








# Leakage Model-flexible Deep Learning-based Side-channel Analysis

Lichao Wu<sup>1</sup> , Azade Rezaeezade<sup>2</sup> , Amir Ali-pour<sup>3</sup> ,  
Guilherme Perin<sup>4</sup>  and Stjepan Picek<sup>5</sup> 

<sup>1</sup> Technical University of Darmstadt, Darmstadt, Germany

<sup>2</sup> Delft University of Technology, Delft, The Netherlands

<sup>3</sup> École de technologie supérieure, Montreal, Canada

<sup>4</sup> Leiden University, Leiden, The Netherlands

<sup>5</sup> Radboud University, Nijmegen, The Netherlands

**Abstract.** Profiling side-channel analysis has gained widespread acceptance in both academic and industrial realms due to its robust capacity to unveil protected secrets, even in the presence of countermeasures. To harness this capability, an adversary must access a clone of the target device to acquire profiling measurements, labeling them with leakage models. The challenge of finding an effective leakage model, especially for a protected dataset with a low signal-to-noise ratio or weak correlation between actual leakages and labels, often necessitates an intuitive engineering approach, as otherwise, the attack will not perform well.

In this paper, we introduce a deep learning approach with a flexible leakage model, referred to as the multi-bit model. Instead of trying to learn a pre-determined representation of the target intermediate data, we utilize the concept of the stochastic model to decompose the label into bits. Then, the deep learning model is used to classify each bit independently. This versatile multi-bit model can adjust to existing leakage models like the Hamming weight and Most Significant Bit while also possessing the flexibility to adapt to complex leakage scenarios. To further improve the attack efficiency, we extend the multi-bit model to profile all 16 subkey bytes simultaneously, which requires negligible computational effort. The experimental results show that the proposed methods can efficiently break all key bytes across four considered datasets while the conventional leakage models fail. Our work signifies a significant step forward in deep learning-based side-channel attacks, showcasing a high degree of flexibility and efficiency with the proposed leakage model.

**Keywords:** Side-channel Analysis · Deep learning · Multi-bit model · Multi-task learning

## 1 Introduction

Side-channel analysis (SCA) is a powerful tool for analyzing unintended leakages during secret processing. Using a divide-and-conquer strategy, SCA categorizes leakage traces to simplify the key search process. SCA in symmetric-key cryptography is commonly divided into non-profiling and profiling attacks. Profiling attacks assume that the adversary has complete control over a clone of the targeted device. With this clone, the adversary can profile the side-channel behavior of the targeted device in advance and then use this knowledge to extract secret information from the targeted device. Conversely, non-profiling attacks assume that the adversary cannot access a duplicate of the targeted device. As a result, the adversary must consider all measurements containing secret information

---

E-mail: [stjepan.picek@ru.nl](mailto:stjepan.picek@ru.nl) (Stjepan Picek)



processing from the targeted device and then use statistical analysis to make correct guesses of secret data.

Research in recent years has led to a remarkable leap forward in the efficacy of profiling attacks, largely attributed to the growing role of deep learning (commonly denoting the deep learning-based SCA as DLSCA). For instance, a dataset like the one introduced in [BPS<sup>+</sup>20] that required significant effort to break a few years ago can now be broken with single trace attacks [PWP22]. The effectiveness of deep learning in SCA stems from its flexibility in identifying and characterizing leakages in the side-channel traces (originating from the switching activities of transistors within the integrated circuit) and correlating these leakages with variations in the data being processed. To make this happen, one must label the collected traces based on an assumption about the underlying physical leakage. Therefore, a function denoted as the *leakage model* is used for labeling the traces. The leakage model should be selected wisely to reflect the feature of actual leakages, thus strengthening the links between the underlying leakage and the processed data. A more common method is to adopt pre-defined leakage models for all bits (such as the Hamming weight or Identity) or specific bits (for instance, the most or least significant bit) [WPP22, ZBHV19, Tim19, PWP22]. However, the existing leakage models may not match the actual leakage, considering the diversity of actual implementations, measurement setups, and leakage pre-processing techniques. Moreover, they impose a degree of pre-existing conditions on bit significance, which may inadvertently reduce the flexibility of the learning process. Stochastic models [SLP05] could be a good candidate for adapting to the physical leakages, but they should incorporate dimension reduction techniques to reduce the computation complexity; besides, when facing a more challenging dataset or profiling dataset number is insufficient, their performance could be mediocre. We provide more details in Section 4.1.

In this work, we propose a flexible leakage model, denoted as *multi-bit model*, that circumvents the constraints of traditional pre-defined leakage models, allowing for dynamic adjustment of the leakage model in real-time during the profiling phase. Our extensive analysis underscores the potential of the multi-bit model to evolve into an optimal leakage model that fits the physical leakages, facilitating outstanding attack performance and effective leakage assessment. This work also makes a significant step in simplifying the conventional profiling SCA process. Typically, SCA requires attacking each secret byte separately to reveal the entire secret, such as in the case of AES, where 16 separate attacks are needed. We streamline this process into a single model training session. This is achieved through our *multi-byte multi-bit model*, which simultaneously targets all subkeys. This model stands out for its capability to branch from a primary framework into multiple sub-branches, enabling concurrent attacks on various subkeys. To our knowledge, this is the first time in the field where all 16 secret key bytes can be attacked simultaneously, marking a major improvement from the state-of-the-art methods that focus on individual byte attacks in computation efficiency and attack performance. Together with the multi-bit model that is free from the constraints of pre-defined models and assumptions, our proposed method is a highly promising and robust solution in DLSCA.

Our main contributions are:

1. We analyze the limitations of the existing profiling attacks, then propose the multi-bit model that allows more flexibility and gives insight into leakage assessment.
2. We propose a new attack method, multi-byte multi-bit DLSCA, that can simultaneously profile all subkeys (in our case, 16 AES subkeys).
3. We perform case studies to showcase our analysis, which is further validated on four publicly available datasets and an advanced hardware AES implementation [SC23] from CHES CTF 2023.

4. We provide a hyperparameter study on several relevant factors for the proposed method: data augmentation, batch size, and the number of training epochs. The results confirm the robustness of our method to diverse settings, and data augmentation is a crucial factor in mounting powerful multi-bit model-based attacks. Finally, the multi-bit model provides a novel leakage assessment method.

The source code is available in <https://github.com/lichao-wu9/MMB>.

The remainder of this paper is structured as follows. We provide the necessary background information in Section 2. Following that, Section 3 discusses related works. In Section 4, we outline the limitations of the current methods and describe the proposed method in detail. Section 5 provides experimental results using five different datasets. The discussion on the proposed method is given in Section 6. Finally, we summarize our findings and discuss potential avenues for future research in Section 7.

## 2 Preliminaries

### 2.1 Notation

We use calligraphic letters such as  $\mathcal{X}$  to represent sets. The corresponding upper-case letters denote random variables ( $X$ ) and random vectors ( $\mathbf{X}$ ) over  $\mathcal{X}$ . The realizations of  $X$  and  $\mathbf{X}$  are represented by lower-case letters ( $x, \mathbf{x}$ ), respectively.

We use  $\mathbf{T}$  to represent a side-channel dataset, a collection of side-channel measurements. Each measurement (side-channel trace)  $\mathbf{t}_i$  is associated with an input value (either plaintext or ciphertext) denoted by  $\mathbf{d}_i$  and a key represented as  $\mathbf{k}_i$ , or  $k_{i,j}$  and  $d_{i,j}$  when partial key recovery on byte  $j$  is under consideration. A key candidate is denoted by  $k$ , with its value drawn from the key space  $\mathcal{K}$ . The correct key is denoted by  $k^*$ . Every trace  $\mathbf{t}_i$  contains multiple features called samples or points of interest. The dataset is partitioned into a training (or profiling) set of size  $N$  and an attack (or test) set of size  $Q$ . In the context of deep learning techniques, we write  $\boldsymbol{\theta}$  to denote the vector of parameters learned in a profiling model (for example, the weights and biases in neural networks).

### 2.2 Threat Model

We assume an adversary has an open and cloned device with a target cipher, and he/she can control the key and plaintext. The adversary could send commands to perform encryption or decryption operations. To launch attacks, the adversary first measures multiple leakage traces from the cloned device and builds a profiling model by mapping the leakage traces and labels of intermediate data that is constructed based on known keys and a priori assumption of the leakage model. Next, he/she inputs the leakage traces measured from the attack device (unknown key) to the profiling model to recover the secret information. This paper evaluates five datasets; besides the SMAesH dataset [SC23], we assume that an attacker does not have knowledge of mask shares.

### 2.3 Profiling Side-channel Analysis

The two most prevalent types are a Gaussian template for the template attack (TA) and machine learning (deep neural network) models for deep learning-based SCA. The TA employs Bayes' theorem to make predictions, dealing with multivariate probability distributions, since the leakage over consecutive time samples is not independent [CRR03]. This attack operates under the assumption that the traces depend on the  $F$  features given the target class. Thus, the posterior probability for each class value  $y$  can be computed as

follows:

$$p(Y = y|X = x) = \frac{p(Y = y)p(X = x|Y = y)}{p(X = x)}, \quad (1)$$

where  $X$  denotes continuous measurements (i.e.,  $\mathbf{t}_i$ ) and  $Y$  denotes discrete class variables. The discrete output's number of classes  $c$  depends on the leakage model and the cryptographic algorithm. As such, the discrete probability  $p(Y = y)$  equals its sample frequency, whereas  $p(X = x|Y = y)$  reflects a density function. In practice,  $p(X = x|Y = y)$  is generally assumed to follow a (multivariate) normal distribution and is parameterized by the mean  $\bar{x}_y$  and covariance matrix  $\Sigma_y$ :

$$p(X = x|Y = y) = \frac{1}{\sqrt{(2\pi)^F |\Sigma_y|}} e^{-\frac{1}{2}(x-\bar{x}_y)^T \Sigma_y^{-1} (x-\bar{x}_y)}. \quad (2)$$

The stochastic attack utilizes linear regression instead of probability density estimation [SLP05]. One critical aspect of the stochastic attack is the choice of regressors (i.e., base functions) [HKSS12]. A natural choice in the SCA context is the bitwise selection of the intermediate variable.

