology (NIST) in 2016, today, we have two Key Encapsulation Mechanism (KEM) algorithms and three Digital Signature Algorithms (DSA) selected. The process is still ongoing with additional signature schemes under evaluation. For a recent overview of side-channel attacks and countermeasures for PQC, we refer readers to [32]. Based on the taxonomy presented in that work, we see that DLSCA (MLSCA) is used to attack KEM schemes (both lattice- and code-based) with static key setup, code-based KEM schemes with ephemeral key setup, and lattice-based DSA schemes. While most of those works do not provide datasets,[3] there are several sources containing side-channel traces for Dilithium, Kyber, and BLISS.

Lightweight Algorithms. NIST launched the Lightweight Cryptography (LWC) initiative in 2018 to standardize cryptographic algorithms for devices with limited resources, with Ascon being selected in 2023. Besides Ascon, we can see that PRESENT is also a common target.

Other Algorithms. There are efforts to provide datasets for less-used ciphers. Examples include DES and Keccak.

3.2 Dataset Difficulty

One challenge in DLSCA research is that it is often unclear which public targets should be used to evaluate which claims. Here, we provide guidelines on which datasets to use for specific purposes. To enable meaningful comparison across

[3] They usually specify the source code and platform, with the common choice being ChipWhisperer versions.

deep learning side-channel benchmarks, we adopt an effect-based dataset difficulty model, i.e., we characterize difficulty based on what the attacker observes in the collected data rather than the underlying causes. We quantify dataset difficulty using three integer-valued scores: (P, I, S), where P captures leakage strength (magnitude) difficulty, I captures temporal instability difficulty, and S captures scope difficulty. Each score reflects the observable effects that directly constrain a DL-based attacker, regardless of whether they originate from deliberate countermeasures or measurement conditions.

A natural alternative would be a cause-based taxonomy that distinguishes difficulty induced by deliberate countermeasures from difficulty induced by the acquisition setup or environment. While conceptually appealing, such a formulation is problematic in practice for three reasons. First, the cause of an observable effect is often undocumented or ambiguous. Public datasets rarely specify whether low SNR results from deliberate noise injection, probe placement, or environmental interference. Similarly, desynchronization may stem from random delays, clock drift, firmware scheduling, or capture instability. A cause-based score would therefore require subjective attribution, reducing reproducibility. Second, deep learning models are sensitive to effects, not causes. From the attacker's perspective, low SNR caused by noise injection and low SNR caused by a poor measurement setup are indistinguishable: in both cases, discriminative features are weak relative to noise. Likewise, temporal misalignment affects learning regardless of whether it originates from shuffling countermeasures or from clock instability. Third, attack design decisions depend on observable difficulty. Choices such as using attention, temporal pooling, heavy augmentation, or domain-robust training are driven by what the traces look like, not by why they look that way. Thus, an effect-based taxonomy aligns more naturally with how DLSCA methods are evaluated and compared. We present (P, I, S) as a lightweight descriptive rubric, not a definitive standard; the goal is to make assumptions explicit and comparisons more structured. We expect absolute thresholds (e.g., SNR cutoffs) may be refined by the community; the current thresholds for, e.g., SNR values and numbers of profiling traces, are heuristic bins derived from our experience and existing literature, designed to separate practically different regimes. Our contribution is the separation into magnitude/instability/scope and the encouragement to report them explicitly. In the following definitions of (P, I, S), each component initially defaults to 0. For each complicating factor, a discrete integer increment is added to the total penalty score.

3.2.1 Leakage Strength (Magnitude) Difficulty (P)

Leakage strength difficulty captures how hard it is for a model to extract informative features due to weak or noisy leakage amplitudes. It combines the presence of mechanisms that reduce exploitable leakage with the observable severity of that reduction. We define: $P = P_{mech} + P_{snr}$.

Mechanism component (P_{mech}). We add +1 for each observable mechanism that reduces effective leakage strength: a) masking (for each order), b) amplitude hiding/blinding/noise injection, and c) other amplitude-reducing defenses.

Severity component (P_{snr}). To capture magnitude instability, we add an SNR tier based on measured peak SNR: a) +0 if $SNR \geq 1$, b) +1 if $0.1 \leq SNR < 1$, and c) +2 if $SNR < 0.1$. For masked implementations, we use the secret share with the lowest SNR. Crucially, SNR is treated as an observable effect, regardless of whether it originates from countermeasures or from the acquisition environment. These current thresholds are chosen to separate high-, medium-, and low-noise datasets. Specifically, $SNR \geq 1$ represents clean, unprotected implementations where standard DL models achieve trivial convergence. At the same time, the threshold $SNR < 0.1$ defines the level of noise at which individual leakage features become visually indistinguishable from background variance, requiring specialized architectures or extensive training data for successful key recovery.

3.2.2 Temporal Instability Difficulty (I)

Temporal instability difficulty captures how unpredictable the temporal location of leakage is across traces, which directly impacts a model's ability to align traces. We model temporal instability through severity tiers, reflecting the magnitude and nature of misalignment: a) +1 for small jitter (bounded misalignment, typically tens of samples), b) +1 additional for large jitter or clock drift (hundreds to thousands of samples), c) +1 additional for dynamic POI movement, where leakage regions relocate non-stationarily across traces, and d) +1 additional for inter-session or inter-device shifts affecting timing distributions. This formulation explicitly distinguishes datasets with mild, easily compensated misalignment from datasets with fundamentally non-stationary leakage. We again emphasize that these effects are considered only in terms of what is observed in the collected data with the given device and measurement, and that the causes of the misalignment are not treated differently.

3.2.3 Scope Difficulty (S)

Scope difficulty reflects the size and complexity of the learning problem, independent of leakage strength or alignment. We define: $S = S_{mech} + S_{dim} + S_{traces}$.

Mechanism component (S_{mech}): a) +1 if segmentation is required to locate the cryptographic operation and b) +1 if target width exceeds 8 bits (16-bit, 32-bit, or multi-byte targets).

Dimensionality component (S_{dim}) for trace length d: a) +0 if $d \leq 1\,000$, b) +1 if $1\,000 < d \leq 5\,000$, c) +2 if $5\,000 < d \leq 20\,000$, and d) +3 if $d > 20\,000$.

Table 1: Breakdown of the dataset difficulty criteria (NOPOI cases use window 20 to resample from raw traces [96]).

Dataset	P_{mech}	P_{snr}	I	S_{mech}	S_{dim}	S_{traces}	Total
ASCADf	1	0	0	0	0	1	2
ASCADf desync 50	1	0	1	0	0	1	3
ASCADf desync 100	1	0	2	0	0	1	4
ASCADf NOPOI	1	0	0	0	2	1	4
ASCADr	1	0	0	0	1	0	2
ASCADr NOPOI	1	0	0	0	3	0	4
CHES_CTF 2018 NOPOI	1	0	0	1	2	1	5
ESHARD	1	1	0	1	1	1	5
AES_HD	0	2	0	0	1	1	4
ASCADv2	2	2	0	2	2	0	8

Trace count component (S_{traces}) for profiling traces n: a) +0 if $n > 100\,000$, b) +1 if $20\,000 < n \leq 100\,000$, and c) +2 if $n \leq 20\,000$. This scoring system reflects the interaction between dimensionality, sample scarcity, and overfitting risk through aggregation. The thresholds are chosen to capture qualitative transitions. For S_{dim}, the boundaries distinguish between baseline traces manageable by standard architectures, moderately long traces that introduce practical dimensionality challenges, and long traces where feature scale dominates computational constraints and exacerbates overfitting risks. For S_{traces}, the thresholds separate the data-rich regime, where overfitting is rarely a limiting factor, from the data-scarce regime, where it typically is, requiring careful consideration of model designs and the training process.

3.2.4 Dataset Difficulty Examples

Using our proposed dataset difficulty model (P,I,S), we score common (versions of) datasets in Table 1. Note that individual components should be analyzed independently. A higher total score indicates more compounding complications, but does not imply that specific datasets are strictly more difficult than others. Additionally, as has already been observed in the ASCAD cases with simulated desynchronization, an existing dataset can be made more difficult by, e.g., simulating additional hiding countermeasures or by limiting the size of the profiling set. Finally, a dataset can be made simpler by assuming knowledge of a (subset of) shares during profiling, essentially reducing the masking order [80].

3.2.5 Interpretation

The resulting (P,I,S) triple characterizes dataset difficulty in terms of what challenges an attacker must overcome, rather than how those challenges were produced. Datasets may therefore be difficult for fundamentally different reasons: one may be dominated by magnitude difficulty (low SNR), another by temporal instability (dynamic POIs), and another by scope (high dimensionality and limited traces). This distinction explains why similar architectures can succeed on one benchmark but fail on another, and enables principled comparison across datasets and DLSCA methods.

3.3 Leakage Simulators and Synthetic Datasets

In addition to measured traces from physical devices, a number of works rely on leakage simulators that generate synthetic traces from a leakage model. Simulators are implemented in a white-box scenario with full knowledge of the design implementation details [112]. As such, they are valuable for isolating algorithmic contributions and stress-testing robustness under precisely specified conditions. However, simulated leakage can differ substantially from real measurements: physical traces often exhibit non-Gaussian noise, analog front-end distortions, and device- and session-dependent shifts that are difficult to model faithfully [16]. Consequently, simulator-based evaluation is most informative when used alongside measured datasets, and when the simulator parameters are explicitly mapped to the observable difficulty factors (P,I,S). More precisely, simulators allow one to set P and I deliberately so as to align with real-world targets, while S can also be easily varied by adjusting trace length/sample rate.

TAKEAWAYS

- Despite rapid progress in DLSCA, papers often leave dataset difficulty and supervision assumptions implicit, making results hard to compare across threat models.
- Public SCA datasets are heavily skewed toward AES. In contrast, publicly available traces for lightweight, PQC, and stream ciphers remain scarce relative to their research interest, which limits reproducibility and cross-paper comparability.
- Today's public benchmarks over-represent "easy" regimes; the community needs mid-to-high difficulty datasets (e.g., stronger P/I) that remain benchmarkable without extreme compute, to reduce overfitting to single-trace-solvable targets.

4 Threat Models

In [100], the authors discussed three threat models: classical (where one device is used for both profiling and attack), portability (with different devices for profiling and attack), and non-profiling supervised (where there is only one device under attack). While it may be argued that these threat models represent the basics of what is possible in DLSCA, we can observe in the last few years a number of papers that extend those threat models, especially the non-profiling ones. As such, the state-of-the-art DLSCA offers significantly more options for mounting attacks. What is more, from an attacker's perspective, the portability scenario seems less threat model-oriented and more dataset-oriented. Indeed, the fact that there are different profiling and attack devices does not change the attack pipeline, but increases the dataset difficulty.

4.1 DLSCA Threat Model Taxonomy

We organize deep learning-based SCA threat models using a two-axis taxonomy defined by **(i) the attacker's learning goal** and **(ii) the supervision state**. This design deliberately balances expressiveness and usability: the two axes capture the dominant dimensions along which DLSCA methods differ, while avoiding overly fine-grained categorizations that would result in sparse or ambiguous classes.

4.2 Axis 1: Learning Goal

The first axis captures *what the attacker aims to achieve with learning*. We distinguish three goals that correspond to distinct stages of a DLSCA pipeline.

Preprocessing. The objective is to transform raw traces into a representation that is easier to learn from, without directly modeling key-dependent leakage. Typical preprocessing goals include denoising, temporal alignment, segmentation, normalization, and data augmentation. While preprocessing does not directly recover the key, it can influence downstream attack performance and robustness.

Feature engineering. The objective is to learn representations that expose leakage-relevant structure while being invariant to factors such as noise, desynchronization, or device-specific artifacts. Feature engineering methods often aim to discover points of interest, learn domain-invariant embeddings, or construct leakage-sensitive latent spaces that can later be reused for key recovery.

Key recovery. The objective is to directly rank or recover secret key material (or key-dependent intermediates) using a learned model. Key recovery is the final attack goal and is typically evaluated using side-channel metrics. Methods in this category may incorporate preprocessing and feature learning implicitly, but they are distinguished by producing key hypotheses as output.

Leakage assessment. The objective is to determine whether a device/implementation exhibits exploitable leakage, localize when leakage occurs, and quantify leakage strength. This goal is conceptually analogous to TVLA-style evaluation, but replaces fixed statistical tests with learned detectors or representation-based criteria. Leakage assessment is distinct from key recovery because its output is not a key hypothesis, but an assessment signal (detection/quantification/localization) that guides evaluation and countermeasure validation.

