Breaking Free: Leakage Model-free Deep Learning-based Side-channel Analysis

Lichao Wu[1], Amir Ali-pour[4], Azade Rezaeezade[2], Guilherme Perin[3] and Stjepan Picek[1]

[1] Radboud University, The Netherlands
[2] Delft University of Technology, The Netherlands
[3] Leiden University, The Netherlands
[4] Independent Researcher

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 that does not assume any specific leakage model, referred to as the multibit model. Instead of trying to learn a representation of the target intermediate data (label), 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 multibit model can align with existing leakage models like the Hamming weight and Most Significant Bit leakage models while also possessing the flexibility to adapt to complex leakage scenarios. To further improve the attack efficiency, we extend the multibit model to simultaneously attack all 16 subkey bytes, which requires negligible computational effort. Based on our preliminary analysis, two of the four considered datasets could only be broken using a Hamming Weight leakage model. Using the same model, the proposed methods can efficiently crack all key bytes across four considered datasets. Our work, thus, signifies a significant step forward in deep learning-based side-channel attacks, showcasing a high degree of flexibility and efficiency without any presumption of the leakage model.

Keywords: Side-channel Analysis · Deep learning · Multibit model · Multi-task learning.

1 Introduction

Side-channel analysis (SCA) is a powerful tool for analyzing unintended leakages during secret processing. 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 does not have access to a duplicate of the targeted device. As a result, the adversary must consider all measurements containing secret information 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+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 needs to label the collected traces based on an assumption about the underlying 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 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. Besides, they impose a degree of pre-existing conditions on bit significance, which may inadvertently reduce the flexibility of the learning process.

Multiple methods have been proposed to address this issue, such as stochastic modeling [SLP05, ZBC+23] or label distribution [WWK+23]. Recently, Zhang et al. introduced a multi-label deep learning-based SCA that treats each bit in a byte as a separate label [ZXF+20]. Despite its seemingly unimpressive attack performance, we regard this method as a promising candidate for resolving the leakage model problem. Drawing inspiration from this method, we propose a novel approach, referred to as multibit model, to address the leakage model issue. Specifically, this strategy bypasses the limitation of pre-defined leakage models, enhancing the profiling model's ability to learn leakages. Furthermore, we extend the multibit model to attack all subkeys simultaneously, leveraging multi-task learning. Based on our knowledge, it is the first time one could attack all 16 bytes simultaneously. Being unrestrained from pre-defined conditions or assumptions, the proposed method emerges as a promising and robust solution in DLSCA.

Our main contributions are:

1. We analyze the limitations of the existing profiling attack methods and then propose the multibit model with no assumption of leakage model.

2. We propose a new attack method, multibyte multibit DLSCA, that can simultaneously attack all subkeys (in our case, 16 AES subkeys).

3. We perform case studies to showcase our analysis, which is further validated on several publicly available datasets.

4. We provide an ablation 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 multibit model-based attacks.

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. Sections 5 and 6 provide a case study of the proposed method and present experimental results using publicly available datasets, respectively. Finally, we summarize our findings and discuss potential avenues for future research in Section 7.

2 Preliminaries

This section introduces the notation we use. Next, we delve into the domain of profiling side-channel analysis, providing specifics on both the template attack and deep learning-based SCAs. Finally, we discuss the techniques employed to evaluate the performance of side-channel attacks.

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, which is 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 $\mathbf{k}_i$. 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 O 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 Profiling Side-channel Analysis

Profiling SCA assumes an identical clone device (or at least similar) to the target device. The clone device is under the full control of the attacker. The attacker employs O measurements from the profiling device to construct a model and then uses Q measurements from the target device to deduce the secret information. The type of profiling model constructed varies depending on the profiling technique used. 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 [CRR02a]. 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:

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

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 Σ_y:

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

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 $\mathsf{f}_{\boldsymbol{\theta}}$ to predict labels on previously unseen data. Most of the supervised learning methods follow the Empirical Risk Minimization (ERM) framework, where the model parameters $\boldsymbol{\theta}$ are obtained by solving the optimization problem:

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t_i}), y_i), \tag{3}$$

where L is the loss function. The trained model $\mathsf{f}_{\boldsymbol{\theta}}$ 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 more profiling traces, which might limit its applicability in some scenarios (while, in practice, this could not be a problem due to GPU support for deep learning algorithms).

2.3 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:

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

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 $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$, 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 Works

The side-channel analysis domain has been immersed in the study of profiling attacks for over two decades. The pioneering work in this area was conducted by Chari et al., who introduced the concept of the template attack. The template attack is considered the most powerful approach from an information-theoretic perspective [CRR02b]. Unfortunately, this attack's practical implementation is complicated due to several assumptions, including unlimited profiling traces and Gaussian-distributed noise [LPB+15]. The stochastic model [SLP05] is another profiling attack approach based on modeling the leakage functions with pre-defined polynomials, but it is less considered in recent research due to its limitation in handling non-linear features.