For DLSCA, deep learning is used to model  $\mathbf{p}(Y = y|X = x)$ . Based on the data and labeling, such algorithms train a model  $\mathbf{f}_\theta$  to predict labels on previously unseen data. Most of the supervised learning methods follow the Empirical Risk Minimization (ERM) framework, where the model parameters  $\theta$  are obtained by solving the optimization problem:

$$\arg \min_{\theta} \frac{1}{N} \sum_i^N \mathbf{L}(\mathbf{f}_\theta(\mathbf{t}_i), y_i), \quad (3)$$

where  $\mathbf{L}$  is the loss function. The trained model  $\mathbf{f}_\theta$  is then used to predict classes  $y$  (more precisely, the probabilities that a certain class would be predicted) based on the previously unseen set of traces  $\mathbf{x}$  of size  $Q$ . It is worth noting that while the template attack requires the assumption of a particular distribution of the traces, DLSCA does not require such assumptions, giving it more flexibility to handle complex leakages. However, DLSCA may require more computational resources and profiling traces, which might limit its applicability in some scenarios (while, in practice, this should not be a problem due to GPU support for deep learning algorithms).

## 2.4 Evaluating the Attack Performance

Upon completing a profiling attack, the result is presented as a two-dimensional matrix with dimensions equivalent to  $Q \times c$ . A common practice is to use the maximum log-likelihood distinguisher to create a cumulative sum  $S(k)$  for each key candidate  $k$ :

$$S(k) = \sum_{i=1}^Q \log(\mathbf{p}_{i,y}). \quad (4)$$

Here,  $\mathbf{p}_{i,y}$  denotes the probability vector given a key  $k$  and input  $d_i$ , the resulting class is  $y \in \mathcal{Y}$ . The class  $y$  is derived from the key and input through a cryptographic function and a leakage model (e.g., the Hamming weight of the Sbox output).

The output of an attack is a key guessing vector  $\mathbf{g} = [g_1, g_2, \dots, g_{|\mathcal{K}|}]$ , which is computed for  $Q$  traces in the attack phase. This vector arranges the key candidates in descending order of probability, with  $g_1$  being the most likely candidate and  $g_{|\mathcal{K}|}$  being the least likely.

Guessing entropy (GE), the average position of  $k^*$  within the key guessing vector  $\mathbf{g}$ , is usually employed to estimate the effort required to uncover the secret key  $k^*$  [SMY09]. In this paper, if an attack method reaches the key rank of zero (meaning that the correct key ranks first), we calculate the required number of attack traces for this key rank to provide us a precise estimation of the attack performance, denoted as  $T_{GE0}$ .

### 3 Related Work

The side-channel analysis domain has been immersed in the study of profiling attacks for over two decades. Chari et al. pioneered this area by introducing the concept of the template attack (TA), which is considered the most powerful approach from an information-theoretic perspective [CRR03]. Schindler et al. proposed stochastic models, also known as linear regression-based profiling attacks, where the authors approximated the real leakage function within a suitable vector subspace [SLP05]. To address the issue of insufficient measurements per class for TA, pooling all covariance matrices into a single one is a viable solution [JW02]. Choudhary and Kuhn, for example, investigated the pooled template attack and achieved performance improvements in secret recovery and computational cost [CK13]. A feature engineering phase is often necessary to reduce the number of points of interest, thus avoiding the need to profile with complicated templates. This can be achieved using machine learning-based feature selection [PHJB19], dimensionality reduction methods such as Principal Component Analysis (PCA) [WEG87, APSQ06, BHvW12], Linear Discriminant Analysis (LDA) [SA08], or Sum of Squared Pairwise T-differences (SOST) [GLRP06], or a combination of these techniques [CDSU23].

The landscape of profiling attacks has evolved significantly by incorporating machine learning (ML) techniques. Initial ML methodologies incorporated techniques such as random forest [LMBM13], support vector machines [HGM<sup>+</sup>11] and naive Bayes [PHG17]. The performance of these techniques often outperformed (or at least matched) that of the template attack and stochastic models, laying the groundwork for the advent of more complex ML approaches. The focus of the SCA community began to pivot toward deep learning in 2016, following the seminal work of Maghrebi et al. [MPP16]. Incorporating deep learning alleviated some challenges related to countermeasures and feature engineering, yet it also introduced difficulties associated with tuning deep learning algorithms. Despite this, early research by Cagli et al. [CDP17] and Kim et al. [KPH<sup>+</sup>19] highlighted the potential of convolutional neural networks (CNNs) in breaking protected targets. Techniques for improving attack performance using regularization were also explored [KPH<sup>+</sup>19, RB24, HK18]. Subsequent works [ZBHV19, WAGP20, PWP22] delved deeper into the design methodologies for CNNs, achieving unprecedented attack performance on datasets secured by masking and hiding countermeasures. *Unfortunately, most of studies discussed so far employ pre-defined leakage models to label leakage traces. These may not necessarily align well with the target dataset.* For example, Perin et al. failed to use the Identity leakage model to break the CHES\_CTF dataset [PWP22]. Wu et al. pointed out that the different fixed keys in the training and validation sets for the CHES\_CTF dataset result in the inefficiency of the Identity leakage model [WWK<sup>+</sup>23], then incorporated label distribution into the profiling phase to address this limitation. Besides, the leakage modeling was explored with the stochastic model [SLP05]. Stochastic models assume that the leakage function can be formed as the sum of a deterministic component and a random one. During the profiling phase, these two components of the leakage function are approximated independently. Building on this work, Zaid et al. developed a conditional variational autoencoder methodology for stochastic attacks [ZBC<sup>+</sup>23]. This approach mitigates the black-box aspect of deep learning and facilitates a more straightforward process for architectural design. Recently, Zhang et al. introduced a multi-label deep learning-based SCA that treats each bit in a byte as a separate label [ZXF<sup>+</sup>20]. Then use all these labels separately as a single-bit leakage model to decide about a byte of intermediate value. Their main contribution is leveraging an ensemble using multiple labels. However, the reasoning behind the application and the method of usage is different from ours. Finally, the performance improvement is insignificant (detailed in Section 6).

So far, the predominant focus of the research has been treating the optimization of the labeling function (leakage model) as a task distinct from training the profiling model. This leaves a research gap as the unification of the profiling model and labeling function has not

been properly addressed. Moreover, the performance enhancements over methods based on pre-defined leakage models are unclear, particularly for complex datasets incorporating countermeasures. For details about DLSCA and challenges to be addressed, we refer readers to [PPM<sup>+</sup>23].

## 4 Leakage Model-flexible DLSCA

### 4.1 Physical Leakages Estimation

Let us consider a leaking device with a secret key byte  $k^*$ . The cryptographic operations involve  $k_i$  and (plain or cipher) text byte  $d_i$  (or  $k_{i,m}$  and  $d_{i,m}$  when byte  $m$  is attacked), taken as an  $n$ -bit word (typically  $n = 8$  in related works as most of DLSCA considers AES, which is a byte-oriented cipher). In this case, the real and unknown leakage function  $\psi$  applies to intermediate data  $y = f(k_i, d_i)$  and some additive noise  $Z$ , modeled as a normal random variable  $Z \sim \mathcal{N}(0, \sigma^2)$ . Eq. (5) gives the resulting leakages.

$$\mathbf{t}_i = \psi(f(k_i, d_i)) + Z. \quad (5)$$

The goal of an adversary is to learn the function  $\psi$ , which maps the finite set  $\mathbb{F}_2^n = \{0, 1\}^n$  to the set of real numbers  $\mathbb{R}$ . In DLSCA, the output variables are represented by sensitive operations, such as Sbox input or output of the AES cipher. Then, we can rewrite Eq. (3) as Eq. (6):

$$\boldsymbol{\theta} = \arg \min_{\boldsymbol{\theta}} \frac{1}{N} \sum_i^N L(f_{\boldsymbol{\theta}}(\mathbf{t}_i), \hat{\psi}(f(k_i, d_i))), \quad (6)$$

where  $\hat{\psi}$  denotes the labeling function that estimates the physical leakages involving  $f(k_i, d_i)$ . Since  $\hat{\psi}$  constructs the true label for a profiling model, it is critical for the effectiveness of an attack. There are two ways to estimate the physical leakages: numerical approximation or making hypotheses.

The numerical approximation of the leakage model closely adheres to the principles of stochastic attacks. Specifically, stochastic attacks strive to find an approximate function,  $\hat{\psi}$ , that is as close as possible to the unknown true function,  $\psi$ . Assuming  $\psi$  as a pseudo-boolean function, it can be constructed as a linear combination of monomial basis vectors  $u \in \mathbb{F}_2^n$ . Consequently, a set of real number coefficients  $a_u$  exists such that for a given sensitive intermediate value  $Y \in \mathbb{F}_2^n$ , the leakage model can be reformulated as:

$$\hat{\psi}(y) = \sum_{u \in \mathbb{F}_2^n} a_u \cdot g_u(y), \quad a_u \in \mathbb{R}, \quad (7)$$

where  $g_u$  signifies the base function of the intermediate data. The prevalent assumption is that the leakage bytes rely on the 8 bits. Therefore, the base functions become  $[1, y[1], y[2], \dots, y[8]]$ , with  $y[j]$  representing the  $j_{th}$  bit of  $y$ . Thus,  $\psi$  can be approximated as a multivariate polynomial in the bit-coordinate  $y[j]$  with coefficients belonging to  $\mathbb{R}$ . The typical method for calculating the coefficient  $a_u$  involves employing the ordinary least squares (OLS) method [CK15, SKS09].

However, Eq. (7) also unveils the core limitations of the stochastic attack. This model approximates the linear portion of  $\psi$  using base functions but fails to encompass non-linear parts. Furthermore, it neglects potential multivariate key-dependent noise terms. These two constraints limit the discriminative power when identifying different leakages, leading to mediocre performance when, for instance, dealing with low numbers of side-channel traces [GLRP06].