4.3 Axis 2: Supervision State

The second axis captures *what type of supervision the attacker has access to during training*. We combine label availability and label provenance into a single supervision axis, resulting in three mutually exclusive supervision states.

Full labels. All training traces are labeled with respect to the learning objective. Labels may be direct leakage model labels (e.g., S-box outputs, Hamming weight) or indirect but deterministic labels derived from the known values (e.g., as done in [123]). This corresponds to fully supervised learning and represents the strongest attacker assumption.

Pseudo labels. No ground-truth labels are available for the sensitive variable. Instead, labels are constructed from data structures or consistency constraints (e.g., clustering, collisions, self-training, correlation). Pseudo labels introduce uncertainty and potential bias, but enable learning in settings where classical profiling labels are unavailable.

No labels. No labels are used or constructed at any stage. Learning relies entirely on unsupervised objectives, such as reconstruction or distribution alignment.

4.3.1 Motivation for the Two-axis Design

This two-axis taxonomy is motivated by three considerations. First, the *learning goal* determines how explanations, evaluations, and comparisons should be interpreted: success in preprocessing or feature engineering does not directly imply successful key recovery. Second, the *supervision state* captures the attacker's assumptions about available information, which fundamentally constrains feasible techniques and achievable performance. Third, combining these axes yields a compact yet expressive structure that accommodates diverse DLSCA methods without conflating orthogonal factors such as architecture choice or dataset difficulty. Importantly, other dimensions (e.g., dataset difficulty) are treated as orthogonal attributes rather than axes, ensuring that each work can be unambiguously placed within the taxonomy.

White-box vs. black-box (implementation knowledge). In side-channel analysis, the terms *white-box* and *black-box* typically refer to the attacker's knowledge of the target implementation (e.g., leakage model, masking/shuffling scheme, and the relation between observed traces and sensitive intermediates), rather than to query access to a trained model. The main distinction here is that for worst-case adversaries (i.e., white-box) that attack protected implementations, countermeasures can be turned off to simplify attacks. Examples here include disabling hiding countermeasures to enable effective trace preprocessing [124] and profiling secret shares using known masking randomness [80]. More uninformed attackers are often unable to select specific feature ranges, leading to a need to process raw traces, which increases the (computational) difficulty of finding an appropriate model [72, 96].

This assumption is orthogonal to our two-axis taxonomy (goal $\times$ supervision), but it strongly constrains which supervision states are feasible. A white-box attacker can often compute leakage model targets (e.g., intermediate values or masked shares) and thus supports full labels profiling. In contrast, a black-box attacker may not be able to derive correct leakage labels due to unknown masks, unknown execution order, or unknown points of leakage, which motivates pseudo

labels strategies (e.g., collision- or clustering-based labeling) and unsupervised preprocessing/representation learning. We, therefore, treat white-/black-box knowledge as an attacker capability attribute that influences the supervision state, rather than as an additional axis.

4.3.2 Representative Approaches

Table 2 summarizes representative DLSCA techniques organized according to the proposed two-axis taxonomy. The table is illustrative rather than exhaustive; many works span multiple cells depending on how they are instantiated.

Next, to clarify the interpretation of our taxonomy and to avoid ambiguity arising from inconsistent terminology in prior work, we introduce a small set of anchor cases. These anchor cases are not meant to survey the literature exhaustively, but rather to delineate category boundaries and fix the intended meaning of each axis. By grounding the taxonomy in a few canonical scenarios, we provide reference points against which both existing and future DLSCA approaches can be consistently classified. We use these anchor cases to fix the interpretation of each category; concrete instantiations and representative works are summarized in Table 2.

Anchor case: profiling attacks. Profiling DLSCA is the canonical example of full labels supervised learning. A profiling attacker has access to an auxiliary device and can compute ground-truth training targets (e.g., sensitive intermediates or leakage model classes) for every trace. The learning task is a standard supervised classification or regression problem, and performance reflects how well the model exploits labeled leakage given the assumed level of dataset difficulty. This case defines the strongest supervision regime in our taxonomy.

Anchor case: non-profiling attacks. Non-profiling DLSCA is often described as "unsupervised", but in practice, many such attacks optimize supervised objectives using constructed targets. Typical examples include partition-based, collision-based, or differential formulations in which labels depend on key hypotheses, trace collisions, or learned groupings rather than on ground truth leakage values. Although these methods do not rely on a profiling device, they still train models using explicit target signals and, therefore, fall naturally under pseudo labels supervision rather than true no labels learning.

Anchor case: leakage assessment vs. key recovery. Leakage assessment addresses a fundamentally different goal from key recovery. Rather than inferring secret values, leakage assessment aims to detect, localize, or quantify the presence of information leakage in a manner conceptually analogous to TVLA [4]. While similar tools (e.g., classifiers, attribution methods, or segmentation techniques) may be used, the output of leakage assessment is a diagnostic signal rather than a key hypothesis. Treating leakage assessment as a distinct goal clarifies that success criteria, supervision requirements, and evaluation metrics differ substantially from those used in attack-oriented settings.

To demonstrate that the proposed taxonomy captures existing practice without resorting to an exhaustive literature survey, we summarize representative examples in Table 6, Appendix C. The table is not intended to be complete, but to illustrate how established DLSCA techniques instantiate different combinations of learning goal and supervision state. Presented examples ground the taxonomy in concrete work while keeping the main text focused on conceptual distinctions rather than enumerating individual attacks.

TAKEAWAYS

- Profiling DLSCA is standard supervised learning with the leakage model labels; many "non-profiling" DL attacks remain supervised, but with constructed targets (pseudo labels) from collisions, partitions, or hypotheses.
- Strict semi-supervised learning (joint labeled+unlabeled training) is still rare in DLSCA; "limited supervision" is usually implemented via pseudo labels or self-supervised pretraining followed by supervised fine-tuning.
- Comparisons across supervision states conflate attacker capability with algorithm quality; meaningful comparisons should be made within the same supervision row and under matched dataset difficulty unless the goal is explicitly to trade off attacker capability.

5 Model Selection, Tuning, and Evaluation

Since the introduction of deep learning (DL) to SCA by Maghrebi et al. [74], one of the main considerations for applying DL in practice is how to design a network architecture to break a target successfully. While initial works were using larger models directly from vision/audio processing [6, 60], the work of Zaid et al. [135] highlighted that much smaller models constructed for SCA can achieve comparable (or even better) attack performance at significantly lower computational cost. Since then, two main lines of work have emerged: 1) automated architecture/hyperparameter searches to find new architectures for any new target and 2) general architectures that should translate directly to any new target. Beyond architectural choices, recent work increasingly shows that algorithmic decisions and hyperparameter regimes, including training objectives, optimization strategies, and regularization, often have an impact comparable to or greater than the choice of network topology itself. Importantly, architectural and hyperparameter choices cannot be meaningfully discussed in isolation: their effectiveness depends on the dataset characteristics (e.g., leakage strength, instability, trace dimensionality) and on the assumed threat model (e.g., availability of labels, profiling vs non-profiling). As a result, model selection and

Table 2: Two-axis threat model taxonomy for DLSCA. Techniques are organized by learning goal (columns) and supervision state (rows). To our knowledge, there are no works demonstrating successful key recovery on symmetric key algorithms in the no labels setup.

Supervision state	Preprocessing	Feature engineering	Key recovery	Leakage assessment
Full labels	Supervised alignment or window selection using labeled traces	Multi-task learning for POI discovery; supervised embedding learning	Classical profiling DLSCA with leakage-model targets	Supervised leakage/no-leakage classifier; supervised POI localization
Pseudo labels	Clustering-based segmentation; collision-driven window discovery	Contrastive or collision-based representation learning	Blind or collision-based DLSCA using constructed labels	Clustering-based leakage detection
No labels	Unsupervised denoising, autoencoder-based alignment	Unsupervised embeddings; distribution alignment	(Rare) Fully unsupervised or heuristic-assisted key recovery	Unsupervised segmentation for leakage localization

tuning act as an implicit interface between dataset difficulty (Section 3) and the attacker's capabilities formalized in our threat model taxonomy (Section 4).

5.1 Hyperparameter and Algorithm Search

To reduce the reliance on single models and manual tuning, the research community has been exploring automated tuning of network architectures/hyperparameters. The main results here are that several methods, including random search, can automatically find architectures that outperform manually designed networks across several datasets [2, 96, 105, 122]. The computational cost of these approaches is also limited; while hundreds of networks must be trained, the individual networks are generally small and quick to train. Additionally, with more computational resources, the training of individual models can also be parallelized.

A key observation across these works is that *the definition of the search space is often more important than the search strategy itself*. For instance, while in [96], for the standard ASCADv1 interval, 25-75% of random models succeed, for more difficult targets, this decreases to 0.16%. A smaller hyperparameter range can improve the rate of successful models, but this requires knowing how and where to narrow the search space. The main alternative for architectures/hyperparameter tuning is guided search. This was first presented in the SCA context in [74] where the authors mentioned they used genetic algorithms. Later, common and well-performing options include reinforcement learning [105] and Bayesian optimization [122]. Finally, additional approaches have mostly focused on using evolutionary techniques [2, 83, 111]. These methods differ primarily in how explicitly prior structure is imposed on the search space and how efficiently poor configurations are discarded. From the perspective of dataset difficulty, this behavior reflects the fact that narrow search spaces implicitly assume low instability and strong leakage (low I and P), whereas datasets with high instability or weak leakage require architectures and optimization regimes that are robust to misalignment and noise.

5.2 General Architectures and Training Regimes

An appealing alternative to repeated tuning is the use of *general architectures and training regimes* that can be applied to new targets with minimal modification. While several works have proposed generic CNN-based models [60] or methodological guidelines for constructing SCA-specific networks [119, 135], transferring these designs to new datasets often requires non-trivial adaptation.

Recent work has explored attention-based and transformer-inspired architectures [45, 46], motivated by their ability to aggregate information across long temporal contexts and to tolerate misalignment. Other approaches aim to train a single "universal" transformer model across multiple targets [17]. While promising, these models typically rely on carefully chosen optimization settings and regularization strategies to remain stable across heterogeneous datasets.

Across this line of work, a recurring pattern is that general architectures are often accompanied by *fixed or minimally adapted hyperparameter configurations*. Although this improves comparability, it also obscures the extent to which success depends on dataset-specific tuning. Ablation studies varying individual hyperparameters provide useful insights into sensitivity, but they rarely capture interactions between architectural depth, receptive field, pooling strategy, and optimization parameters.

From a threat model perspective, the use of general architectures with fixed training regimes implicitly assumes a stable supervision setup (typically full labels profiling) and moderate dataset difficulty. When these assumptions are violated, e.g., in pseudo labels or no labels settings, or under strong instability, general architectures often require non-trivial adaptation to remain effective. This observation helps explain why architectures that generalize well across benchmark datasets do not necessarily generalize across threat models.

For smaller models with short training times, limited manual tuning is often sufficient for competitive performance. In contrast, for large attention-based or transformer architectures, each training run can be computationally expensive, making naive trial-and-error tuning impractical. There, architectural generality trades off against accessibility and reproducibility.

5.3 Metrics and Evaluation Implications

As demonstrated by Picek et al. [98], standard machine learning metrics such as accuracy or precision can be misleading in DLSCA, as they do not directly reflect attack success. Consequently, most works rely on key-centric metrics such as key rank, success rate, and guessing entropy [100], with guessing entropy emerging as the most commonly reported due to its stability and informativeness across different numbers of attack traces.

The interpretation of key-centric metrics such as guessing entropy is inseparable from both dataset difficulty and the assumed threat model: identical guessing entropy curves may correspond to very different attacker capabilities depending on label availability, dataset instability, and trace dimensionality.

However, algorithmic and hyperparameter choices strongly influence how these metrics should be interpreted. For example, aggressive regularization or data augmentation may yield stable classification accuracy while slowing the reduction of guessing entropy. Similarly, two models with comparable final guessing entropy may differ significantly in the number of traces required to reach key rank zero. Metrics such as the average number of traces to reach zero guessing entropy (T_{GE0}), therefore, provide valuable complementary information.

In particular, metrics that emphasize sample efficiency or training stability are more informative for high-scope or high-instability datasets and for non-profiling threat models, whereas peak performance metrics dominate comparisons on low-difficulty benchmarks.