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+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+19] highlighted the potential of convolutional neural networks (CNNs) in breaking protected targets. Techniques for improving attack performance using regularization were also explored [KPH+19, RB22, HK18]. Subsequent work by Zaid et al. [ZBHV19] and Wouters et al. [WAGP20] delved deeper into the design methodologies for CNNs, achieving unprecedented attack performance on datasets secured by masking and hiding countermeasures. Additionally, Perin et al. demonstrated that neural network ensembles could enhance attack performance significantly [PCP20]. They also reported that the random search of neural network architectures yielded exceptional results, making the search for well-performing neural networks potentially easier. Zhang et al. introduced a multilabel learning approach based on bits [ZXF+20], decreasing the model size without adversely impacting the attack performance. In recent studies, Lu et al. showed that using raw traces instead of feature intervals can improve attack performance, albeit at the cost of dealing with 1) large-scale measurements and 2) complex hyperparameter tuning process [LZC+21]. In their follow-up work, Perin et al. used resampled raw traces and smaller neural networks to achieve excellent attack performance [PWP22], even demonstrating scenarios where a single attack trace sufficed to break the target.

The studies discussed thus far employ pre-defined leakage models to label leakage traces. However, these may not necessarily align well with the target dataset. For example, in [PWP22], Perin et al. used both the Identity and Hamming Weight leakage models to attack the CHES_CTF dataset. However, they indicated that the hyperparameter search could not find a CNN model that could recover the key with the Identity leakage model. In [WPP20], 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.

To address this issue, Wu et al. incorporated label distribution into the profiling phase to address this limitation, significantly reducing the number of required training traces [WWK+23]. 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 pioneering work, Zaid et al. developed a conditional variational autoencoder methodology for stochastic attacks [ZBC+23]. This approach mitigates the black-box aspect of deep learning and facilitates a more straightforward process for architectural design.

So far, the predominant focus of the research has been treating the optimization of the labeling function 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 not as pronounced, particularly for complex datasets incorporating countermeasures. For instance, for the CHES_CTF datasets (detailed in Section 6.1), experience shows they are only breakable with the HW leakage model [WPP22, PWP22] while the ID leakage model would always lead the mediocre performance. For details about DLSCA and challenges to be addressed, we refer readers to [PPM+23].

4 Model-free DLSCA

This section introduces the motivation and advantages of learning leakages from the bit level, then proposes multibit and multibyte multibit models without assumptions on leakage models.

4.1 Moving from Byte to Bit

Let us consider a leaking device with a secret key k^*. The cryptographic operations involve k_i and (plain or cipher) text d_i, 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 leakage function ψ applies to intermediate data $y = \mathsf{f}(k_i, d_i)$ and some additive noise Z, modeled as a normal random variable $Z \sim \mathcal{N}(0, \sigma^2)$. Eq. (5) gives the resulting leakage X.

$$\mathbf{t_i} = \psi(\mathsf{f}(y)) + Z. \tag{5}$$

The goal of an attacker is to learn the function ψ, 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):

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t_i}), \hat{\psi}(\mathsf{f}(k_i, d_i)), \tag{6}$$

where $\hat{\psi}$ denotes the labeling function that returns the leakage value involving $\mathsf{f}(k_i, d_i)$. $\hat{\psi}$ constructs the true label for a profiling model and is thus critical for the effectiveness of an attack. From the SCA perspective, $\hat{\psi}$ should represent the leakage characteristic when processing different bytes. From the deep learning perspective, $\hat{\psi}$ determines the number of classes and directly influences the classification complexity. There are two ways to construct such a leakage model, detailed in the following sections.

4.1.1 Approximation of the Leakage Model

The first approach 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, ψ. Considering ψ 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 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:

$$\hat{\psi}(y) = \sum_{u \in \mathbb{F}2^n} a_u \cdot g_u(y), \ a_u \in \mathbb{R}, \tag{7}$$

where g_u signifies the base function of the intermediate data, the prevalent assumption here is that the leakage bytes rely on the 8 bits. Therefore, the base functions become [1, $y[1]$, $y[2]$, $\cdots$, $y[8]$], with $y[j]$ representing the j_{th} bit of y. Thus, ψ 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 ψ using base functions but fails to encompass non-linear parts. Furthermore, it neglects potential multivariate key-dependent noise terms [GLRP06]. These two constraints limit the discriminative power when identifying different leakages, leading to mediocre performance when dealing with noisy or mask-protected datasets.

4.1.2 Pre-defined Leakage Model

The second method, widely used in academia and industry, involves modeling leakages based on certain assumptions of bit coefficients following Eq. (7). 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, it relies on, for instance, the knowledge of the secret key to assess each leakage model, potentially leading to stronger attack assumptions. Although evaluation metrics, such as loss, accuracy, or SCA metric [ZZN+20, WWK+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.

