Blind-DLPA: An End-to-End Framework for Profiled Blind Side-Channel Analysis

Pei Cao, Chi Zhang, and Dawu Gu, *Member, IEEE*

***Abstract*—Conventional profiled side-channel analysis generally assumes that the plaintexts or ciphertexts associated with target traces are available during key recovery. We consider a profiled blind setting in which the attacker can obtain labeled traces from a profiling device using known inputs and keys, but observes only leakage traces from the target device. Under this setting, we propose Blind-DLPA, an end-to-end multi-task framework for profiled blind side-channel analysis (profiled BSCA) that processes complete preprocessed trace windows and jointly estimates the soft Hamming-weight posteriors of an input variable and a key-dependent intermediate variable. These posteriors are combined with pre-computed theoretical joint distributions to rank the key hypotheses, avoiding both manual point of interest (PoI) selection and a parameter-sensitive Gaussian label-error model. Evaluations on unprotected AES and Kyber demonstrate successful key recovery, while experiments on protected AES implementations show robustness to desynchronization and Boolean masking. To the best of our knowledge, Blind-DLPA provides the first reported profiled blind key recovery on the ASCAD dataset, reaching a guessing entropy of zero with 39 attack traces. Complementing these results, we revisit the Ascon Hamming-weight joint-distribution model adopted in prior work and identify previously overlooked collisions between key-dependent theoretical distributions that prevent unique key recovery under this model.**

***Index Terms*—Profiled Blind Side-Channel Analysis, Deep Learning, Multi-Task Learning, Masking Countermeasures, Point of Interest.**

I. Introduction

SIDE-CHANNEL analysis (SCA) poses a severe physical threat to modern cryptographic implementations. These attacks typically fall into two categories: *non-profiled* and *profiled* attacks. While non-profiled methods like Differential Power Analysis (DPA) [1] exploit statistical dependencies on the fly, profiled attacks are considered one of the most powerful SCAs from the information theory point of view [2]. By leveraging a controlled device to extensively characterize the physical leakage beforehand, profiled attacks can extract secret keys from a target device using only a minimal number of traces. The advent of Deep Learning-based SCA (DL-SCA) has further amplified this capability [3]. Deep neural networks can automatically learn complex, non-linear mappings between power consumption and processed data, supporting attacks on implementations protected by countermeasures [4]–[10].

Pei Cao, Chi Zhang, and Dawu Gu are with the School of Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China and the Shanghai Pudong Research Institute of Cryptology (SPRIC), Shanghai, China (e-mail: loccs_cp@sjtu.edu.cn; zcsjtu@sjtu.edu.cn; dwgu@sjtu.edu.cn).

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 62602411 and 62502311, and in part by the Startup Fund for Young Faculty at SJTU (SFYF at SJTU).

However, a limitation of most traditional and deep learning-based SCA techniques is their strict reliance on known plaintexts or ciphertexts associated with the target traces during key recovery [11]. In numerous realistic deployment scenarios, this assumption simply does not hold. For example, far-field EM leakage can be collected in remote acquisition settings without close physical access [12], [13], whereas the adversary may lack an interface to observe or control the corresponding plaintexts and ciphertexts. The same limitation arises in widely deployed encryption operating modes, such as XTS and Offset CodeBook (OCB), which inherently restrict attacker access to the block cipher's inputs and outputs. A similar constraint is observed in the key derivation process of the EMV payment protocol [14], which exposes only a fraction of the plaintext while rendering the ciphertext completely inaccessible. This reality has driven the need for Blind Side-Channel Analysis (BSCA) [15]–[19], where adversaries must recover the secret key relying solely on the observed physical leakages without knowing the plaintexts or ciphertexts associated with the target traces.

Existing BSCA approaches can be divided into two settings according to the availability of supervised profiling data. In the non-profiled setting, empirical leakage information is inferred directly from the target measurements without a labeled profiling dataset [15]–[18]. In the profiled setting, labeled measurements from a controlled profiling device are used to train leakage models, while the inputs and outputs associated with the attack traces remain unknown. Ravi et al. [20] adopt this setting for supervised machine-learning-based key recovery. Clavier and Reynaud [16] span both settings by presenting non-profiled Variance Analysis (VA) labeling together with a profiled linear-regression (LR) variant. We adopt profiled blind side-channel analysis (profiled BSCA) in this work. *The term blind therefore describes the attack phase and does not imply that profiling labels are unavailable.*

Although non-profiled and profiled BSCA grant different adversarial resources, existing implementations of both attack models exhibit several related practical limitations. First, existing BSCA methods typically rely on explicit point of interest (PoI) selection. Non-profiled attacks assume that the samples associated with each target variable can be localized, which is difficult even for aligned, unprotected implementations when the target inputs are unavailable [15]. Profiling facilitates this localization, yet the profiled blind attack of Ravi et al. [20] still relies on correlation-based PoI selection. Fixed PoIs are sensitive to trace misalignment, while masking can suppress first-order correlations with the unmasked target variables and make the relevant PoIs difficult to identify [16]. Second, several likelihood-based BSCA methods model Hamming-weight

estimation errors using a parameterized Gaussian distribution [16], [18], [20]. The resulting key scores can be sensitive to the chosen variance and to deviations from the assumed error model. Finally, masking countermeasures present a severe obstacle. Extensions of BSCA to masked implementations may require additional leakage variables [17] and stronger assumptions about how the protected intermediates can be observed, thereby increasing both statistical and computational complexity.

To address these limitations, we propose Blind-DLPA, a framework for profiled BSCA based on multi-task learning. During the attack phase, no plaintext, ciphertext, or ground-truth Hamming-weight label is used. A shared feature extractor and two task-specific heads estimate the soft Hamming-weight posteriors of an input variable and a key-dependent intermediate from complete preprocessed trace windows, avoiding explicit PoI selection. These posteriors are combined with pre-computed theoretical joint distributions, one per key candidate, to accumulate maximum-likelihood scores for key ranking, without a manually tuned Gaussian model for Hamming-weight estimation errors. For masked implementations, the network is trained to predict the unmasked variables without using the mask values as supervision, so that neither the masking scheme nor the temporal positions of the masked shares need to be known.

The main contributions of this paper are summarized as follows:

- **End-to-end multi-task learning for profiled BSCA.** We cast profiled BSCA as a multi-task learning problem in which a shared feature extractor and task-specific classification heads jointly estimate the Hamming-weight posteriors required by the blind joint-distribution distinguisher. The network processes complete preprocessed trace windows and therefore avoids a separate leakage-dependent PoI-selection and empirical-labeling stage.
- **Neural-driven maximum-likelihood criterion.** We combine the two neural posterior distributions with candidate-specific theoretical joint distributions. This scoring procedure preserves the maximum-likelihood principle while avoiding a manually tuned Gaussian model for Hamming-weight estimation errors.
- **Blind key recovery on desynchronized and masked AES.** We evaluate the framework on a desynchronized AES implementation and the masked ASCAD dataset. To the best of our knowledge, the ASCAD experiment provides the first reported profiled blind key recovery on this dataset and reaches a guessing entropy of zero with 39 attack traces without manual PoI selection or mask-value supervision.
- **Identifiability limitation of the Ascon Hamming-weight joint-distribution model.** By revisiting the Hamming-weight joint-distribution model adopted in prior Ascon work [18], we identify previously overlooked candidate-key collisions in which distinct hypotheses induce identical theoretical distributions. This result establishes that no distinguisher relying solely on this model can uniquely recover the key, irrespective of how accurately the empirical distribution is estimated.

Code Availability: The source code for reproducing the experiments in this paper, including data-processing scripts, model implementations, training configurations, and key-recovery procedures, is publicly available at https://github.com/loccs-cp/Blind-DLPA.

II. Background

A. Profiled Side-Channel Analysis

Side-channel analysis exploits physical leakages such as power consumption [21] or electromagnetic emissions [22] to recover secret information from cryptographic devices. Profiled side-channel analysis is considered one of the most powerful SCAs, operating under a two-phase adversary model. In the profiling phase, the attacker assumes full control over an open device to manipulate the inputs and cryptographic keys, thereby building an accurate leakage model. In the subsequent attack phase, the adversary captures leakage traces from the target device and uses the pre-computed model to infer the unknown secret key.

Let X denote a leakage trace and $y = f(m, k)$ denote an intermediate cryptographic state, which serves as the target variable. During profiling, the attacker models the conditional probability distribution $Pr(X \mid y)$. During the attack phase, given a set of N statistically independent attack traces $\mathbf{X} = \{X_1, X_2, \ldots, X_N\}$ and their corresponding known inputs $\mathbf{P} = \{m_1, m_2, \ldots, m_N\}$, the attacker evaluates the likelihood of each key hypothesis $k \in \mathcal{K}$. By adopting the maximum-likelihood principle, the optimal key guess $\hat{k}$ is determined as:

$$\hat{k} = \arg\max_{k \in \mathcal{K}} \sum_{i=1}^{N} \log \big(Pr(X_i \mid f(m_i, k)) \big). \tag{1}$$

The advent of deep learning has significantly empowered this process. Neural networks can approximate the complex probability distribution without requiring prior statistical assumptions and directly output the posterior probabilities to recover the key.

Despite their undeniable success, traditional profiled attacks and state-of-the-art deep learning methodologies [4]–[10] are fundamentally anchored by a strict assumption. *The attacker must have precise knowledge of the plaintext or ciphertext during the attack phase to compute the hypothetical intermediate state.* When neither is available, this conventional scoring rule cannot be applied directly.

B. Blind Side-Channel Analysis

Unlike traditional methods that rely on known plaintexts or ciphertexts, the BSCA approaches considered here rank key hypotheses using the joint Hamming-weight statistics of multiple intermediate variables. Let m be an unknown input byte and $y = f(m, k)$ be the subsequent non-linear intermediate state depending on the key byte k. This family of attacks commonly follows a three-stage pipeline, which is discussed below. We illustrate the attack flow of the representative joint-distribution-based BSCA framework in Figure 1.