From an efficiency perspective, the number of trainable parameters is frequently reported as a proxy for model complexity. Yet this metric alone does not capture training stability, sensitivity to hyperparameters, or robustness across datasets. Given that many contemporary techniques break standard benchmarks with a single trace, additional criteria, such as performance across all key bytes, sensitivity to training set size, and robustness under perturbations, are increasingly necessary to meaningfully differentiate approaches. Analyzing the metrics proposed in the last few years in DLSCA, we can divide them into 1) key guessing vector-based, 2) mutual information, and 3) training effort-based. We provide a detailed list of metrics in Appendix D.

TAKEAWAYS

- In DLSCA, objectives and hyperparameters can encode stronger assumptions than the stated threat model.
- Hyperparameter search should report success rate and variability (mean±std) of GE/key rank metrics.
- As benchmarks become "single-trace solvable", evaluation must report robustness and efficiency, not only best GE, and clarify which metrics match which dataset/threat-model assumptions.

6 XAI in DLSCA

The practical goals of XAI in DLSCA are twofold: evaluators want to understand what leakage is being exploited by a network that extracts the key, and researchers aim to understand why a model does or does not recover the key. As such, we split our discussion into two axes.

First, in security evaluations, the successful extraction of a (sub)key indicates the presence of practically exploitable leakage. To effectively mitigate this issue, ideally, an evaluator could highlight specific intermediate values and trace points used in this key extraction [40]. This can then be useful for highlighting specific security issues in an implementation under evaluation and resolving them in a targeted manner. Second, to predict the effects that adding specific countermeasures will have on an implementation's resistance to DL-based attacks, we need a better fundamental understanding of *why* neural networks do or do not learn. Here, we should aim to answer broader questions, e.g., about learning dynamics under varying conditions and robustness to transformations. The overall goal here is to move from best-effort evaluations to more streamlined and robust evaluation procedures.

6.1 How do Neural Networks Extract Keys

The three conditions outlined by Gao and Oswald [40] for explainable leakage detection are: 1) a list of variables that contribute to the exploitable leakage, 2) the points in the trace that leak each of those values, and 3) a concrete attack vector exploiting that leakage. Clearly, a model that extracts the (sub)key already fulfills the third condition, but as standard neural networks are difficult to interpret, the first two conditions require additional methods. We note that there is a separate line of work on training more interpretable models, which we discuss in Section 6.3. For the second condition, feature-based attribution methods provide a natural direction as these highlight the parts of the input that contribute to a prediction. Overall, there have been a large number of works that use a variety of techniques to accomplish this goal, see, e.g., [39, 48, 79, 131]. These methods tend to perform reasonably well for synchronized traces, as averaging attributions over a large number of traces removes noise. Highlighting the specific intermediate values used by the network is more difficult to accomplish. By overlaying feature attribution maps with, e.g., SNR plots, an evaluator can determine whether the features the network relies on correspond to leakage points [47, 79]. However, for masked implementations, this is only possible with access to internal randomness (for visualizing intermediate values).

More recently, mechanistic interpretability has also been shown to be effective at discovering which secret shares do (not) get used by models [95], and even extracting the secret shares the model uses directly [56]. As the field of mechanistic interpretability rapidly evolves, methods that simplify

and automate analyses are an extremely promising direction for future work. We distinguish XAI used for attributing leakage (what/where) from XAI used for mechanistic validation (why/when models generalize).

6.2 Why/When do Neural Networks Work

Another key question in DLSCA is what the necessary conditions are for models to learn. In our view, the eventual goal of the research community should be to predict the effects of specific countermeasures on whether networks can learn to extract information under specific settings (see Section 3 for examples). This would then allow for well-informed tradeoffs in the design process and for less ad-hoc evaluations using DLSCA. We split the discussion here into the standard division of masking and hiding countermeasures, as these have distinct effects on the learning problem.

Masking: Defeating masking, even in a single trace, without access to secret shares during profiling, is one of the key benefits of using DLSCA [100]. Significant attention has been given to how this occurs. Overall, while the networks can learn to recombine shares, there is a difficulty in learning the higher-order leakages, which manifests in the "initial plateau" [77], an initial phase during training where the network does not learn anything for a number of steps related to the masking order. There then seem to be tradeoffs in how we can bias the training process to reduce the length of this plateau [20, 58, 77, 110]. However, these works often make a priori assumptions about which shares can be learned. Another concern is that while it is clear that higher-order leakage is more difficult to learn, we have very limited ideas under what conditions it is then still learnable. Simulations are often still relatively easy to learn even at higher orders (see [14]), but in practice, the only successful, non-worst-case, attack against higher-order masking is the attack against the second-order CHES_CTF 2020 target in [45], which can only retrieve a single bit out of the 32-bit state. Furthermore, the practical targets that we can successfully attack are limited to Boolean schemes, and the practical implications of using multiplicative/affine schemes are difficult to evaluate with current results. Finally, while the work in [14] conducts analyses of the profiling complexity of various masking setups in simulations, the effects on real-world implementations remain largely unexplored. This is especially salient as the profiling complexity has direct implementations on the attack efficiency of DL-based non-profiled attacks [114, 123].

Hiding: For hiding countermeasures, there has been significantly more investigation into shift-invariance (and equivariance) properties of architectures [18, 45, 46, 65, 91]. Overall, from results on simulated misalignment (and initial works on jitter [18]), it seems clear that if we can simulate the effects of a hiding countermeasure accurately enough, it is relatively straightforward to learn some robustness to this transformation through data augmentation [65]. However, while these works highlight the theoretical properties of certain model components and show final attack results against transformed traces, empirical investigations into how this invariance manifests in the model are generally absent. It is therefore of interest to conduct concrete investigations into the extent to which a resulting network is internally invariant to well-defined transformations.[4]

Similar to the internal analyses of masked implementations in [56, 95], this type of analysis could yield more specific empirical insights into where and when models are actually robust to misalignment, and into the dynamics of when, during training, this invariance arises. We want to emphasize that, in our view, stating theoretical properties and showing final attack performance is not sufficient to fully understand whether a trained model is actually invariant to certain transformations. The same attack performance can hide fundamentally different internal mechanisms, and the practical implications of experimental results are highly dependent on what is actually happening inside the network. As stated in the previous section, it is also relatively tractable to analyze the mechanisms we want to consider. If we can describe the translations that a network is invariant to theoretically, simply simulating the countermeasures and seeing what happens at specific points in the network is relatively straightforward.

6.3 Explainable Architectures

An orthogonal approach is to replace standard black-box architectures with inherently more interpretable models. Several works have explored this in the context of SCA [129, 132, 133]. While this approach yields networks that are easier to interpret, it often requires additional assumptions and computational overhead, and results in reduced attack performance. Additionally, improvements to conventional architectures do not (directly) translate to these models. This limits the relevance of this line of work in our view.

TAKEAWAYS

- XAI is unusually actionable in SCA because the hypothesis space is small (few relevant intermediates/operations). The open challenge is making analyses reliable and evaluator-friendly.
- Similar attack results are insufficient to demonstrate robustness to countermeasures, as they may arise from different underlying mechanisms. Mechanistic evidence is necessary to support robustness claims.

[4] An example of such an analysis for a vision model can be found in [88].

7 Generative AI for SCA

Generative AI (GenAI) methods are not primarily used in DLSCA to directly recover secret keys. Instead, they address structural bottlenecks of side-channel learning, such as limited data availability, class imbalance, low signal-to-noise ratios, and domain shift, by modifying the effective difficulty of the dataset and supervision regime, rather than replacing discriminative attack models. Nearly all existing GenAI-based SCA works fall into preprocessing or feature engineering goals; direct GenAI-based key recovery remains largely unexplored. From a supervision perspective, generative models are most commonly deployed in full labels or pseudo labels settings. Conditional generative models rely on profiling labels, while unconditional or self-supervised models align more naturally with pseudo labels or no labels regimes, albeit often under additional structural or distributional assumptions. This makes GenAI particularly attractive in scenarios where labeled traces are scarce, but unlabeled traces are abundant.

Generative models interact with dataset difficulty in non-uniform ways. Data augmentation and synthesis primarily mitigate scope difficulty (S) by increasing the effective number of training samples, and may partially alleviate leakage-strength difficulty (P) by smoothing noise distributions. It is also possible that such approaches reduce marginal temporal instability (I) when looking at it as noise and then running noise removal, see, e.g., [124] for a non-generative AI approach.

A central risk of generative approaches is that improved performance on synthetic or augmented data does not necessarily translate to improved key recovery on genuine traces. Without careful validation, generative models may amplify spurious correlations.From the perspective of our threat model taxonomy, generative approaches in SCA predominantly affect preprocessing and feature engineering stages, while rarely altering the supervision state of the final key-recovery task.

Generative Methods for Generating Additional Data. The most natural direction for using generative models in SCA is to generate new traces. Indeed, already in [117], conditional Generative Adversarial Networks (cGANs) were used to generate additional traces to combat the challenges of imbalanced label distribution in SCA [98] and expand the size of the profiling set. This process was improved upon in [82] by updating the training methods. Diffusion models have also been used to generate additional traces in [130], performing similarly to the GANs. Currently, all of these methods operate on a single implementation, and there does not seem to be a clear path towards generative approaches that allow the generation of traces across implementations or cryptographic algorithms.

Generative Methods for Preprocessing. Another application of generative methods is preprocessing side-channel traces. Several methods introduced here aim to remove the effects of hiding countermeasures. This line of research was initiated in [124, 128], with autoencoders trained by using paired clean and protected traces to remove the effects of the hiding countermeasures. Several methods have been proposed to tackle different challenges. We classify methods based on the necessary assumptions and discuss the resulting trade-offs.

Methods Requiring Profiling Labels (Full labels supervision). Several generative and representation learning approaches leverage profiling labels to guide preprocessing or feature extraction. Access to labels enables more expressive objectives, but implicitly confines these methods to profiled or white-box threat models. Representative examples include contrastive and similarity-based learning schemes that use labeled intermediate values to construct structured embeddings, which can subsequently be attacked using conventional techniques [66, 121]. Other approaches exploit auxiliary knowledge from reference implementations. For instance, [58] uses a white-box reference device with similar masking characteristics to emulate feature selection on the target, while [127] relaxes scheme-awareness by learning to recombine shares without assuming identical masking schemes. Label-aware autoencoder variants have also been proposed to improve reconstruction quality [52], while conditional generative models, such as CVAEs, explicitly incorporate known masks or randomness to better capture leakage structure [13, 133]. While highly effective under their respective assumptions, these methods encode strong prior knowledge about the implementation and leakage model. As a result, their applicability does not extend naturally to non-profiled or black-box threat models.

Methods Requiring Clean or Reference Sets (Profiled settings). An alternative form of supervision relies on paired or reference traces rather than explicit labels. This line of work was initiated in [124, 128], where autoencoders are trained to map protected traces to corresponding clean traces obtained from the same implementation without hiding countermeasures. Once trained, such models can be used to preprocess traces from the protected target device, effectively removing the influence of hiding.

Related approaches use generative models to map between different side-channel modalities or domains, exploiting the fact that one domain may be easier to attack than another [41]. In practice, the reference set may also be synthesized through averaging or classical alignment techniques, although the resulting preprocessing quality is fundamentally limited by how well these classical methods suppress countermeasure effects [124, 128]. These approaches avoid explicit leakage labels but still require privileged access to reference measurements, which restricts them to profiled or semi-white-box scenarios.

Fully Unsupervised Methods (No labels preprocessing). Fully unsupervised generative approaches aim to modify traces without relying on labels or reference sets. Autoencoders constitute the most common example: when trained purely on reconstruction objectives, they can act as denois-

ers or dimensionality reducers without supervision [64, 124]. Domain adaptation methods based on adversarial objectives similarly attempt to align representations across devices or sessions without using labeled target-domain data [21].

More recent work has explored diffusion-based models for removing measurement noise in a fully unsupervised manner [59], as well as contrastive objectives to pretrain representations that are more robust to hiding countermeasures or device variability [19]. While these techniques reduce reliance on privileged information, their effectiveness is typically constrained to mitigating noise or representation shift rather than fully undoing strong countermeasures. Consequently, fully unsupervised generative preprocessing remains most effective as a complement to, rather than a replacement for, supervised or pseudo labeled attack pipelines.