The leakage model hypothesis, widely used in academia and industry, involves modeling leakages based on certain hypotheses of bit coefficients following Eq. (7). For instance, the following leakage models are commonly used to construct  $\hat{\psi}$ :



- **Hamming distance (HD) and Hamming weight (HW) model** posit that real power consumption is proportional to the number of bit transitions or varies when storing 1 and 0, assuming each bit *equally* contributes to the leakage variation.
- **Identity (ID) model** uses intermediate values as labels to differentiate power traces. This model assumes *different* importance of bits.
- **Most/Least Significant Bit model** assumes the Most/Least Significant Bit (MSB/LSB) is the *only* factor in leakage changes.

Depending on the leakage model being used, the architecture of deep learning differs, especially the output layer. Figure 1 shows different characterizations of the output model, where a representation of a sub-byte (HW or ID) is attacked directly, and the bit model, where a single bit is attacked. The state-of-the-art DLSCA relies on a good estimation of leakage models, but the current forms of DLSCA do not attempt to characterize/assess the leakage directly as they were constructed based on assumptions about the true leakage model, such as the HW model or ID model. For instance, DLSCA cannot answer the question: “Is bit 2 of target data leaking?”. With an imperfect leakage model as the label of a deep learning model, one can hardly reach the optimal attack performance, meaning that the estimated leakage model used for labeling and the true leakage model in the targeted cryptographic operation match with high probability. A common practice is brute-forcing possible leakage models and selecting the best ones to report. Such an approach is problematic because of its time-consuming nature. Meanwhile, conventional profiling SCA relies on, for instance, setting a fixed key on the profiling device to assess each leakage model, potentially leading to stronger attack assumptions. Although evaluation metrics, such as loss, accuracy, or SCA metrics [GBTP08, BHM<sup>+</sup>19, ZZN<sup>+</sup>20, WWK<sup>+</sup>23], could be an option to assess the black-box attack, the result is not as indicative as, for instance, guessing entropy, which directly represents the attack performance [PHJ<sup>+</sup>18].

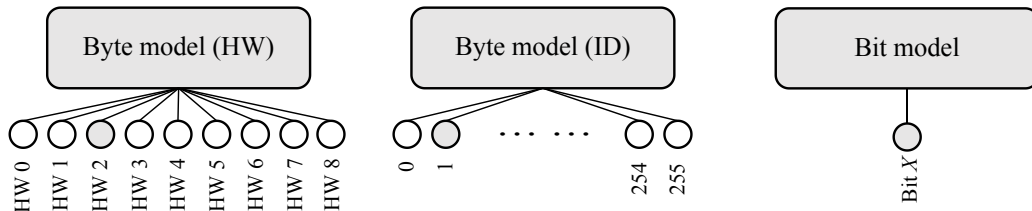


Figure 1: Conventional DLSCA models.

Acknowledging the limitations of the aforementioned methods, an ideal profiling model would require two core capabilities: 1) learning both linear and non-linear aspects of leakage features and 2) enhanced flexibility in learning leakages, avoiding rigid adherence to a pre-defined leakage model. In light of these requirements, a more innate resolution emerges, which involves harnessing the potential of deep learning models. These models, noted for their adeptness in drawing out both linear and non-linear features, can be tailored to ascertain the importance of each bit independently, thus adequately fulfilling both requirements.

## 4.2 Multi-bit Model

The multi-bit model disassembles a byte into separate bits, where each bit is learned individually. Formally, assuming byte  $m$  is attacked, the learning objective is:

$$\theta_m = \arg \min_{\theta} \frac{1}{N} \sum_i^N L(f_{\theta}(t_i), b(f(k_{i,m}, d_{i,m}))), \quad (8)$$

where function  $\mathbf{b}$  binarizes an intermediate data to a finite set  $\mathbb{F}_2^n = \{0, 1\}^n$ ;  $k_{i,m}$  and  $d_{i,m}$  stand for the  $m$ -th byte in a key vector  $\mathbf{k}_i$  and a plaintext vector  $\mathbf{d}_i$ , respectively. Therefore, Eq. (8) can be rewritten as:

$$\boldsymbol{\theta}_m = \begin{cases} \arg \min_{\boldsymbol{\theta}_m} \frac{1}{N} \sum_i^N \mathcal{L}(\mathbf{f}_{\boldsymbol{\theta}_m}(\mathbf{t}_i), \mathbf{b}(\mathbf{f}(k_{i,m}, d_{i,m}))[0])), \\ \arg \min_{\boldsymbol{\theta}_m} \frac{1}{N} \sum_i^N \mathcal{L}(\mathbf{f}_{\boldsymbol{\theta}_m}(\mathbf{t}_i), \mathbf{b}(\mathbf{f}(k_{i,m}, d_{i,m}))[1])), \\ \dots \\ \arg \min_{\boldsymbol{\theta}_m} \frac{1}{N} \sum_i^N \mathcal{L}(\mathbf{f}_{\boldsymbol{\theta}_m}(\mathbf{t}_i), \mathbf{b}(\mathbf{f}(k_{i,m}, d_{i,m}))[n])). \end{cases} \quad (9)$$

Given a probability of each bit with a leakage trace  $\mathbf{t}_i$ , the probability of intermediate data  $y$  can be represented by:

$$\mathbf{p}(y|\mathbf{t}_i) = \prod_j^n \mathbf{p}(\mathbf{b}(\mathbf{f}(k_{i,m}, d_{i,m}))[j]|\mathbf{t}_i; \boldsymbol{\theta}_m). \quad (10)$$

Eq. (9) and Eq. (10) are illustrated in Figure 2. The multi-bit model can be considered a concatenation of bit models that cover all  $n$  bits from a side-channel perspective (here, we assume  $n = 8$ , targeting a single byte). Thanks to its ability to learn each bit, the proposed multi-bit model embodies characteristics shared with both byte (e.g., HW and ID) and bit models (e.g., LSB and MSB) discussed in the previous section: it bears similarities to byte models in that all bits within a byte are taken into account, and akin to bit mode, the bits are treated individually.

On the other hand, the distinguishing features render the multi-bit model particularly advantageous for DLSCA. Indeed, unlike byte models, the multi-bit model does not enforce any pre-conditions on bit importance. The multi-bit model offers flexibility to DLSCA in learning and weighing each bit, thus, sharing more similarity to the stochastic model discussed in Section 4.1. It can easily adapt to any of the existing leakage models, such as the Hamming weight model, where the probability of each bit should be above 50% with the same value, or the LSB leakage model, where only  $\mathbf{p}(\mathbf{b}(k_{i,m}, d_{i,m})[0]|\mathbf{t}_i; \boldsymbol{\theta}_m)$  moving beyond 50%, while the rest remains unchanged.

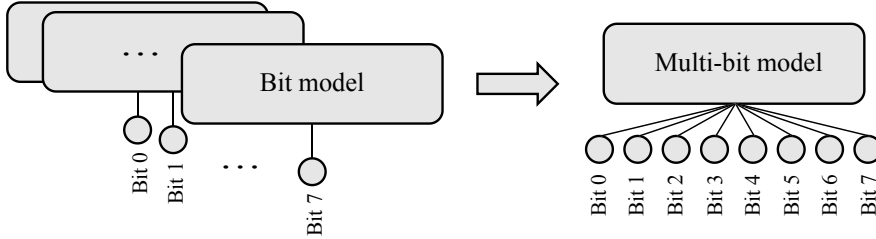


Figure 2: Multi-bit model.

To demonstrate the effectiveness of the multi-bit model, we present attack results with a simulated dataset comprising different leakages following Eq. (11).

$$leakage = \begin{cases} Case1 : y \\ Case2 : hw(y) \\ Case3 : \sum_{i=2}^5 \mathbf{b}(y)[i] \\ Case4 : \prod_{i=0}^3 \mathbf{b}(y)[i] + \prod_{i=4}^7 \mathbf{b}(y)[i] \end{cases} + Z, \quad (11)$$

where  $y$  represents the target sensitive data  $y = \mathbf{Sbox}(d_i \oplus k^*)$ ,  $d_i \in \mathcal{D}$ ;  $d_i$  and  $k^*$  denote a random plaintext byte and a fixed key byte, respectively. Function  $\mathbf{b}$  is a binarization function. Noise  $Z$  is added to all features with  $Z \sim \mathcal{N}(0, \sigma^2)$ . To ensure the noise has the



same effect on each test case, the leakage is normalized between 0 and 1. A total of 10 000 traces were simulated for each test case.

Template attack is used for the benchmark thanks to its interpretable nature. The multi-bit profiling model is constructed by building a Gaussian template on each bit separately<sup>1</sup>; the bit predictions form the probability of a byte following Eq. (10).

We first test the flexibility of the multi-bit model in generalizing to leakage that only leaks ID and HW (cases 1 and 2 in Eq. (11)).  $\sigma$  is set to 0.1 for the low noise setting. The accuracy of each bit is shown in Table 1. Aligned with our expectations, the HW leakages lead to similar accuracy of each bit, indicating the equal contribution to the actual leakage; *bit*<sub>7</sub> of the ID leakage model has the highest accuracy; the rest follows descending order. These observations confirm the ability of the multi-bit model to adjust to different leakage models. Moreover, this result confirms our claim that the commonly used leakage models, including HW and ID, have assumptions on the leakages of each bit.

Table 1: Bit accuracy for ID (case 1) and HW (case 2) leakages.

	<i>bit</i> <sub>7</sub>	<i>bit</i> <sub>6</sub>	<i>bit</i> <sub>5</sub>	<i>bit</i> <sub>4</sub>	<i>bit</i> <sub>3</sub>	<i>bit</i> <sub>2</sub>	<i>bit</i> <sub>1</sub>	<i>bit</i> <sub>0</sub>
ID	0.93	0.60	0.55	0.51	0.50	0.51	0.50	0.50
HW	0.61	0.60	0.60	0.60	0.60	0.60	0.60	0.61

Then, we consider a more realistic scenario for cases 3 and 4, where leakage comes from only specific bits or a combination of bits. Additionally,  $\sigma$  is increased to 0.4 to simulate the realistic noise effect, increasing the attack difficulties. The attack result is shown in Figure 3. Here, the flexibility of the multi-bit model allows it to outperform the HW and ID leakage models, with the leakage models becoming non-ideal in case 3 and case 4. Especially when the real leakage is modeled complexly, as in case 4, the performance gap between the multi-bit and byte models becomes larger. Indeed, the results thus clearly show the advantage of using a multi-bit model in adjusting and extracting complex features.