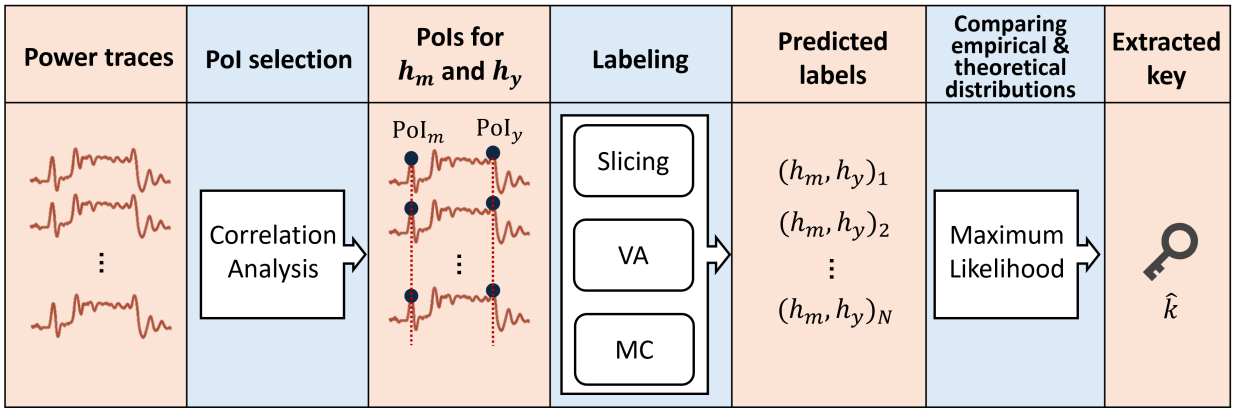


Fig. 1. Representative joint-distribution-based BSCA framework.

1) Phase 1: Pre-computing Theoretical Joint Distributions: The first stage involves constructing a theoretical model that captures the deterministic relationship between the target variables for every possible key guess. Although the actual plaintext m is unknown during the attack, an adversary can iterate through all possible values of m (from 0 to 255 for an 8-bit variable) and compute the corresponding y under a specific key hypothesis k. Assuming the unknown inputs are uniformly distributed, the attacker calculates the theoretical joint probability $Pr_{model}(HW(m), HW(y) \mid k)$ by counting the occurrences of each Hamming weight pair and dividing by the total number of possible inputs. This produces a candidate-specific theoretical probability matrix for each key hypothesis. Unique key recovery under this model requires the candidate-specific distributions to be distinguishable.

2) Phase 2: Obtaining Empirical Distributions from Traces: To utilize the theoretical matrices, the attacker must extract corresponding empirical distributions from the physical traces. This stage bridges the gap between raw power measurements and discrete Hamming-weight values. It fundamentally consists of two steps: selecting PoIs and labeling the traces.

a) PoI Selection: The first step is to identify the temporal samples where m and y leak. Existing methods range from a single PoI [15], [16] to multiple selected samples [18], [20]. In non-profiled BSCA, these samples must be localized without supervised profiling labels[1]. In profiled BSCA, labeled profiling measurements can facilitate PoI selection, as in Ravi et al. [20].

b) Labeling Traces: Once the PoIs are located, the physical amplitudes at these points must be mapped to estimated Hamming-weight labels. In the non-profiled setting, representative techniques include:

- **Slicing Labeling** [15]: The leakage amplitudes at a selected PoI are sorted and partitioned into discrete Hamming-weight classes according to the expected binomial class proportions under uniformly distributed inputs.
- **Variance Analysis (VA)** [16]: VA assumes an affine relationship between the leakage amplitude and the Hamming weight, with additive Gaussian noise. Signal and noise statistics are then used to convert the amplitude at a selected PoI into a continuous Hamming-weight estimate.
- **Multi-point Clustering (MC) Labeling** [18]: A Gaussian Mixture Model is applied to features from multiple selected samples, and the resulting clusters are associated with Hamming-weight classes.

In profiled BSCA, by contrast, the inputs and keys of the profiling traces are known. The labels $HW(m)$ and $HW(f(m,k))$ can therefore be computed directly and used to train a supervised estimator. The trained estimator is subsequently applied to target traces, whose inputs and outputs remain unavailable.

Although their sources of supervision differ, the representative prior non-profiled and profiled blind pipelines discussed above all rely on pre-selected PoIs. This dependence can reduce robustness when traces are temporally misaligned. Moreover, when masking suppresses first-order leakage of an unmasked target variable, estimators based on localized first-order leakage may require additional leakage variables or higher-order modeling.

3) Phase 3: Comparing Empirical and Theoretical Distributions: The final phase of the blind attack pipeline evaluates the likelihood of each key hypothesis by comparing the empirically obtained labels with the theoretical joint distributions. Various statistical metrics can be employed to perform this comparison, including the χ^2 distance, inner product, and harmonic mean [15]. However, previous comprehensive evaluations have demonstrated that the maximum-likelihood criterion consistently yields superior results. Consequently, modern blind side-channel frameworks predominantly adopt the maximum-likelihood criterion approach [18], [20].

To formalize this process, let (h_m^*, h_y^*) denote the true, unobservable Hamming-weight tuple of the target variables m and y. The empirical labels recovered through the techniques described in the previous section are denoted as (h_m, h_y). In recent likelihood-based BSCA methods [16], [18], [20], these recovered labels are assumed to be the true Hamming weights corrupted by additive Gaussian noise, expressed as $h_m = h_m^* + \epsilon_m$ and $h_y = h_y^* + \epsilon_y$, where $\epsilon_m \sim \mathcal{N}(0, \sigma_m^2)$ and $\epsilon_y \sim \mathcal{N}(0, \sigma_y^2)$.

The probability of a specific key guess k given a single

[1]It may not be an easy task, since the statistical T-test or other tests that partition the trace set based on a plaintext-dependent value cannot be used.

noisy observation can be derived using Bayes' theorem:

$$Pr\big(k \mid (h_m, h_y)\big) = \frac{Pr\big((h_m, h_y) \mid k\big) \cdot Pr(k)}{Pr\big((h_m, h_y)\big)}. \quad (2)$$

In this equation, the denominator $Pr((h_m, h_y))$ acts as a normalization term that is strictly independent of the key. Because the objective of the attack is to rank the key hypotheses rather than compute their absolute probabilities, this denominator can be safely ignored. Assuming the prior probability of the key $Pr(k)$ is uniformly distributed across the key space, the posterior probability becomes directly proportional to the likelihood:

$$Pr\big(k \mid (h_m, h_y)\big) \propto Pr\big((h_m, h_y) \mid k\big). \quad (3)$$

To compute the likelihood term $Pr((h_m, h_y) \mid k)$ when the true intermediate values are unknown, the law of total probability is applied. This marginalizes the probability over all possible true Hamming weight combinations:

$$\begin{aligned} Pr\big((h_m, h_y) \mid k\big) = &\sum_{h_m^*, h_y^*} Pr\big((h_m, h_y) \mid (h_m^*, h_y^*)\big) \\ &\times Pr\big((h_m^*, h_y^*) \mid k\big). \end{aligned} \quad (4)$$

The second term in this summation, $Pr((h_m^*, h_y^*) \mid k)$, corresponds precisely to the theoretical joint distribution precomputed in the first phase. The first term represents the empirical probability of the noise. Under the established Gaussian assumption, the probability of observing the specific estimation errors ϵ_m and ϵ_y is explicitly expanded as the product of two Gaussian probability density functions:

$$\begin{aligned} &Pr\big((h_m, h_y) \mid (h_m^*, h_y^*)\big) = \\ &\left(\frac{1}{\sigma_m\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{h_m - h_m^*}{\sigma_m}\right)^2\right]\right) \\ &\times \left(\frac{1}{\sigma_y\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{h_y - h_y^*}{\sigma_y}\right)^2\right]\right). \end{aligned} \quad (5)$$

In a practical attack scenario, the adversary processes a set of N independent trace measurements. The joint probability of the key given all N observations is simply the product of the individual probabilities. By applying a logarithmic transformation to prevent numerical underflow, the overall log-likelihood score for a key candidate is obtained by summing the logarithms of the marginalized probabilities:

$$\begin{aligned} \hat{k} = \arg\max_{k \in \mathcal{K}} \sum_{i=1}^{N} \log \Bigg(&\sum_{h_m^*, h_y^*} Pr\big((h_m, h_y)_i \mid (h_m^*, h_y^*)\big) \\ &\times Pr\big((h_m^*, h_y^*) \mid k\big)\Bigg), \end{aligned} \quad (6)$$

where $(h_m, h_y)_i$ denotes the empirical label extracted from the i-th trace.

However, this traditional formulation has a notable limitation: *it assumes the empirical errors follow a continuous, symmetric Gaussian distribution. Because true Hamming weights are inherently discrete and bounded (e.g., from 0 to 8), applying an unbounded continuous model may not fully capture the true error distribution. This discrepancy renders the maximum-likelihood criterion highly sensitive to the manually chosen variance parameters (σ_m^2, σ_y^2), potentially leading to suboptimal key recovery.*

Attack Metrics. The log-likelihood values are utilized as distinguishing scores. The key candidates are sorted in descending order based on their scores and organized into a guessing vector $\mathbf{G} = (G_0, \ldots, G_{|\mathcal{K}|-1})$, where $\mathcal{K}$ represents the key space. The candidate at index G_0 is the most likely secret key. The index of the correct secret key within this sorted vector is defined as the key rank.

To statistically evaluate the attack performance, the average key rank is computed over multiple independent attack experiments, yielding a standard metric known as Guessing Entropy (GE). A strictly successful and complete blind attack demands the unique and absolute identification of the secret key, achieving a GE of 0. Consequently, the primary metric of efficiency in our evaluation is $\mathrm{NT_{GE}}$ (Number of Traces to Guessing Entropy zero), which represents the minimum number of attack traces required to persistently rank the correct secret key as the absolute best candidate.

III. Deep Learning-based Profiled BSCA

A. Threat Model

We evaluate our methodology under the widely accepted two-phase profiled side-channel threat model, specifically adapted for a blind exploitation scenario. Table I contrasts this setting with conventional profiled SCA and non-profiled BSCA. Specifically, profiled SCA and profiled BSCA both permit a supervised profiling dataset. They differ during key recovery because profiled SCA uses the input or output associated with each target trace, whereas profiled BSCA does not. Non-profiled BSCA likewise withholds the target inputs and outputs but does not provide a supervised profiling dataset. Therefore, in profiled BSCA, the adversary's capabilities and restrictions are distinctly defined across a *profiling phase* and an *attack phase*.

During the *profiling phase*, we assume the adversary possesses an open device that executes the same cryptographic implementation as the target. For each profiling trace, the known values m and k allow the intermediate $y = f(m, k)$ to be computed, yielding the supervised labels $HW(m)$ and $HW(y)$. For protected implementations, the labels refer to the unmasked variables, while the mask values are never required as supervision. We note that the acquired trace window must contain sufficient exploitable leakage from both the masked intermediate values and their corresponding masks to support inference of the unmasked targets.