TAKEAWAYS

- The impact of generative approaches should be evaluated in terms of how they reshape dataset difficulty and supervision assumptions, rather than as standalone attack techniques.
- Label-controlled trace synthesis typically helps only when the label–trace relation is already learnable.
- Preprocessing traces to remove the effects of hiding countermeasures is (relatively) simple when a clean reference set is available. The next challenge is attempting to remove effects without necessitating this clean set, as this will translate to black-box and non-profiled threat models.

8 Reproducibility

Because most DLSCA papers present empirical results, the correctness and reproducibility of those results are crucial. We examine a selection of 29 DLSCA papers[5], and score their reproducibility based on the described metrics in Table 3. To reproduce the reported work, we deem these characteristics relevant, with higher scores indicating easier, more straightforward reproducibility. Available source code, datasets, and a full list of hyperparameters and their values (or ranges for search) enable reproducibility. Finally, there should be comprehensive statistical reporting. The list of considered papers and their scores is available in our public repository; here, we present a summary of the reproducibility scores in Table 4. Approximately 93% (27/29) of the papers provide full access to their datasets or use already public datasets. This allows other researchers to work with the same data, making comparisons easier. Nearly all papers (96.5%) list full training hyperparameters and model topologies, either through code or description in the paper. This indicates that documenting the architecture of utilized DL models and training hyperparameters is a well-integrated standard in the community. Roughly 72.4% (21/29) of the papers provide source code. Although there is still room for improvement, most authors recognize that code is essential for verifying results. However, many conferences and journals that typically accept DLSCA works lack a policy on artifact submission and evaluation. Even those that allow artifacts do not make them mandatory. For instance, for some IACR journals and conferences, artifacts that pass the artifact review process will be archived in the IACR Artifact Archive[6]. In total, there are 203 papers with evaluated artifacts, but only 6 are on DLSCA.[7] Considering the number of DLSCA papers published in general (see Figure 1), and those with provided source code, the number of evaluated artifacts is disappointingly low. Thus, despite many papers publishing their source code, the issue of its functionality and correctness remains. Regarding statistical reporting, 75.8% (22/29) of papers only report the best or a single run (Score 0). The remaining 24.1% (7/29) report a mean or median (Score 1), but none provided full statistical information (Score 2). Given that statistical reporting is the dominant gap (R4), we perform several experiments to illustrate the variability of experiments across different axes. Specifically, we showcase three case studies in Appendix E. Case study 1 shows that retraining identical models can vary significantly, case study 2 shows the variability across hyperparameter searches, and case study 3 shows the benefits of providing a trained model. These findings lead us to summarize minimum reporting guidance as *Reproducibility recommendations* listed here.

REPRODUCIBILITY RECOMMENDATIONS

- Code: Release training + attack scripts (or equivalent pseudocode if release is impossible), including GE computation script; include commands to reproduce key tables/figures; specify the software environment.
- Data: Provide public access to new datasets (or acquisition instructions); document the exact preprocessing pipeline (alignment/filtering/windowing/normalization) used to produce the traces/features in experiments.
- Training: Report splits (train/val/test) and how they are generated; list hyperparameters and selection procedure (search/early stopping/checkpoint rule); report hardware and compute cost for main results.
- Evaluation: Clearly distinguish new results vs. baselines; report standard SCA metrics (e.g., key rank/success rate/guessing entropy) with a fully specified evaluation procedure; state which results are reproducible with released artifacts.

[5] For selecting the papers to score for reproducibility, we consider only DLSCA papers published at TCHES and "top" security and cryptography conferences where reproducibility is central to the contribution.

[6] https://artifacts.iacr.org/

[7] Accessed on February 3, 2026.

Table 3: Explanation of reproducibility scores.

Metric	Category	Description	Score 0	Score 1	Score 2
$R1$	**Code Availability**	Is source code and inference script released?	No	Yes	/
$R2$	**Dataset Transparency**	Are raw traces and preprocessing/alignment steps accessible?	None	Acquisition setup only	Public datasets
$R3$	**Hyperparameters**	Are full training hyperparameters listed (including topology)?	None	Partial	Full
$R4$	**Statistical Reporting**	Are stochastic effects measured?	Best/One training	Mean/Median	Statistical significance

Table 4: Summary of reproducibility scores of the considered 29 papers.

Metric	Category	Count	Percentage
$R1$ **(Code)**	(1) Source code available	21 / 29	72.4%
$R2$ **(Dataset)**	(2) Public dataset	27 / 29	93.1%
$R3$ **(Params)**	(2) Full hyperparameters listed	28 / 29	96.5%
	(0) Best/One training	**22 / 29**	**75.8%**
$R4$ **(Stats)**	(1) Mean/Median	7 / 29	24.1%
	(2) Statistical significance	0 / 29	0.0%

- Statistics: Avoid "best run only"; report variability (e.g., over seeds/splits and/or evaluation sampling) and specify what error bars summarize (mean±std/Confidence Interval, number of runs).

9 Future Directions and Open Problems

Building upon our technical analysis across datasets, threat models, evaluations, and emerging AI paradigms, this section synthesizes the critical gaps in the current DLSCA landscape and outlines future directions.

Benchmarks. Current public benchmarks are heavily skewed towards software implementations of AES-128, which often over-represent "easy" clean regimes (low dataset difficulty scores). To ensure that DLSCA frameworks scale with modern cryptographic developments, we must prioritize creating and distributing datasets for lightweight and PQC schemes. Concurrently, future work should intentionally stress-test models against mid-to-high difficulty targets characterized by severe noise ($SNR < 0.1$), dynamic point-of-interest shifts, and high-order masking. We invite the community to actively adopt and refine the proposed (P, I, S) difficulty metric to establish a shared, rigorous baseline for dataset complexity.

Threat models and objectives. The supervision states and threat models of proposed attacks are often left implicit, which can lead to unfair comparisons. Many non-profiling attacks utilize supervised learning, relying on pseudo labels derived from key hypotheses or collisions. Such cases should not be compared with methods that utilize full labels. Thus, future evaluations must strictly respect these supervision boundaries and compare methods within their respective objectives and supervision states, unless the explicit goal is to map the performance trade-offs of distinct attacker access. Furthermore, true unsupervised learning remains an unexplored but vital prospect for realistic black-box scenarios.

Metrics and reproducibility. While DLSCA papers explore different SCA stages (such as trace preprocessing, alignment, and feature engineering), performance is mostly measured using only attack metrics, such as GE or Success Rate. Such use of metrics can lead to the underemphasis of real contributions at distinct, isolated stages of SCA. Different components of the SCA pipeline should instead be evaluated using direct intermediate metrics, such as SNR improvements or alignment accuracy. Additionally, to manage stochastic volatility and expectations during reproduction, automated hyperparameter tuning frameworks and training in general should transition from reporting only the "best single run" to systematically reporting full performance distributions, including success rates and variability across multiple independent seeds. We also invite the community to actively adopt and refine our reproducibility recommendations.

Automating XAI. As discussed in Section 6, understanding neural nets in DLSCA is remarkably tractable, especially on common research targets. While manual approaches can work, automatically retrieving which intermediate values are leaking without access to masking randomness still remains an open problem. Closing this gap would let evaluators localize exploitable leakage directly from a successful attack rather than treating attack success as the end of the analysis. More broadly, automated tooling of this kind would also sharpen the community's understanding of how and when DLSCA models generalize, by making internal analyses simple to run.

Unsupervised GenAI. One of the main areas we consider underrepresented in the DLSCA domain is unsupervised generative approaches. Most works operate in the full or pseudo labels settings, while almost no approaches aim to learn representations in a fully unsupervised manner. While there is some work applying unsupervised objectives to preprocessing and pretraining [57, 64, 71], the broader potential remains largely underexplored. The ultimate goal would be a simple objective that allows for scalable pre-training while minimizing a priori assumptions. If successful, unsupervised generative modeling could serve as a strong pretraining objective for a given target,

producing representations that capture target-specific structure before any assumptions about the leakage model need to be made.

10 Related Work

This community activity, and consequently the number of papers, requires regular systematization of knowledge to help researchers better understand the current state-of-the-art, as well as potential future trends and challenges. Surprisingly, there are not many works in that direction.[8] Picek [97] outlined the open challenges in DLSCA, while Jin et al. [55] surveyed advances in DLSCA up to 2020. More recently, Krček et al. [63] provided an updated overview of developments in DLSCA until 2022.[9] Finally, Picek et al. provided the first systematization of knowledge for DLSCA [100]. While that work established a vital baseline, given the rapid pace of development in DLSCA, a few years already represent a significant period, with more than 200 papers published since then. Moreover, [100] omits critical modern domains such as post-quantum and lightweight algorithms, and limits its discussion of generative artificial intelligence to Generative Adversarial Networks (GANs). Our work directly addresses these gaps beyond simply updating the literature by introducing: (i) the new threat-model taxonomy, (ii) the PIS dataset-difficulty rubric, (iii) the reproducibility rubric/analysis, and (iv) expanded discussion of emerging directions such as GenAI and XAI.

11 Conclusions

Deep learning-based SCA is an extremely active and practically relevant research direction. While the number of DLSCA papers in recent years (more than 500) indicates broad interest within the SCA community, it also complicates the understanding of the main trends and challenges to be addressed. Previous SoK [100] received significant attention and helped establish certain research directions. However, due to a fast-evolving domain, many challenges were not covered. This work systematizes the current state-of-the-art in DLSCA. We provide a methodology to assess dataset difficulty, a novel taxonomy of threat models, a systematization on model selection, XAI, and GenAI, and reproducibility experiments with the proposed reproducibility scores. We also provide 15 takeaway messages. We hope this work helps establish current trends while also identifying challenges that need to be addressed in the future.

[8] We note several works [3, 49] consider MLSCA but not DLSCA.

[9] There are also papers that cannot be considered surveys only, but also have that perspective clearly outlined, see, e.g., [6, 78].

Acknowledgments

This work has been partially supported by the European commission under SHASAI (Secure Hardware and Software for AI Systems, Grant agreement ID 101225866) project, and through the PROACT project with grant number NWA.1215.18.014.

Ethical Considerations

This SoK synthesizes prior work on deep learning–based side-channel attacks using only publicly available sources and introduces no new attack techniques or datasets. The relevant stakeholders are the academic SCA and ML communities, hardware and software vendors whose implementations may be targeted by such attacks, security evaluators and certification bodies who rely on a clear understanding of attacker capabilities, and end users of cryptographic systems who are ultimately affected by their security. While side-channel research is inherently dual-use, the potential harms of this work are limited: we describe no novel attacks and rely entirely on results that are already public, so adversaries gain no capability they did not already have. The benefits, in contrast, accrue to defenders and evaluators: by clarifying threat models, assumptions, and limitations in the existing literature, we aim to make attack capabilities, reproducibility challenges, and mitigation implications easier to reason about, which directly supports better evaluation practice and more informed countermeasure design. On this basis we consider publication clearly justified.

Open Science

We provide all scripts used in this work at https://zenodo.org/records/20340778. This includes scripts for collecting and filtering citing papers from Google Scholar for the generation of Figure 1, as well as scripts to reproduce the experimental results in Appendix E. Note that our experiments utilized the original public repository of [96] essentially as-is, implementing only minor modifications that can be seen in the provided scripts, while the repository of [110] was used without any changes to any of the scripts.

References

[1] I. J. S. 27. Iso/iec 15408-1:2022: Information security, cybersecurity and privacy protection - evaluation criteria for it security, 2022.

[2] R. Y. Acharya, F. Ganji, and D. Forte. Information theory-based evolution of neural networks for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):401–437, 2023.

[3] L. Batina, M. Djukanovic, A. Heuser, and S. Picek. It started with templates: The future of profiling in side-channel analysis. In G. Avoine and J. Hernandez-Castro, editors, *Security of Ubiquitous Computing Systems*, pages 133–145. Springer International Publishing, 2021.

[4] G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, and T. Kouzminov. Test vector leakage assessment (tvla) methodology in practice. In International Cryptographic Module Conference, vol. 1001, 2013.

[5] D. Bellizia, O. Bronchain, G. Cassiers, C. Momin, F.-X. Standaert, and B. Udvarhelyi. Spook SCA CTF, 2021.

[6] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptographic Engineering*, 10(2):163–188, 2020.