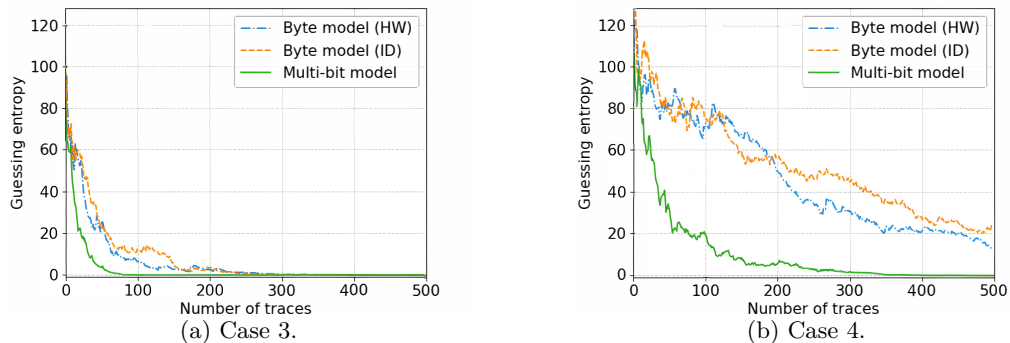


Figure 3: Guessing entropy for case 3 and case 4.

### 4.3 Multi-byte Multi-bit DLSCA

From a deep learning perspective, the multi-bit model compresses multiple bit models into one model. This method aligns with multi-task learning (MTL), where multiple tasks (bit classification) are learned simultaneously. MTL is a well-studied machine learning technique that trains several learning tasks in parallel [Car97, Rud17]. Formally, given

<sup>1</sup>Note that all bits can be modeled with one profiling model with deep learning.

$m$  learning tasks  $\{\mathcal{T}_i\}_{i=1}^m$  where all the tasks or a subset of them are related, multi-task learning aims to learn the  $m$  tasks together to improve the learning of a model for each task  $\mathcal{T}_i$  by leveraging information that result from the training of related tasks [Car97, ZY21].

From an SCA perspective, the side-channel leakages from a target device primarily stem from the switching activities of transistors within the integrated circuit. Considering that the modern CPU/crypto co-processor has at least an 8-bit bus width, different bit classification tasks share a common feature representation based on the original features corresponding to byte processing. multi-bit mode ensures a more powerful representation learned for all the tasks. Meanwhile, the shared representation learned by the related tasks during the training step spares the learning of redundant training parameters, improving the model’s generalization performance.

Formally, Eq. (8) can be extended to Eq. (12) to attack  $n$  bytes.

$$\theta = \begin{cases} \arg \min_{\theta} \frac{1}{N} \sum_i^N L(f_{\theta}(t_i), b(f(k_{i,0}, d_{i,0}))), \\ \arg \min_{\theta} \frac{1}{N} \sum_i^N L(f_{\theta}(t_i), b(f(k_{i,1}, d_{i,1}))), \\ \dots \\ \arg \min_{\theta} \frac{1}{N} \sum_i^N L(f_{\theta}(t_i), b(f(k_{i,n}, d_{i,n}))). \end{cases} \quad (12)$$

We denote the deep learning model following Eq. (12) as multi-byte multi-bit DLSCA, the corresponding leakage model is referred to as multi-byte multi-bit model (MMB). MMB provides multiple advantages. First, the profiling model can better generalize new, unseen leakages by sharing information across multiple bytes. When a profiling model learns to perform multiple tasks, it can lead to capturing common underlying patterns and features beneficial for all tasks. This characteristic makes MMB act as a form of regularization and avoid overfitting to specific patterns or noise. Besides, It can help when individual tasks have limited data. By jointly training on multiple tasks, the profiling model can leverage the data from one task to improve its performance on another, leading to more efficient use of available data.

One can take two approaches in constructing a multi-byte multi-bit DLSCA targeting  $b$  bytes: 1) a single model with  $b * 8$  output nodes, and 2) a tree structure with a main branch and  $b$  subbranches responsible for the classification of bits in each byte. Recall that in a single  $n$ -bit multi-bit model, leakages for different bits may be found at the same time, given that in the target operation, the  $n$  bits are processed simultaneously (e.g.,  $n = 8$ , considering a byte as the basic block). When extending to multiple bytes, we can assume that the bits of each block are processed separately (fits the target software AES implementations used in this paper). In that case, the second approach fits better, wherein a dedicated model (a subbranch) is assigned to each sub-byte to handle its bits separately. We acknowledge that the shared representation of different bytes could still exist. Thus, the main branch is introduced to extract the general features useful for all subbranches.

The proposed architecture is shown in Figure 4. The main branch is responsible for leakage processing and extraction of general features. After the main branch, several multi-bit DLSCAs construct the subbranch for each target byte. The multi-byte multi-bit DLSCA inherits the advantages of multi-bit model, namely, the flexible leakage model. In addition, it brings several benefits. First, learning different tasks ensures that the main branch only learns useful features for all subbranches. Moreover, it is computationally efficient since the entire model has only  $n * 8$  output nodes, while the model in [Mag20] would require  $n * 256$  outputs when attacking all sub-bytes. Knowing that the dimension of the output layer could influence the hyperparameter tuning of a model, our approach reduces the model size, thus increasing the learning efficiency.

Several practical aspects should be emphasized when executing real-world attacks. Like traditional DLSCA, pre-processing leakage measurements is an indispensable step toward an effective attack. Beyond simply normalizing the data, we have identified data augmentation (Section 5.5.1) as the crucial element that drives the success of the proposed

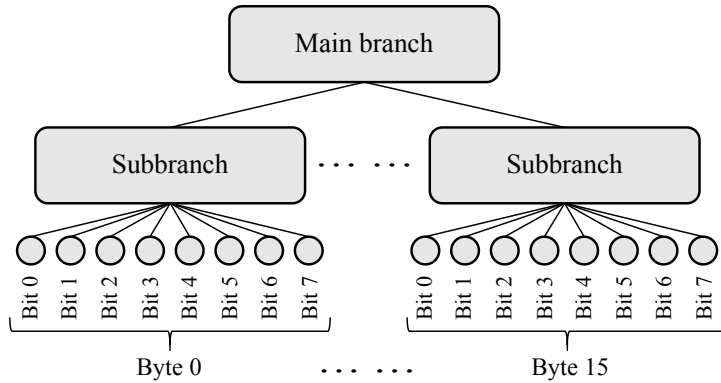


Figure 4: Multi-byte multi-bit DLSCA.

attack. Data augmentation, used as a regularization technique, helps to deter the profiling model from focusing excessively on specific features, allowing it to concentrate instead of global features [Mag20]. Since data leakages are confined to a few features in the realm of SCA, such methodologies can prevent the model from overfitting to non-pertinent features [PSK<sup>+</sup>18, WJB20]. We present a hyperparameter study on data augmentation in Section 5.5.1.

## 5 Experimental Results

This section focuses on investigating the attack performance using various leakage models. In line with Section 4.1, we consider HW, ID, LSB, and MSB. For the proposed methods, we include both multi-bit DLSCA and multi-byte multi-bit DLSCA in the benchmark. To facilitate a fair comparison and offer a comprehensive overview of the general attack performance, all 16 subkeys are attacked.

We use DLSCA to highlight the attack capacity with the multi-bit model. For a fair comparison, the deep learning model and hyperparameters remain *constant* across all attack methods. We acknowledge that tailoring deep learning models (or hyperparameter tuning) for each dataset and method may enhance attack performance. However, this approach also introduces more variables like model complexity and training effort, which could increase the complexity of our benchmarking process significantly. According to the No Free Lunch theorem [HP02], the only way to know which model is best is to evaluate them all, which is impossible. As a result, when trying to see the influence of factors other than the model selection and hyperparameters, the effect of selecting optimal solutions can be neglected by picking a model that works well.

Given its excellent performance in various attack environments, we utilize a convolution neural network based on [PWP22].<sup>2</sup> Since their neural network is one of the best-performing architectures based on our knowledge, we use the same neural network to attack with other leakages models, i.e., HW, ID, LSB, and MSB, for benchmarking purposes. The network structure includes two convolution blocks, each with a convolution layer (kernel numbers: 4, 32; size: 40, 8; stride: 20, 4, for each convolution layer, respectively), an average pooling layer (size: 2; stride: 2), and a batch normalization layer. This is followed by two dense layers with 32 neurons and an output layer with eight neurons. We use Scaled Exponential Linear Unit (*Selu*) for the layer activation [KUMH17], except for the final layer, which uses *Softmax* [Bri90] that converts a vector of  $n$  real numbers into a probability distribution of

<sup>2</sup>The deep learning models were implemented using Python 3.6, with the TensorFlow library version 2.6.0. Training algorithms were executed on a Nvidia GTX 1080TI GPU, managed by Slurm workload manager version 19.05.4.

$n$  possible outcomes. The batch size is set at 512; a hyperparameter study on its influence is given in Section 5.5.2. When the multi-byte multi-bit model is applied, the main branch includes convolution blocks, while each subbranch contains the remaining dense layers and output layers.

Regarding the training epochs (the number of iterations that allow a profiling model to adjust to input data and output labels), the DL model is trained for 200 epochs for each key guess across all test cases. An evaluation of the number of training epochs can be found in Section 5.5.3. To ensure a fair comparison, we apply data augmentation to all DL-based attack methods, achieved by adding a layer right after the input layer that randomly shifts the leakage measurement within a pre-defined augmentation level (the maximum value of random shifting) of 5. We provide a detailed analysis of data augmentation in Section 5.5.1.

As outlined in Section 2.4, we use guessing entropy to evaluate the attack performance for each method. If an attack fails to break the target with the given number of attack traces, its performance is denoted with an 'x'. Otherwise, we calculate the required number of attack traces to achieve a guessing entropy of zero ( $T_{GE0}$ ). Each attack scenario is tested ten times independently to minimize the influence of random elements (e.g., random weight initialization) on the attack performance. The results are averaged to represent the general performance of an attack method.