During the subsequent *attack phase*, the adversary shifts focus to the target device. At this stage, the adversary operates in a blind setting. They can only monitor the physical leakages emitted by the device during its cryptographic operations and

TABLE I
COMPARISON OF DATA-AVAILABILITY ASSUMPTIONS ACROSS SIDE-CHANNEL ATTACK MODELS.

Assumption	Profiled SCA	Non-profiled SCA	Non-profiled BSCA	Profiled BSCA
Supervised profiling dataset	✓	×	×	✓
Inputs/Outputs known during profiling	✓	*N/A*	*N/A*	✓
Inputs/Outputs known during attack	✓	✓	×	×

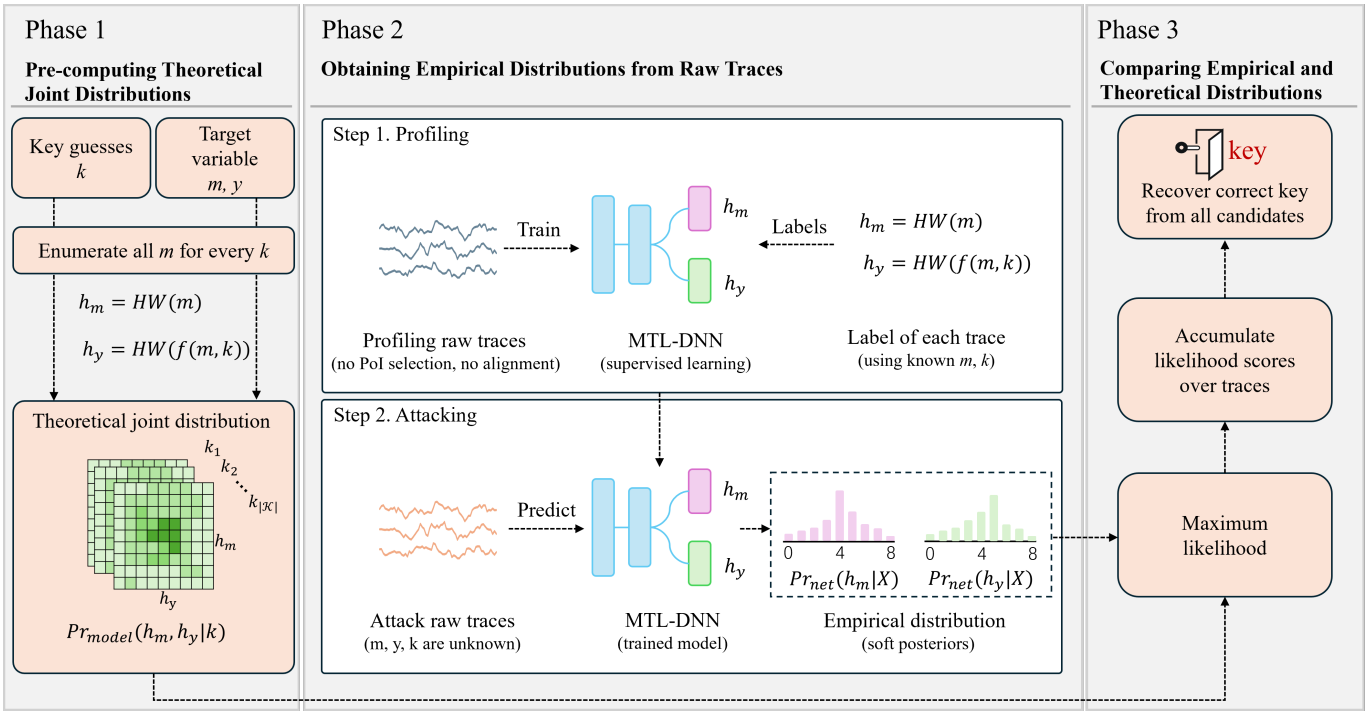


Fig. 2. Overview of the Blind-DLPA attack pipeline.

have absolutely no access to the processed data, meaning both plaintexts and ciphertexts remain unknown. The adversary must infer the secret key relying solely on the captured traces.

Note that key recovery also requires the candidate-specific joint distributions to be distinguishable. Candidates that induce identical distributions receive identical likelihood scores regardless of posterior-estimation accuracy, as examined for Ascon in Section IV-D.

B. Methodology

As established in Section II, the traditional BSCA pipeline faces several practical and theoretical limitations. First, isolating exact PoIs from raw, high-dimensional traces without known data is fundamentally challenging. Second, conventional empirical labeling techniques rely on establishing a direct mapping from localized physical amplitudes to intermediate states. This mapping inherently struggles with complex non-linear leakages and is severely bottlenecked by masking countermeasures[2]. Finally, the traditional maximum-likelihood criterion relies on the assumption that estimation errors follow a continuous Gaussian distribution. This assumption introduces structural mismatches for discrete Hamming weights, which consequently renders the attack highly sensitive to the manually chosen variance parameters.

To address these limitations, we propose an end-to-end profiled framework, Blind-DLPA, that integrates the feature extraction capabilities of deep learning with the joint-distribution exploitation of blind side-channel analysis (as shown in Figure 2). Rather than relying on manual PoI extraction and rigid statistical assumptions, Blind-DLPA leverages multi-task deep neural networks as direct, distribution-free posterior probability estimators. By processing complete preprocessed trace windows to output the categorical probabilities required for key recovery, Blind-DLPA structurally bypasses the necessity for manual PoI selection, empirical amplitude labeling, and parameter-sensitive Gaussian error assumptions, consolidating the traditionally fragmented pipeline into a unified architecture.

1) Attack Pipeline Overview: Figure 2 summarizes the Blind-DLPA workflow. In Phase 1, the attacker enumerates the possible inputs for every key hypothesis and pre-computes the corresponding theoretical joint Hamming-weight distribution. For instance, consider the first-round AES relation $y = \mathrm{Sbox}(m \oplus k)$ for one key byte. Blind-DLPA enumerates

[2]While some prior studies [16] have attempted to analytically adapt these models to masked scenarios, the necessary integration over the uniform mask space severely flattens the theoretical distributions, thereby drastically degrading inter-key distinguishability.

all $m \in \{0, \ldots, 255\}$ for each of the 256 candidate values of k and constructs $Pr_{model}(h_m, h_y \mid k)$, where $h_m = HW(m)$ and $h_y = HW(y)$. Phase 2 contains two steps. During profiling, the known (m, k) values provide two Hamming-weight labels (h_m, h_y) for each profiling trace window, and a multi-task network is trained on preprocessed trace windows from the profiling device using these known intermediate Hamming-weight labels. During the blind attack, the trained network directly converts each preprocessed target trace window into two soft Hamming-weight distributions $Pr_{net}(h_m \mid X_i)$ and $Pr_{net}(h_y \mid X_i)$, without explicit PoI selection or trace alignment. In Phase 3, these soft outputs are compared with the theoretical distribution of every key candidate. The trace-wise log-likelihood scores are accumulated, and the candidate with the largest cumulative score is returned as the recovered key. We next detail the key methodological aspects of Blind-DLPA.

2) Multi-Task Learning for End-to-End Blind-DLPA: A trace window may contain leakage associated with both the input m and the subsequent intermediate y at different temporal locations. Temporal shifts complicate fixed-sample localization, while masking changes the first-order relation between the measured leakage and the unmasked variables. We therefore estimate the two targets jointly from the complete preprocessed trace window using multi-task learning.

As illustrated in Figure 2, our framework employs a multi-task deep neural network (MTL-DNN) architecture to jointly optimize the extraction of leakages related to both the plaintext m and the target intermediate state y. Let $X \in \mathbb{R}^T$ represent a preprocessed input trace window with T time samples. The MTL-DNN is logically divided into *shared layers* and *task-specific layers*. The shared layers, which can be instantiated as either a multi-layer perceptron (MLP) or a convolutional neural network (CNN), directly process the entire trace sequence. This shared representation mitigates the risk of overfitting [23] and enables the network to implicitly capture the temporal correlations between different cryptographic operations. The extracted latent features are subsequently passed to two independent task-specific layers (classification heads). These independent heads output the categorical posterior probabilities, denoted as $Pr_{net}(h_m \mid X)$ and $Pr_{net}(h_y \mid X)$, which correspond to the Hamming weights of m and y, respectively.

When the candidate-specific theoretical distributions are distinguishable, the two posterior outputs support key recovery because the key information lies in the joint relation between h_m and h_y rather than in either Hamming weight alone. For a trace X_i, the product $Pr_{net}(h_m \mid X_i)Pr_{net}(h_y \mid X_i)$ provides soft evidence for every possible pair (h_m, h_y). For each candidate k, this pairwise evidence is combined with the pre-computed matrix $Pr_{model}(h_m, h_y \mid k)$ and summed over all pairs. Accumulating the trace-wise scores ranks candidates according to how consistently their theoretical matrices agree with the network outputs, as formalized in Equation 11. The network therefore estimates leakage-derived evidence rather than the key itself, while the theoretical matrix introduces the key-dependent relation required for ranking.

This end-to-end architecture integrates leakage localization and posterior estimation within the profiled blind inference pipeline. The deep hidden layers of the neural network autonomously localize the relevant leakage patterns across the time domain, eliminating the prerequisite for manual PoI selection [24]. Furthermore, when the shared feature extractor is implemented using convolutional layers, the inherent shift invariance provided by the receptive fields effectively absorbs time-domain desynchronization [4], [5], bypassing the strict requirement for precise trace alignment prior to the attack.

3) Implicit High-Order Profiling for Masked Targets: The introduction of Boolean masking disrupts traditional BSCA methodologies. In a typical first-order masked implementation, a sensitive variable y is split into a masked state $y' = y \oplus u$ and a random mask u, which leak at distinct temporal locations. Because the random mask u renders the Hamming weight of y' statistically independent of the unmasked y, the empirical distribution extracted from the physical traces, which essentially models $(HW(m'), HW(y'))$, fundamentally mismatches the pre-computed unmasked theoretical models $(HW(m), HW(y))$. This inherent mismatch renders the standard maximum-likelihood criterion ineffective even for the correct key.