[7] L. Bettale, J. Eynard, S. Montoya, G. Renault, and R. Strullu. Security assessment of NTRU against non-profiled SCA. In I. Buhan and T. Schneider, editors, *Smart Card Research and Advanced Applications - 21st International Conference, CARDIS 2022, Birmingham, UK, November 7-9, 2022, Revised Selected Papers*, volume 13820 of *Lecture Notes in Computer Science*, pages 248–268. Springer, 2022.

[8] S. Bhasin, H. Boyapally, D. Jap, T. Yap, and Q. Wu.

[9] S. Bhasin, N. Bruneau, J.-L. Danger, S. Guilley, and Z. Najm. Analysis and improvements of the dpa contest v4 implementation. In R. S. Chakraborty, V. Matyas, and P. Schaumont, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 201–218, Cham, 2014. Springer International Publishing.

[10] S. Bhasin, A. Chattopadhyay, A. Heuser, D. Jap, S. Picek, and R. R. Shrivastwa. Mind the portability: A warriors guide through realistic profiled side-channel analysis. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

[11] S. Bhasin, D. Jap, and S. Picek. AES HD dataset - 50 000 traces. AISyLab repository, 2020. https://github.com/AISyLab/AES_HD.

[12] S. Bhasin, D. Jap, and S. Picek. AES HD dataset - 500 000 traces. AISyLab repository, 2020. https://github.com/AISyLab/AES_HD_2.

[13] S. Boussam, M. Carbone, B. Gérard, G. Renault, and G. Zaid. Optimal dimensionality reduction using conditional variational autoencoder. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(3):164–211, Jun. 2025.

[14] O. Bronchain, F. Durvaux, L. Masure, and F. Standaert. Efficient profiled side-channel analysis of masked implementations, extended. *IEEE Trans. Inf. Forensics Secur.*, 17:574–584, 2022.

[15] O. Bronchain, J. M. Hendrickx, C. Massart, A. Olshevsky, and F.-X. Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In A. Boldyreva and D. Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 713–737, Cham, 2019. Springer International Publishing.

[16] I. Buhan, L. Batina, Y. Yarom, and P. Schaumont. Sok: Design tools for side-channel-aware implementations. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '22, page 756–770, New York, NY, USA, 2022. Association for Computing Machinery.

[17] E. Bursztein, L. Invernizzi, K. Král, D. Moghimi, J. Picod, and M. Zhang. Generalized power attacks against crypto hardware using long-range deep learning. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(3):472–499, 2024.

[18] E. Cagli, C. Dumas, and E. Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without preprocessing. In W. Fischer and N. 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 45–68. Springer, 2017.

[19] P. Cao, C. Zhang, X. Lu, and D. Gu. Cross-device profiled side-channel attack with unsupervised domain adaptation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(4):27–56, 2021.

[20] P. Cao, C. Zhang, X. Lu, D. Gu, and S. Xu. Improving deep learning based second-order side-channel analysis with bilinear CNN. *IEEE Trans. Inf. Forensics Secur.*, 17:3863–3876, 2022.

[21] P. Cao, H. Zhang, D. Gu, Y. Lu, and Y. Yuan. ALPA: cross-device profiled side-channel attack using adversarial learning. In R. Oshana, editor, *DAC '22: 59th ACM/IEEE Design Automation Conference, San Francisco, California, USA, July 10 - 14, 2022*, pages 691–696. ACM, 2022.

[22] G. Cassiers and C. Momin. The SMAesH dataset. Cryptology ePrint Archive, Paper 2024/1521, 2024.

[23] S. Chari, J. R. Rao, and P. Rohatgi. Template attacks. In B. S. K. Jr., Ç. K. Koç, and C. 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.

[24] G. Chiari, D. Galli, F. Lattari, M. Matteucci, and D. Zoni. A deep- learning technique to locate cryptographic operations in side-channel traces. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, page 1–6. IEEE, Mar. 2024.

[25] L. Chmielewski, I. Buhan, and K. Tobich. Reassure (h2020 731591) hw des dataset, 2020.

[26] C. Clavier, J. Danger, G. Duc, M. A. Elaabid, B. Gérard, S. Guilley, A. Heuser, M. Kasper, Y. Li, V. Lomné, D. Nakatsu, K. Ohta, K. Sakiyama, L. Sauvage, W. Schindler, M. Stöttinger, N. Veyrat-Charvillon, M. Walle, and A. Wurcker. Practical improvements of side-channel attacks on AES: feedback from the 2nd DPA contest. *J. Cryptogr. Eng.*, 4(4):259–274, 2014.

[27] G. Cler, S. Ordas, and P. Maurine. Improving leakage exploitability in horizontal side channel attacks through anomaly mitigation with unsupervised neural networks. In *Constructive Approaches for Security Analysis and Design of Embedded Systems: First International Conference, CASCADE 2025, Saint-Etienne, France, April 2–4, 2025, Proceedings*, page 554–579, Berlin, Heidelberg, 2025. Springer-Verlag.

[28] J.-S. Coron and I. Kizhvatov. An efficient method for random delay generation in embedded software. Cryptology ePrint Archive, Paper 2009/419, 2009.

[29] V. Cristiani, M. Lecomte, and P. Maurine. Leakage assessment through neural estimation of the mutual information. In J. Zhou, M. Conti, C. M. Ahmed, M. H. Au, L. Batina, Z. Li, J. Lin, E. Losiouk, B. Luo, S. Majumdar, W. Meng, M. Ochoa, S. Picek, G. Portokalidis, C. Wang, and K. Zhang, editors, *Applied Cryptography and Network Security Workshops*, pages 144–162, Cham, 2020. Springer International Publishing.

[30] D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen. X-deepsca: Cross-device deep learning side channel attack. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[31] dian zi yu xin xi xue bao. DTDS: Dilithium dataset for power analysis, Mar. 2025.

[32] P. Dobias, A. Rezaeezade, L. Chmielewski, L. Malina, and L. Batina. SoK: Reassessing side-channel vulnerabilities and countermeasures in PQC implementations. Cryptology ePrint Archive, Paper 2025/1222, 2025.

[33] P. A. Eliasi, S. Mella, L. Weissbart, L. Batina, and S. Picek. Xoodyak under sca siege. In *2024 27th International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 61–66, 2024.

[34] F. Erata, R. Piskac, V. Mateu, and J. Szefer. Towards automated detection of single-trace side-channel vulnerabilities in constant-time cryptographic code. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 687–706, 2023.

[35] H. Fan. Template Attack against First-Order and Second-Order Masked Dilithium Implementation, Apr. 2025.

[36] Federal Office for Information Security (BSI). Guidelines for Evaluating Machine-Learning based Side-Channel Attack Resistance. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_46_AI_guide.pdf?__blob=publicationFile&v=6, 02 2024. Technical Report AIS 46.

[37] D. Galli, G. Chiari, and D. Zoni. Chameleon: A Dataset for Segmenting and Attacking Obfuscated Power Traces in Side-Channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(3):389–412, June 2025.

[38] D. Galli, F. Lattari, M. Matteucci, and D. Zoni. A deep learning-assisted template attack against dynamic frequency scaling countermeasures. *IEEE Transactions on Computers*, 74(1):293–306, 2025.

[39] J. Gammell, A. Raghunathan, A. Hashemi, and K. Roy. Learning to localize leakage of cryptographic sensitive variables. *arXiv preprint arXiv:2503.07464*, 2025.

[40] S. Gao and E. Oswald. A novel framework for explainable leakage assessment. In M. Joye and G. Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part III*, volume 14653 of *Lecture Notes in Computer Science*, pages 221–250. Springer, 2024.

[41] C. Genevey-Metat, A. Heuser, and B. Gérard. Trace-to-trace translation for SCA. In V. Grosso and T. 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 24–43. Springer, 2021.

[42] A. Gohr, F. Laus, and W. Schindler. Breaking masked implementations of the clyde-cipher by means of side-channel analysis: A report on the ches challenge side-channel contest 2020. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):397–437, Aug. 2022.

[43] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[44] S. Guilley. *Geometrical counter-measures against side-channel attacks*. PhD thesis, PhD thesis, ENST/CNRS LTCI, 2007.

[45] S. Hajra, S. Chowdhury, and D. Mukhopadhyay. Estranet: An efficient shift-invariant transformer network for side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2024(1):336–374, 2024.

[46] S. Hajra, S. Saha, M. Alam, and D. Mukhopadhyay. Transnet: Shift invariant transformer network for side channel analysis. In L. Batina and J. Daemen, editors, *Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa, AFRICACRYPT 2022, Fes, Morocco, July 18-20, 2022, Proceedings*, volume 13503 of *Lecture Notes in Computer Science*, pages 371–396. Springer Nature Switzerland, 2022.

[47] B. Hettwer, S. Gehrer, and T. Güneysu. Profiled power analysis attacks using convolutional neural networks with domain knowledge. In C. Cid and M. J. J. Jr., editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 479–498. Springer, 2018.

[48] B. Hettwer, S. Gehrer, and T. Güneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. In K. G. Paterson and D. Stebila, editors, *Selected Areas in Cryptography - SAC 2019 - 26th International Conference, Waterloo, ON, Canada, August 12-16, 2019, Revised Selected Papers*, volume 11959 of *Lecture Notes in Computer Science*, pages 645–666. Springer, 2019.

[49] B. Hettwer, S. Gehrer, and T. Güneysu. Applications of machine learning techniques in side-channel attacks: a survey. *J. Cryptogr. Eng.*, 10(2):135–162, 2020.

[50] J. Heyszl, A. Ibing, S. Mangard, F. De Santis, and G. Sigl. Clustering algorithms for non-profiled single-execution attacks on exponentiations. In *Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, page 79–93, Berlin, Heidelberg, 2013. Springer-Verlag.

[51] V.-P. Hoang, N.-T. Do, and V. S. Doan. Efficient non-profiled side-channel attack using multi-output classification neural network. *IEEE Embedded Systems Letters*, 15(3):145–148, 2023.

[52] F. Hu, J. Shen, and P. Vijayakumar. Side-channel attacks based on multi-loss regularized denoising autoencoder. *IEEE Trans. Inf. Forensics Secur.*, 19:2051–2065, 2024.

[53] A. Ito, R. Ueno, and N. Homma. Perceived information revisited: New metrics to evaluate success rate of side-channel attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):228–254, Aug. 2022.

[54] A. Ito, R. Ueno, and N. Homma. Perceived information revisited ii: Information-theoretical analysis of deep-learning based side-channel attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(1):450–474, Dec. 2024.

[55] S. Jin, S. Kim, H. Kim, and S. Hong. Recent advances in deep learning-based side-channel analysis. *Etri Journal*, 42(2):292–304, 2020.

[56] S. Karayalcin, M. Krček, and S. Picek. Interpreting emergent features in deep learning-based side-channel analysis. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[57] S. Karayalcin, M. Krcek, and S. Picek. A practical tutorial on deep learning-based side-channel analysis. *IACR Cryptol. ePrint Arch.*, page 471, 2025.

[58] S. Karayalcin, M. Krcek, L. Wu, S. Picek, and G. Perin. It's a kind of magic: A novel conditional GAN framework for efficient profiling side-channel analysis. In K. Chung and Y. Sasaki, editors, *Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part VIII*, volume 15491 of *Lecture Notes in Computer Science*, pages 99–131. Springer, 2024.

[59] S. Karayalçin, G. Perin, and S. Picek. Diffuse some noise: Diffusion models for measurement noise removal in side-channel analysis. In C. Boura,

A. Mashatan, and A. Miri, editors, *Selected Areas in Cryptography – SAC 2025*, pages 724–752, Cham, 2026. Springer Nature Switzerland.

[60] J. Kim, S. Picek, A. Heuser, S. Bhasin, and A. Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(3):148–179, May 2019.

[61] P. C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In N. Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[62] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. 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.

[63] M. Krcek, H. Li, S. Paguada, U. Rioja, L. Wu, G. Perin, and L. Chmielewski. Deep learning on side-channel analysis. In *Security and Artificial Intelligence*, pages 48–71. Springer International Publishing, 2022.

[64] M. Krcek and G. Perin. Autoencoder-enabled model portability for reducing hyperparameter tuning efforts in side-channel analysis. *J. Cryptogr. Eng.*, 14(3):475–497, 2024.

[65] M. Krček, L. Wu, G. Perin, and S. Picek. Shift-invariance robustness of convolutional neural networks in side-channel analysis. *Mathematics*, 12(20), 2024.