## 5.1 Datasets

Our experiments consider four datasets consisting of measurements that are software targets protected with a Boolean masking countermeasure. We target the **Sbox** output for all datasets. For all datasets, each key byte is unique (no duplicated key bytes).

**ASCAD\_F.** This dataset contains the measurements from an 8-bit AVR microcontroller running a masked AES-128 implementation [BPS<sup>+</sup>20].

**ASCAD\_R.** This dataset uses the same measurement setup as ASCAD\_F [BPS<sup>+</sup>20]. The difference is that ASCAD\_R also provides traces with random keys (for the profiling phase), providing more sample points per trace. There are also more profiling and attack traces in ASCAD\_R compared to ASCAD\_F. As shown in Table 2, we conduct training and attack with the same number of profiling and attack traces for ASCAD\_F and ASCAD\_R.

**CHES\_CTF** This dataset refers to the CHES Capture-the-flag (CTF) AES-128 measurements released in 2018 for the Conference on Cryptographic Hardware and Embedded Systems (CHES). The traces consist of AES-128 encryption running on a 32-bit STM microcontroller.<sup>3</sup>

**eShard.** This dataset contains EM leakages of a software implementation of AES-128 encryption. The targeted chip is an STM32F446 32-bit microcontroller based on Cortex-M4, running at a clock speed of 30 MHz. A near-field EM acquisition was made using a Langer probe RF-B 0.3-3 through the epoxy package [VTM23].

The raw side-channel measurements from ASCAD\_F, ASCAD\_R, and CHES CTF comprise traces with 100,000, 250,000, and 650,000 sample points per trace, respectively. Handling such long intervals can be time-consuming (and requiring excessively large deep neural network architectures). Thus, we use the resampling technique with a resampling window of 80 [PWP22]. The resampling aims to reduce the dimensionality of the traces; a resampling window indicates the number of trace samples averaged into a single sample. Since the eShard dataset has already been trimmed, we do not conduct any additional

<sup>3</sup><https://chesctf.riscure.com/2018/news>

pre-processing steps. The specific attack settings for these datasets are detailed in Table 2. We use 15% of the profiling traces for the model validation; they are not included in the training process.

Table 2: Summary of the tested datasets.

	ASCAD_F	ASCAD_R	CHES_CTF	eShard
Protection	Boolean masking			
Profiling traces	50 000	50 000	80 000	30 000
Attack traces	5 000			

## 5.2 Performance Evaluation

This section benchmarks the proposed methods with different pre-defined leakage models. The multi-bit and multi-byte multi-bit DLSCA are denoted by MB and MMB, respectively. The MMB attacks 16 sub-bytes simultaneously, while the rest targets each sub-byte in sequence. The attack performance is represented by  $T_{GEO}$ , the number of required attack traces to each guessing entropy of zero.

The performance of the proposed attack scenarios on ASCAD\_F and ASCAD\_R datasets indicates an effective approach to handling various leakage models. The performance across all attack scenarios is similar when attacking the first two key bytes. This is attributed to the lack of masking countermeasures on the first two bytes, allowing MMB, MB, and pre-defined leakage models to extract relevant features and break the target.

MMB and MB models significantly outperform other leakage models when dealing with Boolean masking, displaying superior efficiency in breaking all key bytes. For instance, none of the pre-defined leakage models could recover the  $k_{12}$  of ASCAD\_F, but MMB and MB demonstrated remarkable capabilities by recovering it with just 219 and 59 traces, respectively. Note that our DL model for MMB is very simple. An increased DL complexity could potentially increase the attack performance. This trend is also apparent for the ASCAD\_R dataset, where only MMB and MB could break the target on the same key byte ( $k_{12}$ ). Only 177 and 175 attack traces are required to reach a guessing entropy of zero. These observations confirm our earlier assertion in Section 4.2 that the proposed method can effectively handle complex leakage models. Besides, compared with LSB and MSB leakage models that solely focus on one specific bit, attacking more bits leads to significantly better performance. Due to the capability of the deep learning model to combine physical leakages of mask shares, both MMB and MB break the first-order masking effectively. Still, we note that retrieving these key bytes might be accomplished through refined deep learning architectures or increased training effort [PWP22]. However, it is time-consuming; the tuned model may not function on different leakages. When considering real-world scenarios where the correct key is unknown, an adversary would rely on a fixed model to attack different datasets (a common practice in the industry). In this case, MMB and MB are superior choices due to their flexibility to different leakages.

When comparing MMB and MB, their performance is comparable when attacking these two datasets. One may worry that such a simple main branch would limit the model’s capability to learn so many tasks, but the results show that the features provided by the main branch of the network are sufficient for all 16 tasks. This observation suggests two conclusions. First, each sub-key shares common features and thus can be handled by MMB at once. Second, the used network still has room to be simplified when solely focusing on a single sub-byte.

Tables 5 and 6 show the attack results on the CHES\_CTF and eShard datasets. Aligned with the previous two datasets, MMB and MB maintain outstanding performance compared with other pre-defined leakage models. Interestingly, one can observe that MB

Table 3:  $T_{GEO}$  of each subkey for the ASCAD\_F dataset.

	$k_0$	$k_1$	$k_2$	$k_3$	$k_4$	$k_5$	$k_6$	$k_7$	$k_8$	$k_9$	$k_{10}$	$k_{11}$	$k_{12}$	$k_{13}$	$k_{14}$	$k_{15}$
MMB	1	4	7	20	9	4	22	10	135	15	76	18	219	123	9	69
MB	1	4	5	19	9	7	23	17	185	6	37	35	59	67	9	51
HW	4	4	494	282	419	372	535	590	x	628	1 402	792	x	x	202	x
ID	2	2	x	290	66	74	2 415	111	x	126	1 272	x	x	x	23	x
LSB	17	17	1 552	43	63	81	208	57	2 683	49	138	x	x	1 084	44	61
MSB	27	33	x	1 351	1 079	983	984	866	x	623	x	517	x	x	3 544	x

Table 4:  $T_{GEO}$  of each subkey for the ASCAD\_R dataset.

	$k_0$	$k_1$	$k_2$	$k_3$	$k_4$	$k_5$	$k_6$	$k_7$	$k_8$	$k_9$	$k_{10}$	$k_{11}$	$k_{12}$	$k_{13}$	$k_{14}$	$k_{15}$
MMB	3	5	23	76	28	9	15	35	94	14	74	15	177	110	13	48
MB	2	5	23	89	23	10	17	42	80	27	71	16	145	65	15	26
HW	5	6	1 523	1 088	357	439	1 384	1 160	3 482	808	1 081	824	x	x	415	x
ID	3	2	x	x	309	59	x	x	x	x	x	x	x	x	66	x
LSB	16	11	288	118	43	55	192	98	520	39	89	116	x	391	24	54
MSB	39	60	x	x	4 128	x	1 104	x	x	1 698	x	296	x	x	1 023	x

cannot retrieve all subkeys, while MMB can and performs better in attacking all keys. For instance, only MMB can recover  $k_4$ ,  $k_8$ , and  $k_{12}$  of the eShard dataset, while all other methods fail. Since MMB and MB share identical main branches and subbranches (the only difference is that MMB has more subbranches for each sub-byte), aligned with the discussion in Section 4.3, we conclude that multi-task learning helps generalize each task, leading to robust attack performance. One should note that some attacks/profiling models could perform better than presented results, i.e., attacking different intermediate data [GJS21] or using fine-tuned mode [WPP22, PWP22]. However, they do not influence the conclusion drawn here, as MB and MMB would also benefit from these approaches.

Table 5:  $T_{GEO}$  of each subkey for the CHES\_CTF dataset.

	$k_0$	$k_1$	$k_2$	$k_3$	$k_4$	$k_5$	$k_6$	$k_7$	$k_8$	$k_9$	$k_{10}$	$k_{11}$	$k_{12}$	$k_{13}$	$k_{14}$	$k_{15}$
MMB	89	96	58	100	227	76	48	43	114	55	50	46	3 211	50	59	35
MB	72	57	43	53	78	51	55	56	53	73	40	54	582	40	53	26
HW	2 058	x	803	1 157	x	1 369	1 715	1 121	x	3 013	1 330	4 558	x	x	1 535	1 473
ID	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
LSB	509	812	140	395	387	501	576	228	266	482	439	267	x	533	581	371
MSB	607	486	1 360	x	2 040	3 164	758	x	x	813	2 257	327	x	x	499	x

Finally, aligned with the previous observation, MB and MMB lead to better attack performance than LSB and MSB. Specifically, when attacking the eShard dataset, attacking a single bit seems non-functional; one should combine multiple bits to recover the secret. In this case, MB and MMB are the optimal choices for this task.

### 5.3 Leakage Assessment with the Multi-bit Model

Besides the key recovery, the proposed method provides insight into leakage assessment. Previous results [WPP22, PWP22] show that the CHES\_CTF dataset is only breakable



Table 6:  $T_{GE0}$  of each subkey for the eShard dataset.

	$k_0$	$k_1$	$k_2$	$k_3$	$k_4$	$k_5$	$k_6$	$k_7$	$k_8$	$k_9$	$k_{10}$	$k_{11}$	$k_{12}$	$k_{13}$	$k_{14}$	$k_{15}$
MMB	119	504	858	248	1103	691	488	949	2162	340	106	953	1503	541	417	470
MB	1173	942	908	299	x	1706	1894	1085	x	876	2420	1459	x	1269	1880	1304
HW	1690	2626	1279	922	x	976	973	783	x	1742	1370	1051	x	1208	1557	1176
ID	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
LSB	x	x	4592	2321	x	x	x	4640	x	x	x	x	x	x	x	x
MSB	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

via the HW leakage model, which our results confirm. None of the key bytes can be retrieved via the ID leakage model. Interestingly, the MSB leakage model also leads to mediocre performances. The reasoning for the poor attack performance can be explained from two perspectives. Recall in Section 4.2 when the simulated data has only the ID leakage, MSB is the most significant contributor to actual leakages (see Table 1). However, based on the attack results in Tables 5 and 6, these two datasets contains limited MSB leakages, potentially leading to the failure of the ID leakage model.