To circumvent this distribution mismatch, previous literature has attempted to adapt blind joint distribution analysis to masked implementations. Notably, Clavier and Reynaud [16] proposed an *analytical masked-joint approach*. Assuming that both the input m and the intermediate state y are protected by the exact same mask u, they modified the theoretical models to reflect the joint distribution of the masked couples $(m', y') = (m \oplus u, y \oplus u)$. By recalculating the theoretical probabilities averaged over all possible values of the uniformly distributed input m and mask u, the theoretical models are realigned with the masked empirical distributions. However, this analytical adaptation has three practical limitations:

- **Masking Scheme Assumption:** The standard analytical masked-joint approach requires both the input variable m and the target intermediate state y to *be protected by the exact same mask* u. If independent masks are applied, the standard bivariate joint distribution becomes ineffective[3].
- **PoI Localization Requirement:** Traditional empirical modeling requires explicitly locating specific PoIs for (m', y'). Finding these temporal locations without mask-specific labels can be difficult.
- **Loss of Distinguishability:** Integrating the probabilities over the entire mask space severely flattens the theoretical models. Consequently, the conditional distributions $\Pr((HW(m'), HW(y')) \mid k)$ for different key candidates become highly similar. As visually demonstrated by the contrast between Figure 3 and Figure 4 using the AES S-box as an illustrative example, the distinct probability peaks of the unmasked distributions are severely flattened in the masked domain. This reduction in inter-key distinguishability necessitates a substantially larger number of attack traces to converge.

[3]Later work by Clavier et al. [17] introduced a quadrivariate joint distribution to handle independent masks for AES. However, this theoretical extension assumes masks are reused across rounds or across bytes within a single round, and requires millions of traces to converge even in simulated environments.

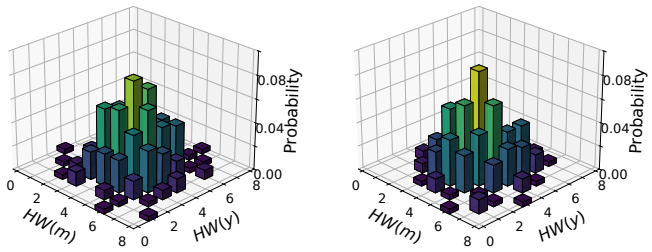


Fig. 3. Theoretical joint distributions of $(HW(m), HW(y))$ for the AES S-box, with $k = 0$xAB (left) and $k = 0$xCD (right).

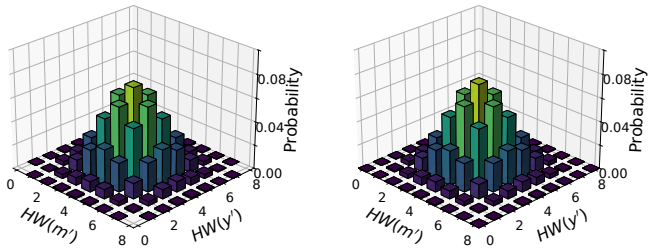


Fig. 4. Theoretical joint distributions of the masked couples $(HW(m'), HW(y'))$ for the AES S-box, with $k = 0$xAB (left) and for $k = 0$xCD (right). The analytical integration over the mask uniformly flattens the distributions, making the key candidates nearly indistinguishable.

Our proposed multi-task learning framework overcomes these barriers through *implicit high-order combination*. Blind-DLPA trains the MTL-DNN on masked traces using the unmasked labels $HW(m)$ and $HW(y)$ computed from the known profiling inputs and keys. When the trace window contains sufficient leakage from the relevant shares, the shared feature extractor can learn higher-order feature combinations that are predictive of the unmasked targets. The resulting posteriors are evaluated against the original unmasked theoretical distributions. This construction uses neither mask values as supervision nor explicitly selected mask-related PoIs (as illustrated in Figure 5).

4) Neural-Driven Maximum-Likelihood Criterion: The final component of our framework reformulates the maximum-likelihood criterion process. As established in Section II, traditional blind attacks evaluate likelihoods by extracting empirical estimations and applying a Gaussian probability density function to model the observation likelihood $Pr\big((h_m, h_y) \mid (h_m^*, h_y^*)\big)$. This generative modeling approach requires carefully tuning variance parameters. We bypass this limitation by leveraging the neural network as a direct posterior probability estimator.

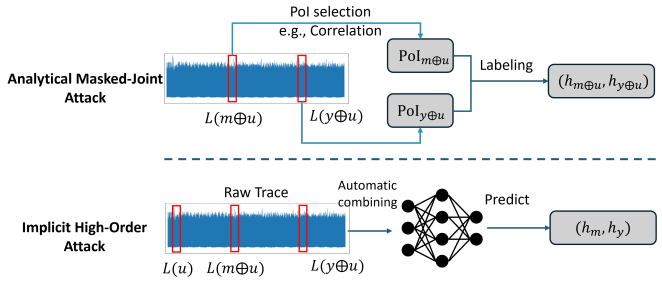


Fig. 5. Comparison between analytical masked-joint attack and our implicit high-order attack.

In a blind setting, the ultimate goal of the maximum-likelihood criterion is to find a candidate key k that maximizes the likelihood of observing the target traces. For a single trace X, the attacker seeks to evaluate $Pr(X \mid k)$. By applying the law of total probability, the likelihood function is expanded as:

$$Pr(X \mid k) = \sum_{h_m, h_y} Pr\big(X \mid (h_m, h_y)\big) \times Pr\big((h_m, h_y) \mid k\big) \tag{7}$$

This formulation reveals a mismatch between the standard maximum-likelihood criterion and deep learning outputs. The maximum-likelihood criterion necessitates a generative probability $Pr\big(X \mid (h_m, h_y)\big)$, which represents the probability of observing a specific trace given a target Hamming weight state. In contrast, our multi-task neural network operates as a discriminative model. Given an input trace X, the network outputs the posterior probabilities of the state given the trace.

To rigorously convert the network's posterior output $Pr\big((h_m, h_y) \mid X\big)$ into the required generative likelihood $Pr\big(X \mid (h_m, h_y)\big)$, we apply Bayes' theorem:

$$Pr\big(X \mid (h_m, h_y)\big) = \frac{Pr\big((h_m, h_y) \mid X\big) Pr(X)}{Pr(h_m, h_y)} \tag{8}$$

Substituting this Bayesian transformation back into Equation 7 yields:

$$Pr(X \mid k) = \sum_{h_m, h_y} \left[\frac{Pr\big((h_m, h_y) \mid X\big) Pr(X)}{Pr(h_m, h_y)} \times Pr\big((h_m, h_y) \mid k\big) \right] \tag{9}$$

By isolating the key-independent term $Pr(X)$, the equation simplifies to:

$$Pr(X \mid k) = Pr(X) \sum_{h_m, h_y} \left[\frac{Pr\big((h_m, h_y) \mid X\big)}{Pr(h_m, h_y)} \times Pr\big((h_m, h_y) \mid k\big) \right] \tag{10}$$

In the context of key recovery, $Pr(X)$ represents the absolute probability of observing the specific trace in nature. Since this term acts as a constant scaling factor identical for all key candidates, it can be safely discarded during the log-likelihood maximization process.

To operationalize this neural-driven scoring function, we explicitly define its components. The term $Pr\big((h_m, h_y) \mid k\big)$ is exactly the theoretical joint distribution pre-computed

TABLE II
BENCHMARK ROLES IN THE EXPERIMENTAL EVALUATION.

Benchmark	Joint model	Evaluation focus
Unprotected implementations		
AES	Bivariate, 8-bit	PoI-free key-recovery
Kyber	Bivariate, 16-bit	Larger key space
Ascon	Trivariate, 8-bit	Distribution collisions
Protected AES implementations		
Desynchronized AES	Bivariate, 8-bit	Against hiding
Masked AES	Bivariate, 8-bit	Against masking

during the initial phase, denoted as $Pr_{model}(h_m, h_y \mid k)$. The joint prior $Pr(h_m, h_y)$ represents the unconditional baseline probability of the state occurring globally, denoted as $Pr_{prior}(h_m, h_y)$. Finally, assuming conditional independence of the target variables given the highly informative leakage trace X, the joint posterior $Pr\big((h_m, h_y) \mid X\big)$ is computed as the product of the marginal outputs from the two independent classification heads of our network, denoted as $Pr_{net}(h_m \mid X) \cdot Pr_{net}(h_y \mid X)$.

By aggregating the log-likelihoods over N independent attack traces, the final distinguishing score for key recovery is formulated as:

$$\hat{k} = \arg\max_{k \in \mathcal{K}} \sum_{i=1}^{N} \log \left(\sum_{h_m, h_y} \frac{Pr_{net}(h_m \mid X_i) Pr_{net}(h_y \mid X_i)}{Pr_{prior}(h_m, h_y)} \times Pr_{model}(h_m, h_y \mid k) \right). \tag{11}$$

The formulation generalizes to additional intermediate variables by adding one task-specific head per variable, as in the trivariate Ascon analysis (see Section IV-D). Because the backbone is shared, the additional network cost grows approximately with the number and size of the added heads. In contrast, storage and evaluation of the joint distribution scale with the product of the class cardinalities.

IV. EXPERIMENTAL SETUP AND RESULTS

Evaluation roadmap. Table II summarizes the role of each benchmark. The evaluation examines both the algorithmic applicability of Blind-DLPA and its robustness to implementation countermeasures. Unprotected AES provides a reference setting, Kyber tests whether the same pipeline extends to a different cryptographic algorithm and a larger key space, and Ascon examines whether the adopted Hamming-weight joint-distribution model uniquely distinguishes the key candidates. Desynchronized and masked AES change the leakage conditions to evaluate robustness to hiding and masking.

All benchmarks retain the same three-phase pipeline and neural-driven key-recovery principle. Section IV-A presents the common setup, followed by the benchmark-specific instantiations in Sections IV-B–IV-F.

A. Common Neural and Computational Setup

All experiments instantiate the MTL-DNN with either an MLP or a CNN shared feature extractor. We use the same multi-task objective and candidate-key-scoring principle across datasets. Only the output cardinality and number of task heads, together with the architecture selected from the common search space, are adapted to the target model and leakage condition.