[66] P. Kulkarni, V. Verneuil, S. Picek, and L. Batina. Order vs. chaos: A language model approach for side-channel attacks. Cryptology ePrint Archive, Paper 2023/1615, 2023.

[67] S. Kumar, V. A. Dasu, A. Baksi, S. Sarkar, D. Jap, J. Breier, and S. Bhasin. Side channel attack on stream ciphers: A three-step approach to state/key recovery. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(2):166–191, 2022.

[68] H. Li, M. Ninan, B. Wang, and J. M. Emmert. Tinypower: Side-channel attacks with tiny neural networks. In *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2024, Tysons Corner, VA, USA, May 6-9, 2024*, pages 320–331. IEEE, 2024.

[69] L. Li and Y. Ou. A deep learning-based side channel attack model for different block ciphers. *Journal of Computational Science*, 72:102078, 2023.

[70] Y. Li, J. Zhu, Z. Liu, M. Tang, and S. Ren. Deep learning gradient visualization-based pre-silicon side-channel leakage location. *IEEE Transactions on Information Forensics and Security*, 19:2340–2355, 2024.

[71] A. Liu, A. Wang, S. Sun, C. Wei, Y. Ding, Y. Wang, and L. Zhu. CL-SCA: A contrastive learning approach for profiled side-channel analysis. *IEEE Trans. Inf. Forensics Secur.*, 20:5109–5122, 2025.

[72] X. Lu, C. Zhang, P. Cao, D. Gu, and H. Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):235–274, 2021.

[73] X. Lu, C. Zhang, P. Cao, D. Gu, and H. Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):235–274, Jul. 2021.

[74] H. Maghrebi, T. Portigliatti, and E. Prouff. Breaking cryptographic implementations using deep learning techniques. In C. Carlet, M. A. Hasan, and V. Saraswat, editors, *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, volume 10076 of *Lecture Notes in Computer Science*, pages 3–26. Springer, 2016.

[75] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks: Revealing the secrets of smart cards*, volume 31. Springer Science & Business Media, 2008.

[76] S. Marzougui, N. Wisiol, P. Gersch, J. Krämer, and J.-P. Seifert. Machine-learning side-channel attacks on the galactics constant-time implementation of bliss. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ARES 2022, page 1–11. ACM, Aug. 2022.

[77] L. Masure, V. Cristiani, M. Lecomte, and F. Standaert. Don't learn what you already know scheme-aware modeling for profiling side-channel analysis against masking. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(1):32–59, 2023.

[78] L. Masure, C. Dumas, and E. Prouff. A comprehensive study of deep learning for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):348–375, Nov. 2019.

[79] L. Masure, C. Dumas, and E. Prouff. Gradient visualization for general characterization in profiling attacks. In I. Polian and M. Stöttinger, editors, *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, volume 11421 of *Lecture Notes in Computer Science*, pages 145–167. Springer, 2019.

[80] L. Masure and R. Strullu. Side-channel analysis against anssi's protected AES implementation on ARM: end-to-end attacks with multi-task learning. *J. Cryptogr. Eng.*, 13(2):129–147, 2023.

[81] T. Moos, F. Wegener, and A. Moradi. Dl-la: Deep learning leakage assessment: A modern roadmap for sca evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):552–598, Jul. 2021.

[82] N. Mukhtar, L. Batina, S. Picek, and Y. Kong. Fake it till you make it: Data augmentation using generative adversarial networks for all the crypto you need on small devices. In S. D. Galbraith, editor, *Topics in Cryptology - CT-RSA 2022 - Cryptographers' Track at the RSA Conference 2022, Virtual Event, March 1-2, 2022, Proceedings*, volume 13161 of *Lecture Notes in Computer Science*, pages 297–321. Springer, 2022.

[83] M. Napoli, A. Leporati, S. Picek, and L. Mariot. Mind the grammar: Side-channel analysis driven by grammatical evolution. Cryptology ePrint Archive, Paper 2025/698, 2025.

[84] E. Nascimento, Ł. Chmielewski, D. Oswald, and P. Schwabe. Attacking embedded ecc implementations through cmov side channels. In R. Avanzi and H. Heys, editors, *Selected Areas in Cryptography – SAC 2016*, pages 99–119, Cham, 2017. Springer International Publishing.

[85] M. Nassar, Y. Souissi, S. Guilley, and J. Danger. RSM: A small and fast countermeasure for aes, secure against 1st and 2nd-order zero-offset scas. In W. Rosenstiel and L. Thiele, editors, *2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, Dresden, Germany, March 12-16, 2012*, pages 1173–1178. IEEE, 2012.

[86] M. Ninan, E. Nimmo, S. Reilly, C. Smith, W. Sun, B. Wang, and J. M. Emmert. A second look at the portability of deep learning side-channel attacks over EM traces. In E. Losiouk, A. Brighente, M. Conti, Y. Aafer, and Y. Fratantonio, editors, *The 27th International Symposium on Research in Attacks, Intrusions and Defenses, RAID 2024, Padua, Italy, 30 September 2024- 2 October 2024*, pages 630–643. ACM, 2024.

[87] Northeastern University TeSCASE. Northeastern University TeSCASE dataset. https://chest.coe.neu.edu/.

[88] C. Olah, N. Cammarata, C. Voss, L. Schubert, and G. Goh. Naturally occurring equivariance in neural networks. *Distill*, 2020. https://distill.pub/2020/circuits/equivariance.

[89] S. Paguada, L. Batina, and I. Armendariz. Toward practical autoencoder-based side-channel analysis evaluations. *Computer Networks*, 196:108230, 2021.

[90] S. Paguada, L. Batina, I. Buhan, and I. Armendariz. Being patient and persistent: Optimizing an early stopping strategy for deep learning in profiled attacks. *IEEE Transactions on Computers*, 74(3):875–886, 2025.

[91] D. Perez, S. Karayalcin, S. Picek, and S. Paguada. Enhancing scale and shift invariance in deep learning-based side-channel attacks through equivariant convolutional neural networks. Cryptology ePrint Archive, Paper 2025/1379, 2025.

[92] D. Perez, S. Karayalcin, S. Picek, and S. Paguada. Enhancing scale and shift invariance in deep learning-based side-channel attacks through equivariant convolutional neural networks. *IACR Cryptol. ePrint Arch.*, page 1379, 2025.

[93] G. Perin, I. Buhan, and S. Picek. Learning when to stop: A mutual information approach to prevent overfitting in profiled side-channel analysis. In S. Bhasin and F. De Santis, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 53–81, Cham, 2021. Springer International Publishing.

[94] G. Perin, L. Chmielewski, L. Batina, and S. Picek. Keep it unsupervised: Horizontal attacks meet deep learning. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):343–372, Dec. 2020.

[95] G. Perin, S. Karayalçin, L. Wu, and S. Picek. I know what your layers did: Layer-wise explainability of deep learning side-channel analysis. *IACR Communications in Cryptology*, 2(4), 2026.

[96] G. Perin, L. Wu, and S. Picek. Exploring feature selection scenarios for deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(4):828–861, Aug. 2022.

[97] S. Picek. Challenges in deep learning-based profiled side-channel analysis. In S. Bhasin, A. Mendelson, and M. Nandi, editors, *Security, Privacy, and Applied*

Cryptography Engineering - 9th International Conference, SPACE 2019, Gandhinagar, India, December 3-7, 2019, Proceedings, volume 11947 of *Lecture Notes in Computer Science*, pages 9–12. Springer, 2019.

[98] S. Picek, A. Heuser, A. Jovic, S. Bhasin, and F. Regazzoni. The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):1–29, 2019.

[99] S. Picek, A. Heuser, G. Perin, and S. Guilley. Profiled side-channel analysis in the efficient attacker framework. In *Smart Card Research and Advanced Applications: 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11–12, 2021, Revised Selected Papers*, page 44–63, Berlin, Heidelberg, 2021. Springer-Verlag.

[100] S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11), Feb. 2023.

[101] J.-J. Quisquater and D. Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *International Conference on Research in Smart Cards*, pages 200–210. Springer, 2001.

[102] A. Rezaeezade, A. Basurto-Becerra, L. Weissbart, and G. Perin. One for all, all for ascon: Ensemble-based deep learning side-channel analysis. In M. Andreoni, editor, *Applied Cryptography and Network Security Workshops*, pages 139–157, Cham, 2024. Springer Nature Switzerland.

[103] A. Rezaeezade, T. Yap, D. Jap, S. Bhasin, and S. Picek. Breaking the blindfold: deep learning-based blind side-channel analysis. In *Proceedings of the 34th USENIX Conference on Security Symposium*, SEC '25, USA, 2025. USENIX Association.

[104] A. Rezaeezade, T. Yap, D. Jap, S. Bhasin, and S. Picek. Side-channel power trace dataset for kyber pair-pointwise multiplication on cortex-m4. Cryptology ePrint Archive, Paper 2025/811, 2025.

[105] J. Rijsdijk, L. Wu, G. Perin, and S. Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):677–707, 2021.

[106] U. Rioja, L. Batina, I. Armendariz, and J. L. Flores. Towards human dependency elimination: AI approach to SCA robustness assessment. *IEEE Trans. Inf. Forensics Secur.*, 17:3906–3921, 2022.

[107] U. Rioja, L. Batina, J. L. Flores, and I. Armendariz. Auto-tune pois: Estimation of distribution algorithms for efficient side-channel analysis. *Comput. Networks*, 198:108405, 2021.

[108] Riscure. CHES CTF 2018. Website, 2018. https://chesctf.riscure.com/2018/news.

[109] D. Robissout, G. Zaid, B. Colombier, L. Bossuet, and A. Habrard. Online performance evaluation of deep learning networks for profiled side-channel analysis. In *Constructive Side-Channel Analysis and Secure Design: 11th International Workshop, COSADE 2020, Lugano, Switzerland, April 1–3, 2020, Revised Selected Papers*, page 200–218, Berlin, Heidelberg, 2020. Springer-Verlag.

[110] N. Rousselot, K. Heydemann, L. Masure, and V. Migairou. Scoop: An optimization algorithm for profiling attacks against higher-order masking. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(3):56–80, Jun. 2025.

[111] F. Schijlen, L. Wu, and L. Mariot. Nascty: Neuroevolution to attack side-channel leakages yielding convolutional neural networks. *Mathematics*, 11(12), 2023.

[112] D. Shanmugam and P. Schaumont. Improving side-channel leakage assessment using pre-silicon leakage models. In E. B. Kavun and M. Pehl, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 105–124, Cham, 2023. Springer Nature Switzerland.

[113] M. Staib and A. Moradi. Deep learning side-channel collision attack. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):422–444, Jun. 2023.

[114] B. Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):107–131, 2019.

[115] R. Ueno, K. Xagawa, Y. Tanaka, A. Ito, J. Takahashi, and N. Homma. Curse of re-encryption: A generic power/EM analysis on post-quantum kems. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022(1):296–322, Nov. 2021.

[116] A. Vasselle, H. Thiebeauld, and P. Maurine. Spatial dependency analysis to extract information from side-channel mixtures: extended version. *Journal of Cryptographic Engineering*, 13(4):409–425, 2023.

[117] P. Wang, P. Chen, Z. Luo, G. Dong, M. Zheng, N. Yu, and H. Hu. Enhancing the performance of practical profiling side-channel attacks using conditional generative adversarial networks. *CoRR*, abs/2007.05285, 2020.

[118] L. Weissbart, S. Picek, and L. Batina. One trace is all it takes: Machine learning-based side-channel attack on eddsa. In S. Bhasin, A. Mendelson, and M. Nandi, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 86–105, Cham, 2019. Springer International Publishing.

[119] L. Wouters, V. Arribas, B. Gierlichs, and B. Preneel. Revisiting a methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):147–168, Jun. 2020.

[120] L. Wu, G. Perin, and S. Picek. On the evaluation of deep learning-based side-channel analysis. *IACR Cryptol. ePrint Arch.*, page 952, 2021.

[121] L. Wu, G. Perin, and S. Picek. The best of two worlds: Deep learning-assisted template attack. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(3):413–437, 2022.

[122] L. Wu, G. Perin, and S. Picek. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. *IEEE Trans. Emerg. Top. Comput.*, 12(2):546–557, 2024.

[123] L. Wu, G. Perin, and S. Picek. Weakly profiling side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2024(3):707–730, Nov. 2024.

[124] L. Wu and S. Picek. Remove some noise: On pre-processing of side-channel measurements with autoencoders. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(4):389–415, 2020.