The observations can be validated with an evaluation metric. We illustrate the validation accuracy for each bit (256) of the multi-byte multi-bit model in Figure 5. The mean and standard deviation are represented by the blue line and shaded blue areas, respectively. Some studies [KPH<sup>+</sup>19, WPP22] suggest that validation accuracy may be unreliable when working with pre-defined leakage models. However, their conclusion is drawn from the HW and ID leakage model; the bit model we used is unexplored. On the other hand, other evaluation metrics, such as precision and recall, could also evaluate each bit’s prediction. However, since the two classes, 0 and 1, are balanced for each bit, validation accuracy could be considered a good metric for this task.

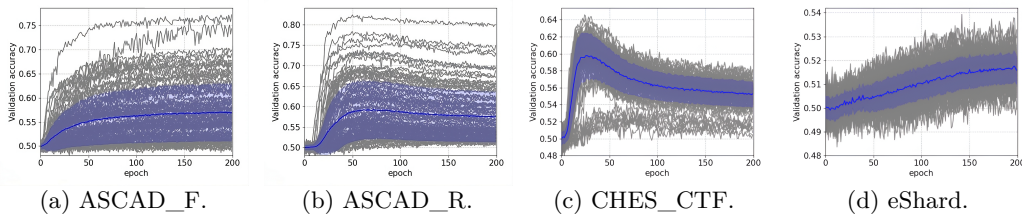


Figure 5: Validation bit accuracy for each dataset.

As shown in Figure 5, for ASCAD\_F and ASCAD\_R, there is a sharp rise in the standard deviation, indicating a broader distribution of bit validation accuracy. The top five bits yielding the highest accuracy are either  $b_7$  (MSB) or  $b_6$  of a byte. This observation aligns with the results in Tables 3 and 4, indicating that the ID leakage model functions well for some bytes. On the other hand, when examining CHES\_CTF and eShard, the accuracy of each bit increases relatively uniformly, as evidenced by their small standard deviation. This supports the observation of the HW accuracy made in Section 4.2 and also corroborates the earlier discussion about the mediocre performance of the ID model.

It is clear that the ASCAD\_R and CHES\_CTF datasets are prone to overfitting, a phenomenon we explore in detail in Section 5.5.3. Additionally, for the CHES\_CTF dataset, one might observe that some bits emerge as ‘outliers’ during the early stages of training. All these outlier bits are associated with sub-byte 12, suggesting that they exhibit distinct leakage features compared to the other sub-bytes. Still, one can observe a steady increase in the validation bit accuracy of the corresponding bits, indicating that our multi-byte multi-bit model is generalized to features that all tasks can share.

## 5.4 Case Study of the SMAesH Challenge

The previous experimental results evaluate the performance of MB and MMB on 8-bit intermediate values. Next, we assess their performance on a more advanced implementation, SMAesH, used as the target for the Capture The Flag challenge in CHES 2023. SMAesH is a hardware implementation of the AES block cipher that uses masking as a countermeasure against side-channel attacks [SC23]. The masking technique in use, Hardware Private Circuits (HPC) [CGLS21], is a glitch-resistant hardware-specialized implementation that can compose arbitrary higher-order masking. The SMAesH dataset on Artix-7 FPGA has  $2^{24}$  traces with random keys for profiling and  $2^{24}$  traces with a fixed key for the attack. There are two versions of the dataset enabling or disabling the knowledge of mask share during the profiling phase. We consider the former dataset in this section. Specifically, the plaintexts and their shares, the keys and their shares, and the random seed (the random number generator output for each cryptography operation) are given along with a simulation package that allows the generation of desired intermediate values [SC23].

Implementation-wise, the data is processed in 32-bit format. Besides, two intermediate shares are processed simultaneously. These characteristics complicate the leakage scenario, as a single-time sample of the trace carries information of multiple shares and bytes. In this case, using multi-task learning and the MB model is a good match. The reasons are twofold: first, learning multiple tasks simultaneously that share information across multiple intermediate values helps to avoid overfitting and learning unrelated noise patterns. Second, using deep learning and a bit model as the last layer of the neural network makes it possible to learn the complex leakage in the case of SMAesH implementation from the bit level. Specifically, aligned with the previous experimental setting, we apply two multi-bit models to attack the Sbox input and output shares, respectively.<sup>4</sup> Then, soft analytical side-channel analysis (SASCA) [VGS14] is applied for the key recovery.<sup>5</sup> The attack works under the same assumptions as profiling attacks. First, we need to profile every intermediate value we are attacking. Then, in the attack phase, we exploit the outcome probabilities for all intermediate values over multiple traces. The last step is combining the exploited information with output probabilities (acquired utilizing the profiles) employing belief propagation and factor graphs as introduced by SASCA. For the attack setting, we started by attacking 8 bits (a single byte) for each intermediate data, resulting in 16 binary outputs for each model (because we have two 8-bit shares). Then, we increased the number of attacked bits to 16, 32, and 128 key bits. Furthermore, to validate our method with different DL architectures, we employed a simple multilayer perceptron (MLP) model to attack this target.

Table 7: The best rank of the keys attacking the SMAesH dataset.

Attacked key size	8 bits	16 bits	32 bits	128 bits
Best key rank/total key space	$2^{5.12}/2^8$	$2^{9.2}/2^{16}$	$2^{17.98}/2^{32}$	$2^{66.90}/2^{128}$
Attack traces	5 000 000			

Table 7 shows the key rank results we reached after the attack using 5 000 000 attack traces. The key rank of the full key (128 bits) is  $2^{66.9}$ , which is better than the attack reported by “Morningstar-1.3” [Cry24]. We compare our attack with theirs because it is the closest attack to our approach.<sup>6</sup> They employed multi-task learning with the ID leakage model to exploit the information spread over the Sbox input, Sbox output, and the transition leakage on the Sbox input wires. In their attack, they could reach the key rank

<sup>4</sup>We have also performed attacks on unmasked intermediate data, but the performance is mediocre.

<sup>5</sup>SASCA includes attacking multiple cryptography operations and their input and output intermediate values simultaneously.

<sup>6</sup>Still, we acknowledge there are better attacks for this dataset. We refer interested readers to [SC23] for more details on these attacks.

of  $2^{68}$  for recovering the whole 128 bits of the key using 5 000 000 traces. With our attack, we can reduce the key space to less than half of the key space they reached. Considering that we did not make any extra effort to optimize the neural network used to attack the SMAesH dataset, the results are surprisingly good, resulting from learning the most accurate leakage model from the bit level. One can observe a relatively lower key space when involving more intermediate data in our model. The key rank is around  $2^{5.13}$  when the target is only one key byte. Assuming each byte leads to the same attack performance, the remaining key space for 16 bytes is  $2^{82.08}$ . Following the same assumption, we would expect that the key space was around  $2^{73.60}$  when we attacked 16 bits and around  $2^{71.92}$  when we attacked 32 bits. Therefore, we conclude that using multi-task learning and seeing all the outputs simultaneously is helpful for the neural network to characterize the input leakages better and extract critical and shared features that may overlap in the same time stamp.

## 5.5 Hyperparameter Study

In this section, we explore the influence of various hyperparameters on the DL model with the multi-bit model. Instead of focusing on one sub-byte, the multi-byte multi-bit DLSCA is used for benchmarking, and all sub-bytes are considered. For a fair comparison, the attack results on 16 sub-bytes are *averaged* to demonstrate the influence of hyperparameter changes.

### 5.5.1 Data Augmentation

As discussed in Section 4.3, the role of data augmentation is crucial for the multi-bit DLSCA. Consequently, we conduct a hyperparameter study specifically focusing on data augmentation, aiming to assess the impact of the augmentation level on the attack performance. The findings from this study are comprehensively presented in Table 8.

Table 8: Hyperparameter study on data augmentation (DA).

	DA-0	DA-5	DA-10	DA-20
ASCAD_F	51	46	243	1 654
ASCAD_R	137	46	88	938
CHES_CTF	274	270	428	328
eShard	x	716	395	450

When the level of data augmentation (the maximum value of random shifting) is set to zero, the multi-bit DLSCA fails to break the eShard dataset. However, a significant performance improvement is noted with the introduction of random shifts in the datasets. Optimal results are consistently observed within the data augmentation range of DA-5 across all test scenarios. As the level of data augmentation reaches 20, there is a notable decline in attack performance in various configurations.

From these observations, we can ascertain the critical role of data augmentation in enhancing the effectiveness of the multi-bit DLSCA. However, employing an excessively high level of data augmentation can have adverse effects, reducing the model’s attack performance. This high augmentation level can increase the complexity of fitting the model to the leakage (as the time location of the leakages becomes more random), requiring either longer training periods or larger models, thus increasing the computation effort.

### 5.5.2 Batch Size

The concept of batch size in a deep learning model pertains to the number of examples (pairings of inputs and outputs) utilized in a single training iteration. This parameter plays

a significant role in shaping a deep learning model’s behavior and overall performance. Notably, a discernible generalization gap exists between small and large batch sizes. Studies illustrate that smaller batch sizes can offer a regularizing effect, often resulting in superior generalization performance [KMN<sup>+</sup>16, ML18]. This suggests that models trained with smaller batches may exhibit enhanced performance on unfamiliar data. As observed in Table 9, this aligns with our expectation that smaller batch sizes generally lead to improved attack performance in the proposed method.

Table 9: Study on the influence of the data size.

	BS-256	BS-512	BS-768	BS-1024
ASCAD_F	44	46	112	177
ASCAD_R	26	45	65	79
CHES_CTF	157	270	165	190
eShard	875	715	695	673

While smaller batch sizes in deep learning models generally lead to better attack performance and superior generalization, it is crucial not to overlook the implications for computational efficiency. Larger batch sizes enable more efficient utilization of computational resources, such as GPUs, which tend to perform optimally when handling computations in larger blocks. When balancing performance against resources, one should consider that larger batch sizes can significantly reduce training time. For instance, in our experiments, a batch size of 1024 could complete tests on all datasets four times faster than a batch size of 256 (around 8 hours). Thus, in designing and training deep learning models, careful consideration should be given to finding the optimal balance between batch size, computational efficiency, and model performance.