The MTL-DNN uses a shared feature extractor followed by one classification head for each variable in the target joint model. The bivariate AES and Kyber experiments use $\mathcal{V} = \{m, y\}$, whereas the Ascon boundary analysis uses $\mathcal{V} = \{m_0, m_1, y\}$. Each head performs a categorical Hamming-weight prediction, with 9 classes for an 8-bit variable and 17 classes for a 16-bit Kyber coefficient. The network is optimized by summing the Categorical Cross-Entropy (CCE) losses over all task heads:

$$\mathcal{L}_{total} = \sum_{v \in \mathcal{V}} \mathcal{L}_{CCE}(\hat{h}_v, h_v). \tag{12}$$

To guarantee a rigorous evaluation, we systematically define our hyperparameter search space based on established bounds in SCA literature [4], [5], [24], [25]. For the MTL-CNN, we vary the convolutional blocks (from 2 to 6) and filter sizes (from 4 to 128). For the MTL-MLP, we vary the depth (from 3 to 8 layers) and width (from 128 to 1024 nodes). Models are trained using the Adam optimizer with learning rates logarithmically sampled between 10^{-5} and 10^{-2}. To mitigate overfitting on noisy and desynchronized traces, we evaluate configurations with regularization techniques, including dropout and batch normalization. We average all GE curves over 100 independent attacks, each drawing a random subset from the disjoint attack set.

Computational platform. All experiments were conducted on a workstation equipped with a single NVIDIA GeForce RTX 4060 Laptop GPU with 8 GB of GPU memory and 32 GB of system RAM. As a representative runtime reference, under the AES configuration with 7,500 training traces, 500 validation traces, 5,000 samples per trace, and a batch size of 256, training the MTL-MLP for 80 epochs required 21 seconds in total. Training the MTL-CNN for 80 epochs required 65 seconds. These wall-clock measurements include both training and validation but exclude data preprocessing, hyperparameter search, and key-recovery evaluation.

B. Evaluation on Unprotected AES

*1) **Attack Point**:* We target the non-linear Substitution Box (S-box) operation in the first AES-128 encryption round. The two variables are the plaintext byte m and the corresponding S-box output $y = \mathrm{Sbox}(m \oplus k)$, where k denotes the target subkey-byte hypothesis.

*2) **Dataset**:* We utilize the publicly available ChipWhisperer dataset, a standard benchmark widely adopted in side-channel literature [26], [27]. The measurements are acquired from an unprotected AES-128 software implementation executing on an 8-bit Atmel XMEGA microcontroller, which is mounted on a ChipWhisperer CW308 UFO target board [28].

The dataset comprises a total of 10,000 power consumption traces captured with a fixed secret key, where each trace encompasses 5,000 time samples. We use 8,000 traces for profiling and 2,000 traces for the attack. As for the classic non-profiled BSCA, we use all 10,000 traces to label and attack.

TABLE III
KEY-RECOVERY RESULTS ON UNPROTECTED AES.

Methodology	# PoI	Result
Non-profiled BSCA		
Slicing Labeling [15]	1	GE = 3.26
VA Labeling [16]	1	GE = 7.12
MC Labeling + DNN [18]	50	GE = 9.37
Profiled BSCA		
Linear Regression (LR) [16]	1	GE = 6.32
Random Forest [20]	50	NT_{GE} = 983
This Work (MTL-MLP)	None	NT_{GE} = 123
This Work (MTL-CNN)	None	**NT_{GE} = 72**

Note: Methods are grouped into non-profiled BSCA and profiled BSCA to reflect their different assumptions about access to labeled profiling data. Results from the two groups are reported for context rather than as equal-resource comparisons. GE denotes the guessing entropy after processing all attack traces. NT_{GE} represents the minimum number of traces required to reach GE = 0. "None" indicates that no explicit PoI selection is performed.

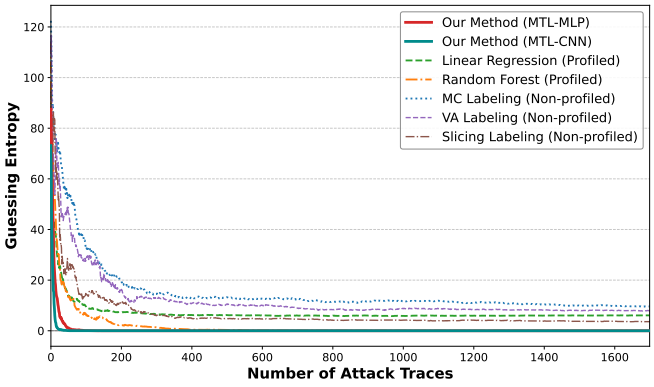


Fig. 6. Guessing entropy for the unprotected AES dataset.

*3) **Experimental Results**:* First, we evaluate the classical and recent state-of-the-art baselines on the unprotected AES dataset. As summarized in Table III and illustrated in Figure 6, the non-profiled blind attacks, specifically Slicing [15], VA labeling [16], and the recent MC labeling combined with a DNN [18], struggle significantly, failing to fully recover the secret key within 1,700 traces. Similarly, among the profiled baselines, the LR approach [16] stalls at a high GE plateau. While the Random Forest baseline manages to achieve full key recovery, it converges slowly, requiring 983 traces.

Next, we deploy our proposed multi-task learning framework using both MLP and CNN architectures. Operating end-to-end on all 5,000 samples of each input trace window without prior feature engineering, Blind-DLPA reaches GE=0 with fewer attack traces in this experiment. The MTL-CNN architecture achieves the best overall performance, driving the NT_{GE} down to just 72.

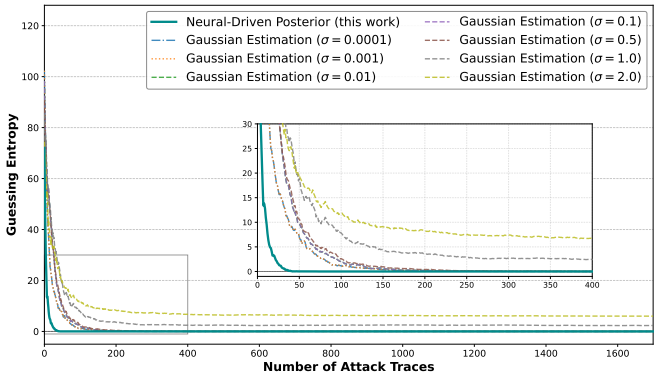


Fig. 7. Comparison between the proposed neural-driven maximum-likelihood criterion and Gaussian empirical estimation proposed by [16].

Ablation Study on Likelihood Estimation. To validate the effectiveness of our neural-driven maximum-likelihood criterion, we conduct an ablation study isolating the scoring function. Using the same trained MTL-CNN backbone, we compare two distinct likelihood evaluation strategies. To replicate the traditional generative approach, we collapse the network's continuous probability outputs into discrete point estimates (i.e., extracting a single Hamming weight via an $\arg\max$ operation) to serve as the empirical observations, and evaluate their likelihood using the classical Gaussian probability density function defined in Equation 5. We vary the standard deviation parameter σ of the Gaussian baseline across multiple magnitudes. As illustrated in Figure 7, the traditional Gaussian estimation is highly sensitive to the chosen variance, with larger variances uniformly flattening the distinguishability. More importantly, even under its optimal variance configuration, it fails to match the peak efficiency of our framework. By directly leveraging the full posterior distributions without parameter tuning, our neural-driven approach significantly outperforms all Gaussian variants.

C. Evaluation on Kyber

*1) **Attack Point**:* Kyber is selected to determine whether the same Blind-DLPA pipeline extends beyond an 8-bit S-box to post-quantum polynomial arithmetic with 16-bit intermediate representations and a substantially larger candidate space. We evaluate Kyber768, standardized as ML-KEM, with parameters $\ell = 3$, $n = 256$, and $q = 3329$ over the polynomial ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$.

During a key exchange, the critical operation for side-channel adversaries is the `Decapsulation` procedure, which internally invokes the public-key decryption primitive (CPAPKE.Dec) to decrypt the received ciphertext. The precise sequence of this decryption is detailed in Appendix A, Algorithm 1. To accelerate polynomial multiplication from quadratic to linear time, Kyber leverages the Number Theoretic Transform (NTT). Because the chosen modulus q only permits an incomplete NTT, a polynomial of 256 coefficients is transformed into 128 polynomials of degree 1. Consequently,

the polynomial multiplication, specifically the core operation $\hat{s}^T \circ \text{NTT}(u)$ highlighted in Line 4 of Algorithm 1, is computed via pair-pointwise multiplication (i.e., base multiplication) in the NTT domain.

Let $\hat{u}$ and $\hat{s}$ denote the ciphertext and secret key polynomials in the NTT domain, respectively. The base multiplication of two degree-1 polynomials, $\hat{u} = \hat{u}_0 + \hat{u}_1 X$ and $\hat{s} = \hat{s}_0 + \hat{s}_1 X$, modulo $X^2 - \zeta$ (where ζ is a root of unity), yields the resulting coefficients:

$$\begin{aligned} r_0 &= \hat{s}_0\hat{u}_0 + \hat{s}_1\hat{u}_1\zeta \pmod q \\ r_1 &= \hat{s}_0\hat{u}_1 + \hat{s}_1\hat{u}_0 \pmod q \end{aligned} \tag{13}$$

In this work, we define our generic public variable m as the ciphertext coefficient $\hat{u}_0$ (which remains unknown to the attacker during the attack phase), and our target intermediate state y as the partial product $\hat{s}_0\hat{u}_0$ (highlighted in red in Eq. (13)).

These specific points are chosen because the computation of $\hat{s}_0\hat{u}_0$ involves multiplying a fixed, unobservable secret key coefficient ($\hat{s}_0$) with a varying ciphertext component ($\hat{u}_0$). Because the theoretical joint distribution of the Hamming weights $(HW(\hat{u}_0), HW(\hat{s}_0\hat{u}_0))$ exhibits a strong, unique dependency on the exact value of $\hat{s}_0$, we can directly configure our multi-task neural network to estimate their posterior probabilities. Notably, both m and y are modeled using the Hamming weight of their 16-bit integer representations, consistent with the int16-based data representation used in the reference implementation.

*2) **Dataset**:* For the evaluation on the post-quantum KEM, we utilize the reference CRYSTALS-Kyber768 dataset recently published in [18]. The traces are acquired from a standard pqm4 [29] software implementation executing on a 32-bit ARM Cortex-M4 microcontroller (STM32F3) clocked at 7.372 MHz, mounted on a ChipWhisperer CW308 target board [28]. The power consumption measurements are captured using a high-precision LeCroy 610Zi oscilloscope at a sampling rate of 500 MS/s.

The original dataset encompasses 100,000 execution traces with a fixed secret key, where each trace initially contains 50,000 time samples. Following the standard preprocessing methodology established in the original study, the raw traces are compressed using a window resampling technique [24], yielding a dimension of 10,000 samples per trace. We adhere to the same data partitioning as [18]: $N_t = 80{,}000$ traces are allocated for the profiling phase, while the remaining $N_a = 20{,}000$ traces constitute the disjoint attack set.