[125] L. Wu, S. Tiran, G. Perin, and S. Picek. Plaintext-based side-channel collision attack. *IACR Communications in Cryptology*, 1(3), 2024.

[126] L. Wu, L. Weissbart, M. Krček, H. Li, G. Perin, L. Batina, and S. Picek. Label correlation in deep learning-based side-channel analysis. *Trans. Info. For. Sec.*, 18:3849–3861, Jan. 2023.

[127] C. Xiao, M. Tang, S. Karayalcin, and W. Cheng. LDPA: distilling univariate leakage for deep learning-based profiling attacks. *IEEE Trans. Inf. Forensics Secur.*, 20:17–30, 2025.

[128] G. Yang, H. Li, J. Ming, and Y. Zhou. CDAE: towards empowering denoising in side-channel analysis. In J. Zhou, X. Luo, Q. Shen, and Z. Xu, editors, *Information and Communications Security - 21st International Conference, ICICS 2019, Beijing, China, December 15-17, 2019, Revised Selected Papers*, volume 11999 of *Lecture Notes in Computer Science*, pages 269–286. Springer, 2019.

[129] T. Yap, A. Benamira, S. Bhasin, and T. Peyrin. Peek into the black-box: Interpretable neural network using SAT equations in side-channel analysis. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):24–53, 2023.

[130] T. Yap and D. Jap. Creating from noise: Trace generations using diffusion model for side-channel attack. In M. Andreoni, editor, *Applied Cryptography and Network Security Workshops - ACNS 2024 Satellite Workshops, AIBlock, AIHWS, AIoTS, SCI, AAC, SiMLA, LLE, and CIMSS, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part I*, volume 14586 of *Lecture Notes in Computer Science*, pages 102–120. Springer, 2024.

[131] T. Yap, S. Picek, and S. Bhasin. Occpois: Points of interest based on neural network's key recovery in side-channel analysis through occlusion. In S. Mukhopadhyay and P. Stanica, editors, *Progress in Cryptology - INDOCRYPT 2024 - 25th International Conference on Cryptology in India, Chennai, India, December 18-21, 2024, Proceedings, Part II*, volume 15496 of *Lecture Notes in Computer Science*, pages 3–28. Springer, 2024.

[132] K. Yoshida, S. Karayalcin, and S. Picek. Can kans do it? toward interpretable deep learning-based side-channel analysis. In M. Manulis, editor, *Applied Cryptography and Network Security Workshops - ACNS 2025 Satellite Workshops: AIHWS, AIoTS, QSHC, SCI, PrivCrypt, SPIQE, SiMLA, and CIMSS 2025, Munich, Germany, June 23-26, 2025, Revised Selected Papers, Part I*, volume 15653 of *Lecture Notes in Computer Science*, pages 40–57. Springer, 2025.

[133] G. Zaid, L. Bossuet, M. Carbone, A. Habrard, and A. Venelli. Conditional variational autoencoder based on stochastic attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2023(2):310–357, 2023.

[134] G. Zaid, L. Bossuet, F. Dassance, A. Habrard, and A. Venelli. Ranking loss: Maximizing the success rate in deep learning side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(1):25–55, Dec. 2020.

[135] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli. Methodology for efficient CNN architectures in profiling attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):1–36, 2020.

[136] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli. Efficiency through diversity in ensemble models applied to side-channel attacks: – a case study on public-key algorithms –. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):60–96, Jul. 2021.

[137] J. Zhang, M. Zheng, J. Nan, H. Hu, and N. Yu. A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(3):73–96, Jun. 2020.

[138] Z. Zhang, A. A. Ding, and Y. Fei. A fast and accurate guessing entropy estimation algorithm for full-key recovery. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(2):26–48, Mar. 2020.

[139] J. Zhu and J. Lu. Leading degree: A metric for model performance evaluation and hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(2):333–361, Mar. 2025.

A Publicly Available Datasets

We list publicly available datasets for side-channel attack benchmarks, along with basic information in Table 5. Note that datasets from the website `aisylabdatasets.ewi.tudelft.nl` are no longer available for download. While other websites might host them, we did not find DPAv1, AES_HD, and AES_HD_Ext available for download.

The reference [100] suggests using the same format for the datasets: we observe that this is not consistently applied. In the list of datasets, one can find various formats, including npz, trs, and mat. Thus, more effort should be made to standardize the format to improve reproducibility and ease of use across different public datasets for benchmarking.

B Dataset Recommendations

We briefly discuss concrete dataset recommendations for AES. For algorithms besides AES, the lack of publicly available datasets and results makes it difficult to provide clear recommendations at this time.

The ASCADv1 datasets (with the pre-selected window) are still the most common benchmark in DLSCA. While the dataset is considered relatively easy for current approaches, it still provides a well-understood test case for evaluating new approaches. The availability of masking randomness also allows for more controlled experiments to validate that a method is performing as intended. As such, ASCADv1 (and more specifically, the variable key version) should still be used for exploratory testing of new approaches. One of the main advantages of ASCADv1 is that leakage from several operations is included, and specific bits leak, as opposed to the Hamming weight of the full byte.

For approaches that aim to explore class imbalance issues for DLSCA, the non-shuffled version of eShard represents a relatively easy benchmark.

To test the broader applicability of approaches to more black-box scenarios using (resampled) raw traces, the dataset versions from [96] provide a good benchmark. A fairly significant effort in architecture/hyperparameter tuning is required to make approaches work, especially for CHES_CTF 2018 and DPAv4.2.

More difficult datasets, like ASCADv2, SMAesH, and GE_wars, should be used for testing whether general improvements in earlier performance can result in more practical attacks. At the same time, the size and difficulty in attacking these datasets make it difficult to draw robust conclusions about the general efficacy of a specific approach. For testing attacks against higher-order masking, the only two datasets available are ASCADv2 and Spook. The main limitation with ASCADv2 is that, due to the low leakage of some shares, it remains difficult to say whether it is possible to learn the mask recombination in practice. For Spook, the only attacks that work in black-box settings target a specific bit of the state due to the imbalanced leakage [45], and these only work for 2nd-order masking.

C Threat Model Examples

Table 6 provides concrete examples of works belonging to all threat model categories we presented in Section 4.

D Evaluation Metrics

Next, we enumerate the metrics appearing in the last few years to help evaluate the behavior of DLSCA.

- Key Guessing vector: search success [96], guessing entropy estimation algorithm (GEEA) [138], Cross Entropy Ratio (CER) [137], comparing the success rate discrepancy on the training/validation sets to judge the generalization capacity of a model [109], area of hit [90], Label Correlation (LD) [126], (simplified) Leading Degree(LD) [139], different summary statistics on key guessing vectors [120].
- Mutual Information (MI): MI transferred to the output layer [93], Perceived information (PI) [78], Hypothetical Information (HI) [15], Ranking Loss (RkL) [134], Ensembling Loss (EL) [136], efficient cross-entropy (ECE) and efficient PI (EPI) [53], latent (LPI) [54].
- Minimal number of training traces [99], number of evaluated neural networks [96][10].

E Reproducibility Experiments

We conduct the experiments on a high-performance computing environment (HPC) equipped with NVIDIA GeForce RTX 2080 Ti (11GB) GPUs. When reproducing related works, we

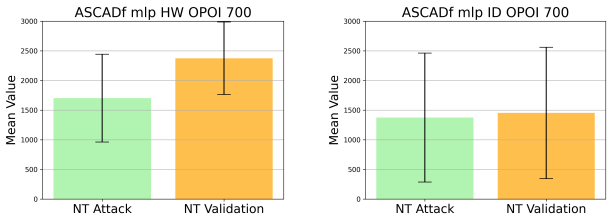
[10] This metric is connected with the search success metric.

Dataset	Platform	Traces (Features)	Keys	Implementation	Countermeasures
DPAv1 [44] (2008)	home-made acquisition platform		Fixed	DES	
DPAv2 [26] (2010)	SASEBO GII (FPGA)	1 000 000 (3 253) power traces	Random keys	AES128	
DPAv4 [85] link (2013)	Atmel ATMega- 163 (Software)	100 000 (4350 02)	1 fixed key	AES256	First-order masking (RSM)
DPAv4_2 [9] link (2014)	Atmel ATMega- 163 (Software)	80 000 (1 704 400)	16 different keys	AES128	First-order masking (RSM)
AES_HD [11] link (2020)	SASEBO GII (FPGA)	50 000 (1 250)	1 fixed key	AES128	
AES_HD_Ext [12] (2020)	SASEBO GII (FPGA)	500 000 (1 250)	1 fixed key	AES128	
AES_HD_MM [87] link (2014)	SASEBO GII (FPGA)	5 600 000 (3 125)	1 fixed key	AES128	Boolean + Affine Masking
AES_RD [28] link (2009)	Atmel AVR (Software)	50 000 (3 500)	1 fixed key	AES128	Hiding (Random Delay Interrupt)
ASCADf [6] link (2018)	Atmega (Software)	60 000 (100 000)	1 fixed key	AES128	First-order boolean masking
ASCADv1 [6] link (2018)	Atmega (Software)	300 000 (250 000)	Random keys + 1 fixed key	AES128	First-order boolean masking
ASCADv2 [80] link(2021)	STM32 (Software)	810 000 (1 000 000)	Random + 1 fixed key	AES128	Affine Masking + Hiding (Shuffling)
CHES_CTF 2018 [108] link(2018)	STM32 (Software)	42 000 (650 000)	Random Keys + 3 fixed keys	AES128	First-order BM (XOR)
Portability [10] (2020)	Atmega (Software)	50 000 (600)	4 fixed keys	AES128	
SMAesH [22] link(2023)	Xilinx Artix-7 and Spartan-6 (FPGAs)	6×2^{24} (4 450/4 400), power	Random + Fixed key	AES128	First-order masking
eShard [116] link (2021)	STM32F446 (Software)	90 000/10 000 (1 400), EM	Fixed key	AES128	Boolean masking (+ simulated shuffling)
CrossEM [86] link(2024)	AVR XMEGA (software) STM32F3(Software) Atmega	3 million EM traces from 9 probe locations over 4 devices, 150 000 (5 000) (3 000), power	Fixed key	AES128	Random delay
X-DeepSCA [30] link (2020)	Nucleo-F429RZ (nucleo Cortex-M)	150 000 (90 000 and 50 000 with 96MHz and 168MHz, resp.), EM	fixed key	AES128	no countermeasures
DFS_DESYNCH [38] link (2025)	STM32 (Software)	6 400 (1 000), power	Random Ephemeral keys	EdDSA	
GE_wars [8] link (2025)		256 000 (200 000), power	16 keys	AES128	DFS desynchronization
Chameleon [37] link (2025)	ARM Cortex-A72 (Software)	500K profiling, 100K attack traces (7 000), EM	Random keys + fixed key	AES128	Jitter and noise, no masking
Kecak (and AES) [115] link (2021)	RISC-V CPU (Software) (Software & hardware)	256, 256, 512, 512, and 1 024 (134 217 550), power	Segment traces to detect crypto	AES128 Kecak, AES	Dynamic frequency scaling, random delay, morphing, and chaffing Unprotected and masked AES, unprotected Keccak
AES_PTv2 [106, 107] link (2022)	STM32F4 and Riscue Piñata	600 000 (× 4 devices), power/EM	random and fixed key	AES128	unprotected, and masked AES
TinyPower [68] link (2024)	AVR XMEGA and ARM STM32 and FPGAs (Xilinx Artix-7)	2.6 million (5 000 and 200), power and EM	fixed key	AES128	unmasked
Nucleo datasets [92] link (2025)	Nucleo-F429RZ (nucleo Cortex-M)	150 000 (90 000 and 50 000 with 96MHz and 168MHz, resp.), EM	fixed key	AES128	no countermeasures
Ed25519 (WolfSSL) [118] link (2019)	STM32 (Software)	6 400 (1 000), power	Random Ephemeral keys	EdDSA	
Curve25519 (μNaCl) REASSURE [25] link (2020)	STM32F4 (Software)	5 997 (5 500)	Random Ephemeral keys	Curve25519 NaCl Montgomery Ladder scalar multiplication	CSWAP, Coord./Scalar Rand.
Curve25519 [84] (2016)	STM32 (Software)	300 (8 000)	Random keys	EdDSA	CSWAP, Coord./Scalar Rand.
Curve25519 [84] (2016)	STM32 (Software)	300 (1 000)	Random keys	EdDSA	CSPOINTER, Coord./Scalar Rand.
SCAAML ECC CM0 [17] link (2024)	NXP K82F (Hardware accelerated)	57 344/16,384 (1 600 000), power	Random Ephemeral keys	ECC scalar multiplication	
SCAAML ECC CM1 [17] link (2024)	NXP K82F (Hardware accelerated)	194 544/16,384 (5 000 000), power	Random Ephemeral keys	ECC scalar multiplication	Additive blinding
SCAAML ECC CM2 [17] link(2024)	NXP K82F (Hardware accelerated)	112 880/16,384 (10 000 000), power	Random Ephemeral keys	ECC scalar multiplication	multiplicative blinding
SCAAML ECC CM3 [17] link(2024)	NXP K82F (Hardware accelerated)	112 880/16,384 (17 500 000), power	Random Ephemeral keys	ECC scalar multiplication	Combination of CM1 + CM2
Spook [5] link (2020)	Software/Hardware	200 000/100 000 †	Fixed/random	Clyde-128	ISW masking ⋆
ASCON [102] link(2024)	STM32F4(Software)	50 000/10 000 (772), power	Random/Fixed	Ascon-128 v1.2	
ASCON protected [102] link(2024)	STM32F4(Software)	500 000/60 000 (1 408), power	Random/Fixed	Ascon-128 v1.2	Domain Oriented Masking
TRIVIUM [67] link(2022)	ARM Cortex-M3	2 362 000, EM		TRIVIUM	
Xoodyak [33] (2024)	Sakura-G FPGA (Hardware)	200 000/50 000 (5 600), power	Random/Fixed	Xoodyak	
PRESENT (and AES) [69] link (2023)	Sakura-G (FPGA)	50 000 (10 000/2 500), power		PRESENT, AES	
PRESENT [8] link (2021)	SAKURA-G (Spartan-6 FPGA)	1 000-50 000 000 (2 000-200 000), voltage drop over a 1 ■ shunt resistor		PRESENT	(Un)Protected, (Mis)Aligned, Randomized Clock
Dilithium [31] link (2025)	Cortex M4	60 000, power		Dilithium	
Dilithium [35] link. (2025)		POIs, power		Dilithium	First and second-order masking
Kyber [104] link (2025)	STM32F3 (ARM Cortex-M4)	100 000 (500 000), power		Kyber pair-pointwise multiplication in NTT Domain	
BLISS [76] link. (2022)	Cortex-M4 (software)	1 532/1 450 492 power traces	random key + fixed key	GALACTICS (BLISS)	
SCANTRU [7] link (2022)	STM32F407	20 000 (700 000), EM	Random keys	NTRU	Shuffling