### 5.5.3 Training Epochs

Increasing the number of training epochs does not necessarily enhance the mapping capability of a deep learning model from input to output. On the contrary, it could diminish the model’s ability to generalize on unseen datasets, a phenomenon known as ‘overfitting’. Figure 5 clearly shows that ASCAD\_R and CHES\_CTF suffer from overfitting when training with 200 epochs. Table 10 further illustrates the performance fluctuations of the proposed method when trained with varying numbers of epochs. Indeed, training with just 50 epochs proves to be sufficient for most configurations, while an additional 150 epochs (totaling 200) yield stable attack results except the CHES\_CTF, as the attack performance deteriorates with increased epoch training. Indeed, Figure 5c shows that training for 30 epochs results in the peak of validation accuracy for all bytes except byte 12, after which the accuracy diminishes with additional epochs. When training with 30 epochs, excluding byte 12, the required attack traces for each byte drop to an average of only 12 traces per byte for key recovery, surpassing the outcomes shown in Table 5. As for the eShard dataset, there is a consistent decrease in key rank value, in line with the observation in Figure 5d.

Table 10: Study on the influence of the training epoch.

	EP-30	EP-50	EP-100	EP-200	EP-300
ASCAD_F	503	137	44	46	50
ASCAD_R	88	47	40	45	46
CHES_CTF	301	119	137	270	286
eShard	x	3889	1081	715	761

Several strategies can be employed to mitigate overfitting. Data augmentation, as discussed in Section 5.5.1, is one effective solution. Additionally, one could apply an early stopping technique that halts model training if a monitored metric fails to increase over a certain number of epochs. Our results indicate the efficacy of validation accuracy as a metric to mitigate overfitting in the MB and MMB models.

Regarding computational efficiency, the mean training time per dataset is approximately 30 minutes. Given that both sets of 16 bytes are attacked simultaneously, the efficiency of each byte’s attack (less than two minutes) is comparable with the state-of-the-art method, even with careful hyperparameter tuning [ZBHV19, RWPP21].

Our analysis highlights that hyperparameter tuning, specifically adjusting the data augmentation level, impacts the attack performance. Nonetheless, the model demonstrates resilience to hyperparameter variations, with only one hyperparameter setting (zeroing the data augmentation level) leading to a failed attack. It is important to note that we employ a single model to attack all four datasets, achieving consistent performance. This underscores the simplicity of our model’s hyperparameter tuning. Furthermore, it emphasizes the robustness of the proposed multi-byte multi-bit approach, making it a reliable profiling solution across varying attack scenarios.

## 6 Discussion

The multi-bit model effectively bridges two approaches in estimating physical leakages: numerical approximation and leakage hypothesis. Typically, numerical approximations decompose the target intermediate data into bits and then estimate their coefficients. The multi-bit model follows the same approach by estimating all bits simultaneously. Integrating deep learning enables extracting non-linear features from side-channel leakages, potentially improving the estimation of physical leakages. Compared to relevant methods, such as stochastic models [SLP05], the multi-bit leakage model can be dynamically adapted to various pre-defined leakage models during the profiling phase with different implementations. This adaptability offers enhanced flexibility, allowing the profiling model to learn more effectively from physical leakages without being constrained by suboptimal leakage model selection. Compared with [ZXF+20] that profiles each bit individually, we further investigate the multi-bit model’s leakage model-free characteristic. Briefly speaking, multi-label learning is familiar to the machine learning community. From our perspective, they reuse multi-label learning as a new architecture of DLSCA with the hope of producing better results. However, from the original paper, the improvement is insignificant. We put much effort into discussing the capability of the multi-bit model to adapt to different leakage models on both the simulated and the real datasets. Based on these studies, the reason why the multi-bit model is better for SCA is well-understood. Besides, we extend the multi-bit model to the multi-byte multi-bit model, attacking all 16 bytes simultaneously. This is the first paper that can recover all key bytes at once. Another notable strategy is the One-vs-All (OvA) method, which decomposes a multi-class problem into multiple binary classification problems. For a problem with  $N$  classes, OvA establishes  $N$  distinct binary classifiers, each differentiating one class from the rest. Acharya et al. utilized the OvA method to train 256 sub-models for each key byte, aggregating these models for the final prediction [AGF22]. However, this method assumes that each intermediate value produces distinct physical leakages that may not fit the reality. Improved label encoding, such as using Hamming Weight, could yield better OvA results. Nonetheless, potential class imbalances and high computational complexity, particularly for attacking all 16 bytes (requiring training of  $256 \times 16$  models), are significant challenges. From both attack and computation perspectives, our approach demonstrates superior performance.

The multi-bit model, however, is not without its limitations. The bit decomposition of intermediate data does not guarantee that all bits can be accurately learned, leading

to uncertainties in certain bits. For example, in Table 1, when the real leakage model is ID,  $bit_0$  to  $bit_4$  exhibit low accuracy, akin to random guessing. Deep learning models, which often employ gradient descent for loss function minimization, tend to learn easier features first, such as the Most Significant Bit (MSB). This preference can leave complex features, like the Least Significant Bit (LSB), less learned in the initial training stages. Gohr et al. observed similar patterns and proposed scattershot encoding to address this issue [GLS22]. This technique involves labeling traces with the HW of random bit subsets and training multiple models on these varied encodings, ultimately recovering all bits. Although effective, this approach demands significant computational resources, particularly when attacking all 16 bytes.

Finally, the multi-bit model applies to both software and hardware implementations of symmetric cryptography. In hardware contexts, the model can adapt to scenarios where the power consumption of a circuit is influenced by the number of bit transitions in a register [FYH<sup>+</sup>23]. Since hardware crypto implementations often reuse gates for round calculations to conserve chip area [HAHH06], the proposed multi-byte multi-bit model (MMB) is also effective in attacking hardware implementations, showcased in Section 5.4. As mentioned, MMB can be easily adapted to different implementations. For instance, when the data bus is 32-bit, one can adapt the output of each subbranch from 8 to 32.

## 7 Conclusions and Future Work

This paper introduces a novel multi-bit model that learns each bit separately. Unlike conventional profiling attacks, our method is adaptable to any specific leakage model, which offers increased flexibility in fitting the actual leakages. Simultaneously, we employ multi-task learning to attack multiple sub-bytes concurrently, leading to efficient key recovery without the need to attack each byte separately. By applying our framework to four publicly available masked AES datasets, we obtain profiling attack results that significantly surpass models using pre-defined leakage models for leakage labeling. Importantly, no effort is expended in hyperparameter tuning, demonstrating its generality across different attack scenarios.

There are several potential avenues of investigation. First, the deep learning network could be enhanced, e.g., through residual networks, to strengthen the connection between the input and each task. Specifically, the shortcut could directly connect with the model’s subbranch, potentially reducing the reliance on the main branch and its feature extraction capability. Second, while the current method treats each bit independently, exploring methods to reinforce inter-bit connections would be worthwhile. For instance, building an interconnection between each sub-branch could be interesting to explore. Lastly, it will be interesting to explore the capability of the proposed method in attacks without masking knowledge, e.g., testing on another version of the SMAesH dataset without mask shares.

## Acknowledgments

This work received funding in the framework of the NWA Cybersecurity Call with project name PROACT with project number NWA.1215.18.014, which is (partly) financed by the Netherlands Organisation for Scientific Research (NWO). Additionally, this work was supported in part by the Netherlands Organization for Scientific Research NWO project DISTANT (CS.019).



## References

- [AGF22] Rabin Y Acharya, Fatemeh Ganji, and Domenic Forte. Information theory-based evolution of neural networks for side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2022. doi:[10.46586/tches.v2023.i1.401-437](https://doi.org/10.46586/tches.v2023.i1.401-437).
- [APSQ06] Cédric Archambeau, Eric Peeters, F-X Standaert, and J-J Quisquater. Template attacks in principal subspaces. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 1–14. Springer, 2006. doi:[10.1007/11894063\\_1](https://doi.org/10.1007/11894063_1).
- [BHM<sup>+</sup>19] Olivier Bronchain, Julien M Hendrickx, Clément Massart, Alex Olshevsky, and François-Xavier Standaert. Leakage certification revisited: Bounding model errors in side-channel security evaluations. In *Advances in Cryptology-CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39*, pages 713–737. Springer, 2019. doi:[10.1007/978-3-030-26948-7\\_25](https://doi.org/10.1007/978-3-030-26948-7_25).
- [BHvW12] Lejla Batina, Jip Hogenboom, and Jasper GJ van Woudenberg. Getting more from pca: first results of using principal component analysis for extensive power analysis. In *Topics in Cryptology-CT-RSA 2012: The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27–March 2, 2012. Proceedings*, pages 383–397. Springer, 2012. doi:[10.1007/978-3-642-27954-6\\_24](https://doi.org/10.1007/978-3-642-27954-6_24).
- [BPS<sup>+</sup>20] Ryad Benadjila, Emmanuel Prouff, Rémi Strullu, Eleonora Cagli, and Cécile Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *J. Cryptographic Engineering*, 10(2):163–188, 2020. doi:[10.1007/s13389-019-00220-8](https://doi.org/10.1007/s13389-019-00220-8).
- [Bri90] John S Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications*, pages 227–236. Springer, 1990. doi:[10.1007/978-3-642-76153-9\\_28](https://doi.org/10.1007/978-3-642-76153-9_28).
- [Car97] Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997. doi:[10.1007/978-1-4615-5529-2\\_5](https://doi.org/10.1007/978-1-4615-5529-2_5).
- [CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures: Profiling attacks without pre-processing. In *Cryptographic Hardware and Embedded Systems-CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 45–68. Springer, 2017. doi:[10.1007/978-3-319-66787-4\\_3](https://doi.org/10.1007/978-3-319-66787-4_3).
- [CDSU23] Gaëtan Cassiers, Henri Devillez, François-Xavier Standaert, and Balazs Udvarhelyi. Efficient regression-based linear discriminant analysis for side-channel security evaluations: Towards analytical attacks against 32-bit implementations. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):270–293, 2023. doi:[10.46586/tches.v2023.i3.270-293](https://doi.org/10.46586/tches.v2023.i3.270-293).
- [CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Trans. Computers*, 70(10):1677–1690, 2021. doi:[10.1109/TC.2020.3022979](https://doi.org/10.1109/TC.2020.3022979).