*3) **Experimental Results**:* The evaluation on the Kyber dataset reveals a pronounced performance gap among existing BSCA methodologies, as detailed in Table IV and Figure 8. We observe that Slicing, VA labeling, and the DNN-assisted MC Labeling all fail to extract the secret key within the available trace limit. Within the profiled category, the LR method similarly plateaus at a high Guessing Entropy, rendering the key unrecoverable. Although the Random Forest algorithm eventually accomplishes a successful key recovery, it requires 9,735 traces and relies strictly on 50 manually selected PoIs.

Conversely, our Blind-DLPA framework recovers the key with substantially fewer traces in this challenging scenario.

TABLE IV
KEY-RECOVERY RESULTS ON KYBER.

Methodology	# PoI	Result
Non-profiled BSCA		
Slicing Labeling [15]	1	GE = 1416.22
VA Labeling [16]	1	GE = 1319.56
MC Labeling + DNN [18]	50	GE = 883.09
Profiled BSCA		
Linear Regression (LR) [16]	1	GE = 556.77
Random Forest [20]	50	NT_{GE} = 9735
This Work (MTL-MLP)	None	**NT_{GE} = 606**
This Work (MTL-CNN)	None	NT_{GE} = 695

Note: Methods are grouped into non-profiled BSCA and profiled BSCA to reflect their different assumptions about access to labeled profiling data. Results from the two groups are reported for context rather than as equal-resource comparisons. GE denotes the guessing entropy after processing all attack traces. NT_{GE} represents the minimum number of traces required to reach GE = 0. "None" indicates that no explicit PoI selection is performed.

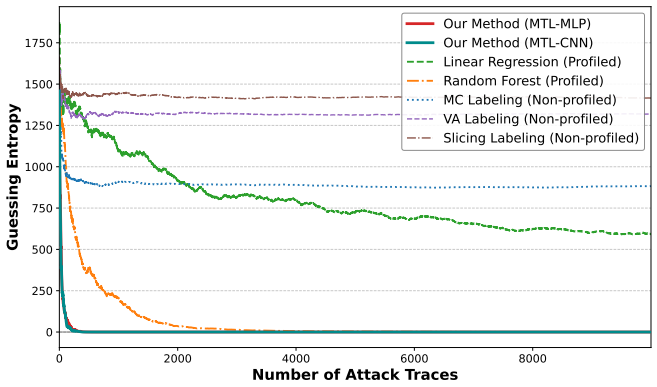


Fig. 8. Guessing entropy for the Kyber dataset.

As depicted in Figure 8, both architectural variants recover the key. Attaining the best overall performance, the MTL-MLP model pushes the required trace count (NT_{GE}) to 606. This capability translates to an approximate $16\times$ reduction in required traces compared to the baseline [20], consistent with the trend observed on AES.

D. Evaluation on Ascon

*1) **Attack Point**:* Ascon is included as a diagnostic boundary case rather than as another successful-key-recovery benchmark. Ascon is the winner of the NIST Lightweight Cryptography project, designed to provide authenticated encryption for resource-constrained environments. Ascon operates on a 320-bit state organized as five 64-bit words and updated by a sponge-based permutation.

As depicted in Figure 9, the non-linear substitution layer applies a 5-bit S-box vertically across the state words, denoted as x_0 through x_4. During the initial permutation phase, the 320-bit state is initialized such that x_0 contains the 64-bit algorithmic Initialization Vector (IV), x_1 and x_2 contain the 128-bit secret key, while x_3 and x_4 contain the 128-bit public

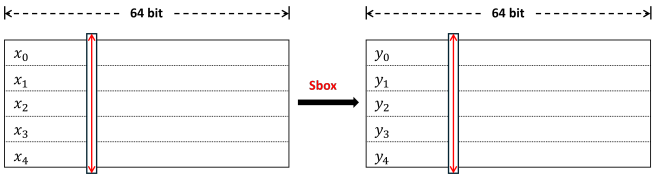


Fig. 9. Ascon 5-bit column-wise (bitsliced) S-box, illustrating the vertical non-linear transformation applied across the five 64-bit words (x_0 to x_4).

nonce. We specifically target the algebraic normal form of the y_4 S-box output because it creates a direct, non-linear algebraic dependency between the key bits and the nonce variables and has been adopted as an attack point in prior Ascon side-channel studies [18], [30], including the blind setting of [18].

Following the standard divide-and-conquer strategy established in recent BSCA literature [18], recovering the full 64-bit x_1 (the first half of the secret key) is practically executed by targeting independent 8-bit chunks. Let y denote the targeted 8-bit chunk of the y_4 output. By substituting the state words with their corresponding initial values, this 8-bit intermediate value is computed bitwise as:

$$y = m_0 \oplus m_1 \oplus (IV_0 \& k) \oplus (m_1 \& k) \oplus k, \tag{14}$$

where k is the targeted 8-bit key chunk from x_1, m_0 and m_1 represent the corresponding 8-bit chunks of the public nonce words (x_3 and x_4), and IV_0 is the corresponding 8-bit chunk of the known constant IV.

Unlike AES and Kyber, which typically involve a two-variable dependency, this specific Ascon formulation explicitly illustrates a three-variable correlation. To conduct a blind attack, prior work [18] consequently proposed modeling the 3D joint distribution $(HW(m_0), HW(m_1), HW(y))$.

*2) **Dataset**:* We use the publicly available Ascon dataset [31], which contains power traces collected from a 32-bit STM32F4 microcontroller during the first round of the initialization permutation. The dataset comprises 100,000 traces with random keys and 100,000 traces with a fixed key, with each trace containing 772 time samples. We use 100,000 random-key traces for training and 10,000 fixed-key traces as the attack set. The complete trace windows are provided to the network without manual PoI selection or trace alignment.

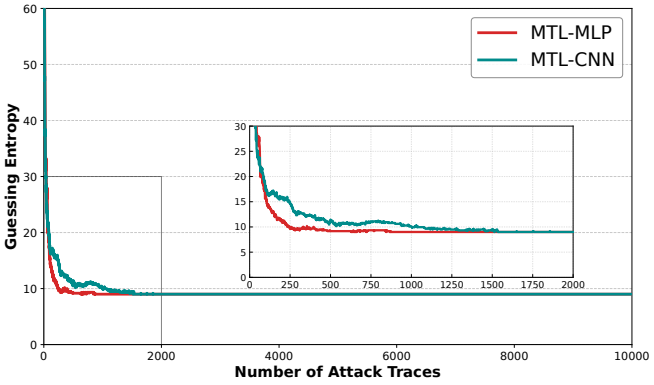


Fig. 10. Guessing entropy for the Ascon dataset.

*3) **Experimental Results**:* As shown in Figure 10, the GE decreases rapidly but then stabilizes above zero despite convergence of the three prediction tasks. The initial decrease shows that the predicted posteriors contain useful key-dependent evidence. However, a set of remaining candidates cannot be distinguished. A similar GE floor was reported in [18] and was attributed to overlapping leakage points. The following analysis identifies a stronger, estimator-independent cause. These candidates induce exactly the same theoretical Hamming-weight joint distribution and therefore receive identical likelihood scores even if the network posteriors are estimated perfectly.

*4) **Theoretical Analysis of Distribution Collisions**:* Specifically, we show that *the Ascon leakage model suffers from severe distribution collisions*. This collision phenomenon originates from the combination of bitwise independence in the Ascon S-box and the permutation invariance of the Hamming weight function. As defined in Equation 14, the computation of y is purely bitwise. The i-th bit of the output, $y^{(i)}$, depends exclusively on the i-th bits of the inputs: $m_0^{(i)}$, $m_1^{(i)}$, $IV_0^{(i)}$, and $k^{(i)}$.

During the theoretical-distribution construction phase, the nonces m_0 and m_1 iterate uniformly over all 2^{16} possible byte combinations. Suppose we evaluate two candidate keys, k and k', which differ only by a swapped pair of bits at indices i and j where the constant IV_0 shares the same value (i.e., $IV_0^{(i)} = IV_0^{(j)}$). Under these conditions, the underlying Boolean functions computing the i-th and j-th bits of y become mathematically identical. Because m_0 and m_1 are uniformly distributed, simultaneously swapping their i-th and j-th bits constitutes a perfect bijection over the input space. Crucially, the Hamming weight operator strips away spatial information. Applying this bit-swap permutation alters the physical bit positions but strictly preserves the macroscopic values of $HW(m_0)$, $HW(m_1)$, and $HW(y)$.

For instance, considering the case where $IV_0 = 0x80$ (binary 10000000_2), the constant shares the same bit-value of 0 at seven positions (indices 0 to 6). If the targeted secret key is $k = 0x85$ (binary 10000101_2), which contains two '1's among these seven positions, there exist exactly $\binom{7}{2} = 21$ keys with the same bit-count in those positions. Since the induced joint distributions are mathematically indistinguishable, any statistical distinguisher will assign these 21 keys the exact same likelihood score.

Denote the accumulated neural-driven log-likelihood by $S_N(k)$. For any two colliding candidates k and k', the equality

$$\begin{aligned} Pr_{model}(h_{m_0}, h_{m_1}, h_y \mid k) = Pr_{model}(h_{m_0}, h_{m_1}, h_y \mid k') \\ \Longrightarrow S_N(k) = S_N(k'), \end{aligned} \tag{15}$$

holds for every number of traces N and for any network outputs. Therefore, under this marginal Hamming-weight mapping, the Ascon key cannot be uniquely identified and the GE cannot converge to zero. The observed floor is thus a fundamental limitation of the joint-distribution representation, rather than of Blind-DLPA's neural feature extractor.

E. Robustness to Desynchronization

*1) **Attack Point**:* The attack point remains the AES S-box relation $y = \mathrm{Sbox}(m \oplus k)$ defined in Section IV-B. By changing only the temporal alignment of the traces, this experiment isolates the effect of desynchronization from changes in the target computation.

*2) **Dataset**:* We construct the dataset by injecting independent random shifts of up to 50 samples in either direction into the unprotected AES traces. We use $N_t = 8{,}000$ traces for profiling and average the results over 100 attacks, each using 1,700 randomly selected attack traces.

TABLE V
KEY-RECOVERY RESULTS ON DESYNCHRONIZED AES.