Table 5: Publicly available datasets for side-channel analysis research (extended from [100]). † the number of measurements and features will depend on the acquisition setup, as the authors provide source code. ⋆ ISW masking implemented with 3, 4, 6, and 8 shares. There are several more available datasets, but due to the dataset being too "simple" or a lack of support (or other reasons), those datasets are not commonly used.

Table 6: Concrete examples mapped to the two-axis DLSCA threat-model taxonomy (goal × supervision state). Note that [50] is not a deep learning, but a machine learning approach. Furthermore, a single paper may instantiate multiple cells depending on the training objective; this table classifies techniques by their primary learning goal.

Supervision state	Preprocessing	Feature engineering	Key recovery	Leakage assessment
Full labels	[24] [18]	[96] [13]	[135] [73] [123]	[81] [29]
Pseudo labels	[114] [94]	[113] [125]	[51] [103]	[34] [70]
No labels	[124] [59]	[89] [19]	[50]	[27]



(a) HW leakage model. (b) ID leakage model.

Figure 2: Mean of the number of attack traces (T_{GE1}) from 100 random MLP models for the ASCADf dataset in the OPOI setting for both HW and ID leakage models.

use their code with the same versions of Python and libraries, such as TensorFlow and Keras, and PyTorch. The goal here is not to point fingers at specific works, but to highlight pitfalls that DLSCA researchers should be aware of. As such, we select case studies that illustrate the variability of experiments across different axes. Case study 1 shows that retraining identical models can vary significantly, case study 2 shows the variability across hyperparameter searches, and case study 3 shows the benefits of providing a trained model.

Case Study 1 We selected the "Exploring Feature Selection Scenarios for Deep Learning-based Side-channel Analysis" paper, investigating different feature selection methods in DLSCA [96]. This paper provides source code, and following the instructions, it was fairly easy to run and try to reproduce the reported results. There was a minor mistake in the command examples, but this was easily noticed and corrected. The best models, as reported in the paper, are provided in code with the best found hyperparameters and a random seed set. However, it is not a trained model, and no weights are provided, which leads to the issues shown in our results in Table 7 and discussed in more detail in Appendix E. **Thus, if one requires exact reproducibility, the trained model should be directly provided. Otherwise, the results should include error bars to understand the average behavior and variance, along with statistical significance of the results compared to baselines.** In our experiments, shown in Table 7, we reproduce results on ASCADf with three feature selection methods, OPOI, RPOI, and NOPOI, on both MLP and CNN models and HW and ID leakage models. We report the number of attack traces needed for $GE = 1$. We see that in all cases, we obtained worse results, where we needed more attack traces. Note that in some cases, we did not reach $GE = 1$ on the first run, but we then ran again and reported the first successful results (reaching $GE = 1$). Appendix E provides more statistical data, but here, aligned with [96], we report results from one run (training and GE calculation).

Using the Wilcoxon Signed-Rank Test, we obtain $p = 0.005$ for comparison of reported and reproduced values (number of attack traces). The reproduction study showed a statistically significant increase in the number of traces required to reach $GE = 1$ ($p < 0.05$ with the Wilcoxon signed-rank test), suggesting that the original reported performance may be difficult to reach under the current reproduction parameters. There are also some other discrepancies observed. For example, [96] reports 62 396 trainable parameters for the CNN model and the ID leakage model in the OPOI case, but when running the provided scripts, we have 7 776 trainable parameters. Next, for the NOPOI case, while the authors report 2 500 POIs (both leakage models), the best model provided expects 10 000, so our results use 10 000 POIs. In this work, the baseline results are not reported directly, but the authors note that their results align with state-of-the-art performance on the OPOI case, the most common scenario in the literature. In conclusion, providing single run values impedes realistic expectation upon reproducing, and providing a full description of models with random seed but without specific weight values may not be enough to achieve successful reproduction of the results.

Table 7: Reproducing results on ASCADf datasets from [96].

ASCADf	Model Type	Reported T_{GE1} (HW/ID)	Reproduced T_{GE1} (HW/ID)
OPOI	MLP	480/104	615/137
	CNN	744/87	1012/138
RPOI	MLP	5/1	9/2
	CNN	5/1	8/2
NOPOI	MLP	7/1	7/4
	CNN	7/1	24/1

Case Study 2 We conducted experiments to demonstrate variability in results due to the stochastic nature of DLSCA and to motivate more statistically relevant reporting of results in the field. We perform a **random search** for model hyperparameter tuning following the ranges from [96] for the ASCADf dataset and the MLP model. For the MLP model and the HW leakage model, we show the number of attack traces ($T_{GE=1}$). In the random search, from 100 models (trained), the minimum number of traces needed was 871 for validation

and 607 for the attack set. Note that the best model reported in [96] required 480 traces. Regarding the success rate of the tuning with random search, we achieve 85% of the 100 models reaching $GE = 1$, while [96] reported 82.80%. Figure 2a shows the mean value of the number of attack traces, and the standard deviation (612.06 for validation and 740.68 for the attack set).

For the MLP model with the ID leakage model, we present results in Figure 2b. In this setup, the minimum number of attack traces in 100 random models was 96 for validation and 123 for the attack set. Note that the best model in [96] required 104 traces. The success rate of the random search is 74%, while [96] reported 68.8%. Given that, the standard deviation of the number of attack traces is 1108.37 for validation and 1088.19 for the attack set, as shown in Figure 2b.

Table 8: Variations of T_{GE1} from different GE calculations on a single trained model (best from [96]) on the ASCADf dataset. Each row represents one training, and the results are statistics from 10 GE calculations. Only two examples per setup are given due to space.

ASCADf	Min	Max	Mean	Median	Std dev
Best MLP-HW OPOI	543	833	635.2	608.5	79.93
	462	596	497.1	484.5	36.77
Best MLP-HW RPOI (200)	8	10	8.6	8.5	0.66
	8	9	8.9	9	0.3

Due to the way the **GE and other SCA metrics** are calculated, there is variability in the results coming from that randomness, so we also report on this in Table 8. We use the best model for the described setting on the ASCADf dataset and train it once, while calculating GE 10 times. The results show variation in $T_{GE=1}$ from these calculations. The variation comes from the random selection of traces for the GE calculation. In the case of OPOI, the variation was commonly more significant than in the RPOI case.

Next, we train the best model 10 times to examine the variance in results due to **the training process**. Again, we use the ASCADf dataset and test MLP training for OPOI, RPOI, and NOPOI using the HW leakage model. The results in Table 9 show the $T_{GE=1}$ statistics, such as the average minimum $T_{GE=1}$ with the lowest achieved noted in parentheses. Similarly, we report average maximum values for $T_{GE=1}$ with the highest achieved overall, and the mean and standard deviation of $T_{GE=1}$. In the case of OPOI, we obtained a better result (a minimum of 432 traces), while for RPOI, we could not reach the same result of only 5 traces for $GE = 1$. We note that the Best MLP-HW in the NOPOI case with 10 000 POIs, from 10 trainings, had 1/10 training where GE did not reach 1 (the attack is not successful). In the same scenario, with CNN-HW, 5/10 training ends in unsuccessful attacks, so we report those results in Table 9 as they showcase how even training the same model (same hyperparameters and seed) can lead to significant variation, where 50% of the training ended in unsuccessful attacks. **These experiments demonstrate the inherent variability of DLSCA results due to random search in tuning, training, and even GE calculation. Therefore, statistical results must be reported.**

Table 9: Variation from different trainings of a model with the same hyperparameters.

ASCADf	Reported T_{GE1}	Reproduced Avg min T_{GE1}	Reproduced Avg max T_{GE1}	Reproduced Mean T_{GE1}	Reproduced Std dev T_{GE1}
MLP-HW OPOI	480	545.7 (min: 432)	722 (max: 897)	619.83	49.72
MLP-HW RPOI	5	8 (min: 8)	9.9 (max: 11)	8.85	0.62
CNN-HW NOPOI	7	1523.3 (min: 9)	1528.5 (max: 3000)	1525.68	1.59

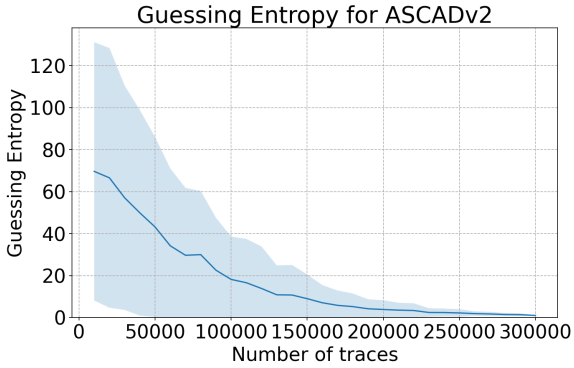


Figure 3: Reproduced GE results of the provided trained model from [110].

Case Study 3 We selected the "Scoop: An Optimization Algorithm for Profiling Attacks against Higher-Order Masking" paper [110] as it is a recent paper proposing an optimizer for DLSCA. The highlight result of this paper is that it is the first to successfully retrieve a subkey from the ASCADv2 dataset in a DLSCA black-box setting.[11]

As a trained model is provided with attack scripts, reproducing the main GE plot is straightforward. We average GE over 40 runs, although we note that the full attack set is used which means there is no more variance at 300 000 traces. We show results for the GE in Figure 3. The GE plot closely follows the reported results (see [110, Figure 6.3.1]). We include standard deviations for the individual key ranks but there are no significant discrepancies with the provided results. **These results show that direct reproducibility is significantly simplified by providing trained models.** This is especially relevant as training an individual model takes significant time and compute resources for this case.

[11] After contacting the authors to receive clarification on minor hyperparameter inconsistencies in the code, we were informed that the attacks were not valid due to an inconsistency in the dataset which authors will release a note on; this does not impact the reproducibility case study directly.