- [CK13] Omar Choudary and Markus G. Kuhn. Efficient template attacks. In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, volume 8419 of *LNCS*, pages 253–270. Springer, 2013. URL: [http://dx.doi.org/10.1007/978-3-319-08302-5\\_17](http://dx.doi.org/10.1007/978-3-319-08302-5_17), doi:10.1007/978-3-319-08302-5\_17.
- [CK15] Marios O Choudary and Markus G Kuhn. Efficient stochastic methods: Profiled attacks beyond 8 bits. In *Smart Card Research and Advanced Applications: 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers 13*, pages 85–103. Springer, 2015. doi:10.1007/978-3-319-16763-3\_6.
- [CRR03] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4*, pages 13–28. Springer, 2003. doi:10.1007/3-540-36400-5\_3.
- [Cry24] Simple Crypto. Smaesh challenge leaderboard, 2024. Accessed: 2024-07-08. URL: <https://smaesh-challenge.simple-crypto.org/leaderboard.html>.
- [FYH<sup>+</sup>23] Yuta Fukuda, Kota Yoshida, Hisashi Hashimoto, Kunihiro Kuroda, and Takeshi Fujino. Profiling deep learning side-channel attacks using multi-label against aes circuits with rsm countermeasure. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 106(3):294–305, 2023. doi:10.1587/transfun.2022cip0015.
- [GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 426–442. Springer, 2008. doi:10.1007/978-3-540-85053-3\_27.
- [GJS21] Aron Gohr, Sven Jacob, and Werner Schindler. Subsampling and knowledge distillation on adversarial examples: New techniques for deep learning based side channel evaluations. In *Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers 27*, pages 567–592. Springer, 2021. doi:10.1007/978-3-030-81652-0\_22.
- [GLRP06] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. stochastic methods. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 15–29. Springer, 2006. doi:10.1007/11894063\_2.
- [GLS22] Aron Gohr, Friederike Laus, and Werner 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*, pages 397–437, 2022. doi:10.46586/tches.v2022.i4.397-437.
- [HAHH06] Panu Hamalainen, Timo Alho, Marko Hannikainen, and Timo D Hamalainen. Design and implementation of low-area and low-power aes encryption hardware core. In *9th EUROMICRO conference on digital system design (DSD'06)*, pages 577–583. IEEE, 2006. doi:10.1109/dsd.2006.40.

- [HGM<sup>+</sup>11] Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle. Machine learning in side-channel analysis: a first study. *J. Cryptogr. Eng.*, 1(4):293–302, 2011. doi:10.1007/s13389-011-0023-x.
- [HK18] Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *CoRR*, abs/1806.03852, 2018. URL: <http://arxiv.org/abs/1806.03852>, arXiv:1806.03852.
- [HKSS12] Annelie Heuser, Michael Kasper, Werner Schindler, and Marc Stöttinger. A new difference method for side-channel analysis with high-dimensional leakage models. In *Lecture Notes in Computer Science*, pages 365–382. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-27954-6\_23.
- [HP02] Yu-Chi Ho and David L Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115:549–570, 2002. doi:10.1023/a:1021251113462.
- [JW02] Richard Arnold Johnson and Dean W. Wichern. *Applied multivariate statistical analysis*. Prentice Hall, Upper Saddle River, NJ, 5. ed edition, 2002. URL: [http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+330798693&sourceid=fbw\\_bibsonomy](http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+330798693&sourceid=fbw_bibsonomy), doi:10.1007/978-3-540-72244-1.
- [KMN<sup>+</sup>16] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.
- [KPH<sup>+</sup>19] Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic. Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 148–179, 2019. doi:10.46586/tches.v2019.i3.148-179.
- [KUMH17] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980, 2017. doi:10.5555/3294771.3294864.
- [LMBM13] Liran Lerman, Stephane Fernandes Medeiros, Gianluca Bontempi, and Olivier Markowitch. A Machine Learning Approach Against a Masked AES. In *CARDIS*, Lecture Notes in Computer Science. Springer, November 2013. Berlin, Germany. doi:10.1007/978-3-319-14123-7\_5.
- [Mag20] Housseem Maghrebi. Deep learning based side-channel attack: a new profiling methodology based on multi-label classification. *Cryptology ePrint Archive*, 2020.
- [ML18] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.
- [MPP16] Housseem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff. Breaking cryptographic implementations using deep learning techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 3–26. Springer, 2016. doi:10.1007/978-3-319-49445-6\_1.
- [PHG17] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. Template attack versus bayes classifier. *Journal of Cryptographic Engineering*, 7(4):343–351, September 2017. doi:10.1007/s13389-017-0172-7.

- [PHJ<sup>+</sup>18] Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco 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):209–237, Nov. 2018. URL: <https://tches.iacr.org/index.php/TCHES/article/view/7339>, doi:10.13154/tches.v2019.i1.209–237.
- [PHJB19] Stjepan Picek, Annelie Heuser, Alan Jovic, and Lejla Batina. A systematic evaluation of profiling through focused feature selection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(12):2802–2815, 2019. doi:10.1109/tvlsi.2019.2937365.
- [PPM<sup>+</sup>23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11), feb 2023. doi:10.1145/3569577.
- [PSK<sup>+</sup>18] Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay. On the performance of convolutional neural networks for side-channel analysis. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 157–176. Springer, 2018. doi:10.1007/978-3-030-05072-6\_10.
- [PWP22] Guilherme Perin, Lichao Wu, and Stjepan Picek. Exploring feature selection scenarios for deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 828–861, 2022. doi:10.46586/tches.v2022.i4.828–861.
- [RB24] Azade Rezaeezade and Lejla Batina. Regularizers to the rescue: fighting overfitting in deep learning-based side-channel analysis. *Journal of Cryptographic Engineering*, pages 1–21, 2024. doi:10.1007/s13389-024-00361-5.
- [Rud17] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.
- [RWPP21] Jorai Rijdsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021(3):677–707, Jul. 2021. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8989>, doi:10.46586/tches.v2021.i3.677–707.
- [SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 411–425. Springer, 2008. doi:10.1007/978-3-540-85053-3\_26.
- [SC23] SIMPLE-Crypto. Smaesh: Technical documentation. <https://www.simple-crypto.org/activities/smaesh/>, 2023. Accessed: 2024-07-02.
- [SKS09] François-Xavier Standaert, François Koeune, and Werner Schindler. How to compare profiled side-channel attacks? In *Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings 7*, pages 485–498. Springer, 2009. doi:10.1007/978-3-642-01957-9\_30.

- [SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In *Cryptographic Hardware and Embedded Systems—CHES 2005: 7th International Workshop, Edinburgh, UK, August 29–September 1, 2005. Proceedings 7*, pages 30–46. Springer, 2005. doi:10.1007/11545262\_3.
- [SMY09] François-Xavier Standaert, Tal G Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In *Advances in Cryptology—EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26–30, 2009. Proceedings 28*, pages 443–461. Springer, 2009. doi:10.1007/978-3-642-01001-9\_26.
- [Tim19] Benjamin Timon. Non-profiled deep learning-based side-channel attacks with sensitivity analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 107–131, 2019. doi:10.46586/tches.v2019.i2.107-131.
- [VGS14] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Soft analytical side-channel attacks. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 282–296. Springer, 2014. URL: [https://doi.org/10.1007/978-3-662-45611-8\\_15](https://doi.org/10.1007/978-3-662-45611-8_15), doi:10.1007/978-3-662-45611-8\_15.
- [VTM23] Aurélien Vasselle, Hugues Thiebauld, and Philippe Maurine. Spatial dependency analysis to extract information from side-channel mixtures: extended version. *Journal of Cryptographic Engineering*, pages 1–17, 2023. doi:10.1007/s13389-022-00307-9.
- [WAGP20] Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart 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. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8586>, doi:10.13154/tches.v2020.i3.147-168.
- [WEG87] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. doi:10.1016/0169-7439(87)80084-9.
- [WJB20] Yoo-Seung Won, Dirmanto Jap, and Shivam Bhasin. Push for more: On comparison of data augmentation and smote with optimised deep learning architecture for side-channel. In *Information Security Applications: 21st International Conference, WISA 2020, Jeju Island, South Korea, August 26–28, 2020, Revised Selected Papers 21*, pages 227–241. Springer, 2020. doi:10.1007/978-3-030-65299-9\_18.
- [WPP22] Lichao Wu, Guilherme Perin, and Stjepan Picek. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. *IEEE Transactions on Emerging Topics in Computing*, 2022. doi:10.1109/tetc.2022.3218372.

- [WWK<sup>+</sup>23] Lichao Wu, Léo Weissbart, Marina Krček, Huimin Li, Guilherme Perin, Lejla Batina, and Stjepan Picek. Label correlation in deep learning-based side-channel analysis. *IEEE Transactions on Information Forensics and Security*, 2023. doi:10.1109/tifs.2023.3287728.
- [ZBC<sup>+</sup>23] Gabriel Zaid, Lilian Bossuet, Mathieu Carbone, Amaury Habrard, and Alexandre Venelli. Conditional variational autoencoder based on stochastic attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 310–357, 2023. doi:10.46586/tches.v2023.i2.310-357.
- [ZBHV19] Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli. Methodology for efficient cnn architectures in profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(1):1–36, Nov. 2019. URL: <https://tches.iacr.org/index.php/TCHES/article/view/8391>, doi:10.13154/tches.v2020.i1.1-36.
- [ZXF<sup>+</sup>20] Libang Zhang, Xinpeng Xing, Junfeng Fan, Zongyue Wang, and Suying Wang. Multilabel deep learning-based side-channel attack. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(6):1207–1216, 2020. doi:10.1109/TCAD.2020.3033495.
- [ZY21] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021. doi:10.1109/TKDE.2021.3070203.
- [ZZN<sup>+</sup>20] Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai 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*, pages 73–96, 2020. doi:10.46586/tches.v2020.i3.73-96.