Methodology	# PoI	Result
Non-profiled BSCA		
Slicing Labeling [15]	1	GE = 21.41
VA Labeling [16]	1	GE = 12.23
MC Labeling + DNN [18]	50	GE = 177.52
Profiled BSCA		
Linear Regression (LR) [16]	1	GE = 40.09
Random Forest [20]	50	GE = 21.18
This Work (MTL-MLP)	None	GE = 7.61
This Work (MTL-CNN)	None	**$\mathrm{NT_{GE}}$ = 78**

Note: Methods are grouped into non-profiled BSCA and profiled BSCA to reflect their different assumptions about access to labeled profiling data. Results from the two groups are reported for context rather than as equal-resource comparisons. GE denotes the guessing entropy after processing all attack traces. $\mathrm{NT_{GE}}$ represents the minimum number of traces required to reach GE = 0. "None" indicates that no explicit PoI selection is performed.

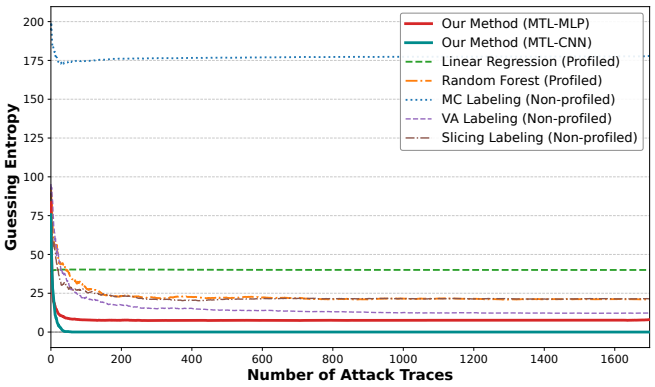


Fig. 11. Guessing entropy for the desynchronized AES dataset.

*3) **Experimental Results**:* As detailed in Table V and Figure 11, the injected temporal shifts degrade all baseline methodologies. Reliant on static PoIs, both the classical techniques and the recent DNN-assisted MC Labeling stall at unrecoverable Guessing Entropy (GE) plateaus. When deploying our framework directly on the raw, unaligned traces, the MTL-MLP architecture achieves a notably lower GE (7.61) but ultimately falls short of full key recovery due to the rigid, position-dependent nature of fully connected layers.

Conversely, the MTL-CNN emerges as the only configuration reaching GE=0 under this setting, rapidly extracting the secret key with an $\mathrm{NT_{GE}}$ of just 78. This superiority is explicitly driven by the inherent shift-invariance of convolutional networks, firmly establishing the robustness of our end-to-end framework against hiding countermeasures.

F. Robustness to Boolean Masking

*1) **Attack Point**:* The attack point remains the AES S-box relation defined in Section IV-B, while the relevant intermediate variables are protected by first-order Boolean masking. This experiment therefore evaluates whether Blind-DLPA can infer the same unmasked targets from leakage distributed across the masked values and mask shares.

*2) **Dataset**:* We use the ASCAD dataset [25] and perform no manual PoI selection or mask-aware feature extraction. A generic window-resampling operation [24] is applied only to reduce the trace dimension. The resulting dataset contains $N_t = 40{,}000$ profiling traces and 20,000 attack traces.

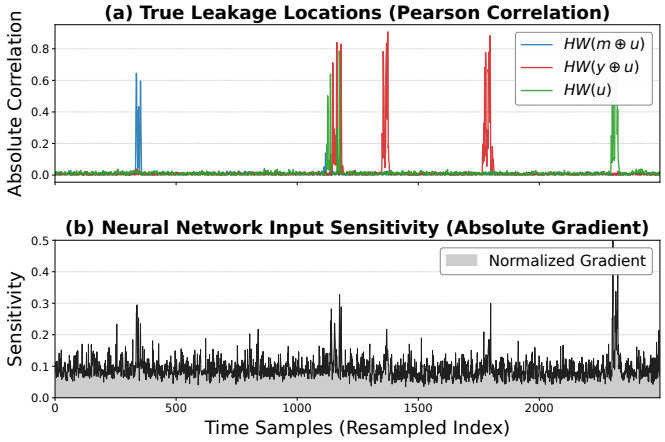


Fig. 12. Evaluation of feature selection capability of MTL-DNN against masking.

*3) **Experimental Results**:* Before evaluating key recovery, Figure 12 validates our network's feature selection via gradient-based input sensitivity. The network's sensitivity peaks align precisely with the true physical leakages (derived via CPA) of the masked components ($m \oplus u$, $y \oplus u$) and the mask (u), confirming that the MTL-DNN performs genuine high-order unmasking rather than exploiting artifactual noise.

As detailed in Table VI and Figure 13, all conventional first-order baselines fail. The random mask successfully obfuscates any direct correlation between the raw leakages and the unmasked targets.

We subsequently evaluate the Analytical Masked-Joint approach[4]. While mathematically valid for this specific dataset, as ASCAD protects both m and y with the same mask u, it requires 17,526 traces (CNN) and 18,355 traces (MLP) to succeed. The higher trace requirement is consistent with our

[4]We adapt only the theoretical modeling from [16]. Instead of their original manual PoI extraction, we train our MTL-DNN to predict the flattened masked distributions $(HW(m'), HW(y'))$ from the complete preprocessed trace windows.

TABLE VI
KEY-RECOVERY RESULTS ON MASKED AES.

Methodology	# PoI	Result
Non-Profiled BSCA		
Slicing Labeling [15]	1	GE = 63.14
VA Labeling [16]	1	GE = 54.08
MC Labeling + DNN [18]	50	GE = 41.25
Profiled BSCA		
Linear Regression (LR) [16]	1	GE = 156.79
Random Forest [20]	50	GE = 176.40
Masked-Joint (MTL-MLP)	None	NT_{GE} = 18355
Masked-Joint (MTL-CNN)	None	NT_{GE} = 17526
Implicit High-Order (MTL-MLP)	None	**NT_{GE} = 39**
Implicit High-Order (MTL-CNN)	None	NT_{GE} = 646

Note: Methods are grouped into non-profiled BSCA and profiled BSCA to reflect their different assumptions about access to labeled profiling data. Results from the two groups are reported for context rather than as equal-resource comparisons. GE denotes the guessing entropy after processing all attack traces. NT_{GE} represents the minimum number of traces required to reach GE = 0. "None" indicates that no explicit PoI selection is performed.

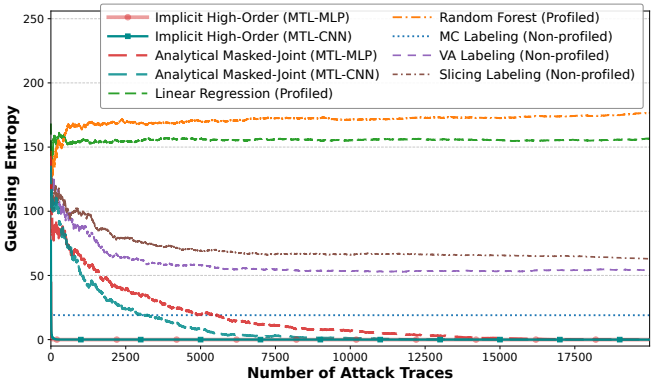


Fig. 13. Guessing entropy convergence for the masked AES (ASCAD) dataset.

earlier theoretical analysis: integrating over a uniform mask space results in lower inter-key distinguishability.

Conversely, our Implicit High-Order Profiling methodology largely avoids this degradation. The MTL-CNN effectively recovers the key in 646 traces, while the MTL-MLP achieves NT_{GE} of just 39. This is reasonable since the aligned ASCAD traces retain stable temporal locations for the relevant shares, enabling the position-sensitive MTL-MLP to directly combine leakage components for high-order inference. To the best of our knowledge, this study represents the first reported profiled blind key recovery on the ASCAD dataset, while maintaining a required number of attack traces comparable to state-of-the-art profiled methodologies [6], [24], [32] that utilize known plaintext information during the attack phase.

Cross-benchmark summary. Table VII consolidates the key-recovery outcomes reported above. The methods are grouped by attack model so that profiled and non-profiled results are not interpreted as equal-resource comparisons. Table VII distinguishes an empirical failure to reach $GE = 0$ within the available trace limit ($\times$) from the structural identifiability limit observed for Ascon (†). In the latter case, colliding key candidates induce identical theoretical distributions and cannot be distinguished using only that representation.

V. RELATED WORK

Blind side-channel analysis based on joint distributions was introduced by Linge et al. [15], who estimated Hamming weights by partitioning the amplitudes at a single point of interest, and extended by Clavier and Reynaud [16], who proposed Variance Analysis for continuous estimates and also examined a profiled linear-regression variant. Their extensions to masked implementations adapt the theoretical model to protected variables. The bivariate construction considers variables protected by the same mask [16], whereas the quadrivariate construction treats separate masks at the cost of a higher-dimensional joint model [17].

Machine learning has subsequently been incorporated under both blind attack models. Ravi et al. [20] considered profiled BSCA and trained Random Forest estimators from labeled profiling traces while the inputs and outputs associated with target traces remained unavailable. Rezaeezade et al. [18] retained a non-profiled setting and used multi-point clustering to estimate Hamming-weight labels from target measurements. Although both approaches use multiple selected features rather than a single leakage sample, their supervision assumptions differ and both retain an explicit feature-selection stage. Blind-DLPA adopts the data-availability model of Ravi et al., and differs in how the empirical evidence is obtained, estimating soft Hamming-weight posteriors from complete preprocessed trace windows rather than hard labels from selected samples.

Prior profiled SCA studies applied deep learning and ensembles to improve key recovery on Ascon [30], [31]. In the blind setting, Rezaeezade et al. [18] adopted a three-variable Hamming-weight joint-distribution model and reported a nonzero GE floor, which they associated with overlapping leakage points. Our analysis revisits this model and establishes that distinct key candidates can induce identical theoretical distributions, revealing an identifiability limit that is independent of leakage estimation and neural-network performance.

Multi-task and multi-head learning have also been explored in profiled SCA. Maghrebi [33] introduced multi-label profiling to learn leakage targets associated with multiple AES key bytes or intermediate operations jointly. Masure and Strullu [34] used a multi-task network to target several intermediates in a masked AES implementation. In a related PUF setting, MLMSA [35] used separate heads for the response and side-channel information. Blind-DLPA builds on this established architecture but assigns its task-specific heads a different role in profiled blind inference, namely estimating soft Hamming-weight posteriors of an unknown target input and a key-dependent intermediate for blind key ranking.

VI. CONCLUSION

In this work, we presented Blind-DLPA, an end-to-end framework for profiled BSCA. Under the considered threat

TABLE VII
OVERALL KEY-RECOVERY OUTCOMES UNDER THE EVALUATED CONFIGURATIONS.

Methodology	**# PoI**	**Unprotected AES**	**Kyber**	**Ascon**	**Desync AES**	**Masked AES**
Non-profiled BSCA						
Linge et al. [15]	1	×	×	–	×	×
Clavier & Reynaud (VA) [16]	1	×	×	–	×	×
Rezaeezade et al. [18]	50	×	×	†	×	×
Profiled BSCA						
Clavier & Reynaud (LR) [16]	1	×	×	–	×	×
Ravi et al. [20]	50	✓	✓	–	×	×
This Work	**None**	✓	✓	†	✓	✓

Note: Methods are grouped into non-profiled BSCA and profiled BSCA to reflect their different assumptions about access to labeled profiling data. Results from the two groups are reported for context rather than as equal-resource comparisons. *Symbols:* ✓: $GE = 0$ is reached; ×: $GE = 0$ is not reached within the available trace limit; –: not evaluated; †: unique recovery is prevented by collisions in the theoretical Hamming-weight joint distribution.

model, the attacker has access to labeled measurements with known inputs and keys during profiling, while the plaintexts and ciphertexts associated with the target traces remain unavailable during key recovery. Blind-DLPA jointly estimates the soft Hamming-weight posteriors of an input variable and a key-dependent intermediate from complete preprocessed trace windows, and combines these posteriors with candidate-specific theoretical joint distributions for key ranking. This design avoids explicit point of interest selection and a manually parameterized Gaussian model for Hamming-weight estimation errors.

Experiments on unprotected AES and Kyber demonstrate successful key recovery under the evaluated profiled blind setting. The experiments on desynchronized AES and masked AES further show that the learned feature extractor can accommodate temporal shifts and combine leakage information distributed across masked values and masks, provided that the acquired trace window contains sufficient exploitable leakage from the relevant components. On the ASCAD dataset, Blind-DLPA reaches a guessing entropy of zero with 39 attack traces without using the plaintexts, ciphertexts, or mask labels associated with the attack traces.

Complementing the successful key-recovery results, we revisit the Hamming-weight joint-distribution model adopted in prior blind side-channel analyses of Ascon. We show that distinct key hypotheses can induce identical theoretical distributions and therefore cannot be uniquely distinguished, regardless of labeling accuracy, neural-network performance, or the number of attack traces. This finding reveals a previously overlooked fundamental cause of the failure to uniquely recover the Ascon key.

APPENDIX
KYBER DECRYPTION ALGORITHM

Algorithm 1 Kyber CPAPKE.Dec (Decryption)

Require: Secret key $sk \in \mathcal{B}^{12\cdot\ell\cdot n/8}$, Ciphertext $c \in \mathcal{B}^{d_u\cdot\ell\cdot n/8+d_v\cdot n/8}$
Ensure: Message $m \in \mathcal{B}^{32}$
1: $u \leftarrow \text{Decompress}_q(\text{Decode}_{d_u}(c), d_u)$
2: $v \leftarrow \text{Decompress}_q(\text{Decode}_{d_v}(c + d_u \cdot \ell \cdot n/8), d_v)$
3: $\hat{s} \leftarrow \text{Decode}_{12}(sk)$
4: $m \leftarrow \text{Encode}_1\left(\text{Compress}_q\left(v - \text{NTT}^{-1}(\hat{s}^T \circ \text{NTT}(u))\right), 1\right)$
5: **return** m

REFERENCES

[1] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual international cryptology conference*. Springer, 1999, pp. 388–397.

[2] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2002, pp. 13–28.

[3] H. Maghrebi, T. Portigliatti, and E. Prouff, "Breaking cryptographic implementations using deep learning techniques," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*. Springer, 2016, pp. 3–26.

[4] G. Zaid, L. Bossuet, A. Habrard, and A. Venelli, "Methodology for efficient CNN architectures in profiling attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 1, pp. 1–36, 2020.

[5] 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*, pp. 147–168, 2020.

[6] 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*, pp. 235–274, 2021.

[7] P. Cao, C. Zhang, X. Lu, D. Gu, and S. Xu, "Improving deep learning based second-order side-channel analysis with bilinear CNN," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3863–3876, 2022.

[8] L. Wu, G. Perin, and S. Picek, "Weakly profiling side-channel analysis," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2024, no. 3, pp. 707–730, 2024.

[9] P. Cao, C. Zhang, X. Lu, and D. Gu, "Cross-device profiled side-channel attack with unsupervised domain adaptation," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 27–56, 2021.

[10] 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*, vol. 2025, no. 2, pp. 333–361, 2025.

[11] A.-T. Hoang, N. Hanley, and M. O'Neill, "Plaintext: A missing feature for enhancing the power of deep learning in side-channel analysis?" *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020.

[12] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming channels: When electromagnetic side channels meet radio transceivers," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 163–177.

[13] H. Wang, D. He, T. Deng, and J. Wang, "Feature reconstruction: Far field EM side-channel attacks in complex environment," *IEEE Transactions on Information Forensics and Security*, vol. 20, pp. 10 066–10 081, 2025.

[14] L. EMVCo, "EMV integrated circuit card specifications for payment systems book 2 security and key management version 4.3," 2011.
[15] Y. Linge, C. Dumas, and S. Lambert-Lacroix, "Using the joint distributions of a cryptographic function in side channel analysis," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2014, pp. 199–213.
[16] C. Clavier and L. Reynaud, "Improved blind side-channel analysis by exploitation of joint distributions of leakages," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 24–44.
[17] C. Clavier, L. Reynaud, and A. Wurcker, "Quadrivariate improved blind side-channel analysis on boolean masked AES," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2018, pp. 153–167.
[18] A. Rezaeezade, T. Yap, D. Jap, S. Bhasin, and S. Picek, "Breaking the blindfold: Deep learning-based blind side-channel analysis," in *34th USENIX Security Symposium (USENIX Security 25)*, 2025, pp. 5777–5796.
[19] N. Hanley, M. Tunstall, and W. P. Marnane, "Unknown plaintext template attacks," in *International Workshop on Information Security Applications*. Springer, 2009, pp. 148–162.
[20] P. Ravi, D. Jap, S. Bhasin, and A. Chattopadhyay, "Machine learning based blind side-channel attacks on PQC-based KEMs-a case study of Kyber KEM," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 01–07.
[21] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2004, pp. 16–29.
[22] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The EM side—channel (s)," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2002, pp. 29–45.
[23] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.
[24] 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*, pp. 828–861, 2022.
[25] R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas, "Deep learning for side-channel analysis and introduction to ASCAD database," *J. Cryptogr. Eng.*, vol. 10, no. 2, pp. 163–188, 2020. [Online]. Available: https://doi.org/10.1007/s13389-019-00220-8
[26] L. Wu, L. Weissbart, M. Krček, H. Li, G. Perin, L. Batina, and S. Picek, "Label correlation in deep learning-based side-channel analysis," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3849–3861, 2023.
[27] 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 *International Conference on Cryptology in India*. Springer, 2024, pp. 3–28.
[28] C. O'Flynn and Z. Chen, "ChipWhisperer: An open-source platform for hardware embedded security research," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2014, pp. 243–260.
[29] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "PQM4: Post-quantum crypto library for the ARM cortex-M4," 2019.
[30] A. Rezaeezade, A. Basurto-Becerra, L. Weissbart, and G. Perin, "One for all, all for Ascon: Ensemble-based deep learning side-channel analysis," in *International Conference on Applied Cryptography and Network Security*. Springer, 2024, pp. 139–157.
[31] L. Weissbart and S. Picek, "Lightweight but not easy: Side-channel analysis of the Ascon authenticated cipher on a 32-bit microcontroller," *Cryptology ePrint Archive*, 2023.
[32] J. Yu, Y. Wang, S. Qu, Y. Zhao, Y. Shi, P. Cao, X. Lu, C. Zhang, D. Gu, and C. Hong, "End-to-end non-profiled side-channel analysis on long raw traces," in *European Symposium on Research in Computer Security*. Springer, 2025, pp. 526–544.
[33] H. Maghrebi, "Deep learning based side-channel attack: A new profiling methodology based on multi-label classification," IACR Cryptology ePrint Archive, Paper 2020/436, 2020. [Online]. Available: https://eprint.iacr.org/2020/436
[34] L. Masure and R. Strullu, "Side-channel analysis against ANSSI's protected AES implementation on ARM: End-to-end attacks with multi-task learning," *Journal of Cryptographic Engineering*, vol. 13, no. 2, pp. 129–147, 2023.
[35] Y. Gao, J. Yao, L. Pang, W. Yang, A. Fu, S. F. Al-Sarawi, and D. Abbott, "MLMSA: Multilabel multiside-channel-information enabled deep learning attacks on APUF variants," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 9, pp. 2863–2876, 2023.



Pei Cao received the B.S. degree in information security from the University of Electronic Science and Technology of China and the Ph.D. degree in computer science and technology from Shanghai Jiao Tong University. He is currently an Assistant Research Fellow at Shanghai Jiao Tong University. His current research interests include hardware security and side-channel analysis.



Chi Zhang received the B.S. degree from Southeast University and the Ph.D. degree from Shanghai Jiao Tong University. He is currently an Assistant Research Fellow with the School of Computer Science, Shanghai Jiao Tong University, Shanghai, China. His research interests include side-channel analysis and cryptographic engineering.



Dawu Gu (Member, IEEE) received the B.S. degree in applied mathematics from Xidian University, Xi'an, China, in 1992, and the M.S. and Ph.D. degrees in cryptography in 1995 and 1998, respectively. He is a Distinguished Professor with the School of Computer Science, Shanghai Jiao Tong University (SJTU), Shanghai, China. He is the Vice President of the Chinese Association for Cryptologic Research, China. He has got over 200 scientific papers in top academic journals and conferences, such as CRYPTO, IEEE SECURITY AND PRIVACY, ACM CCS, NDSS, and TCHES. His research interests include crypto algorithms, crypto engineering, and system security. Dr. Gu was the winner of Chang Jiang Scholars Distinguished Professors Program in 2014 by Ministry of Education of China. He won the National Award of Science and Technology Progress in 2017. He also served as PC Member of international conferences for more than 30 times.