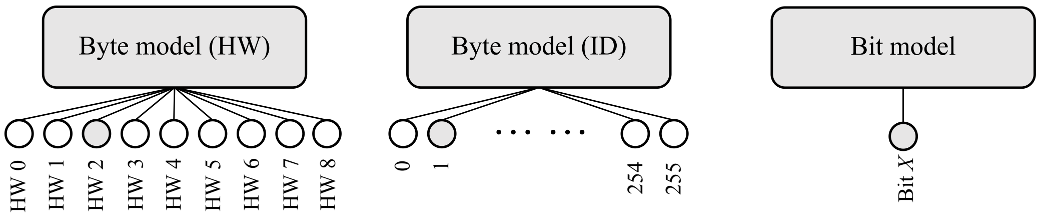


Figure 1: Conventional DLSCA models.

4.1.3 An 'Ideal' Solution

Acknowledging the constraints 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. Details of this solution are elaborated in the following sections.

4.2 Multibit Model

The multibit model disassembles a byte into separate bits, where each bit is learned individually. Formally speaking, the learning objective is:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \mathsf{b}(k_i, d_i)), \tag{8}$$

where function b maps an intermediate data to a finite set $\mathbb{F}_2^n = \{0,1\}^n$. Therefore, Eq. (8) can be rewritten as:

$$\boldsymbol{\theta} = \begin{cases} \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \mathsf{b}(k_i, d_i)[0]), \\ \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \mathsf{b}(k_i, d_i)[1]), \\ \cdots \\ \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i}^{N} \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \mathsf{b}(k_i, d_i)[n]). \end{cases} \tag{9}$$

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

$$\mathbf{p}(y|\mathbf{t}_i) = \prod_{j}^{n} \mathbf{p}(\mathsf{b}(k_i, d_i)[j]|\mathbf{t}_i; \boldsymbol{\theta}). \tag{10}$$

Eqs. (9) and Equation 10 are better illustrated in Figure 2. Indeed, the multibit 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). The proposed multibit model embodies characteristics shared with both byte and bit models 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. However, the distinguishing features render the multibit model particularly advantageous for DLSCA. Unlike byte models, the multibit model does not enforce any pre-conditions on bit importance. The multibit model offers flexibility to DLSCA in learning and weighing each bit. It can easily fit 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 $\mathbf{p}(\mathsf{b}(k_i, d_i)[0]|\mathbf{t}_i; \boldsymbol{\theta})$ moving beyond 50% while the rest remains unchanged. A case study is presented in Section 5 to validate this assumption.

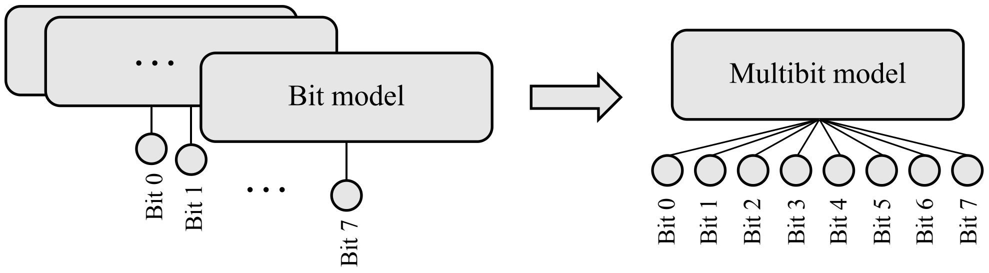


Figure 2: Multibit model.

4.3 Multibyte Multibit DLSCA

From a deep learning perspective, the multibit 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]. We note that the multi-task paradigm is not new for SCA. Indeed, Maghrebi in [Mag20] and Masure and Strullu in [MS21] used multi-task learning. Formally, given 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 power/EM dissipation from a target device primarily stems from the switching activities of transistors within the integrated circuit. Considering that the modern CPU/crypto co-processor has at least 8-bit bus width, different bit classification tasks share a common feature representation based on the original features corresponding to byte processing. Multibit 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. (9) can be extend to Eq. (11) to attack n bytes.

$$\boldsymbol{\theta} = \begin{cases} \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_i^N \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \mathsf{b}(k_0, d_0)), \\ \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_i^N \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \mathsf{b}(k_1, d_1)), \\ \dots \\ \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_i^N \mathsf{L}(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t}_i), \mathsf{b}(k_n, d_n)). \end{cases} \tag{11}$$

One can take two approaches in constructing a multibyte multibit model targeting n bytes: 1) a single model with $n * 8$ output nodes, and 2) a tree structure with a main branch and n subbranches responsible for the classification of bits in each byte. Let us recall that in a single n-bit multibit 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). For a multibyte multibit model, we can assume that the bits of each block are processed separately (fits the target soft 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 in this case. Thus, the main branch is introduced to extract the general features useful for all subbranches.

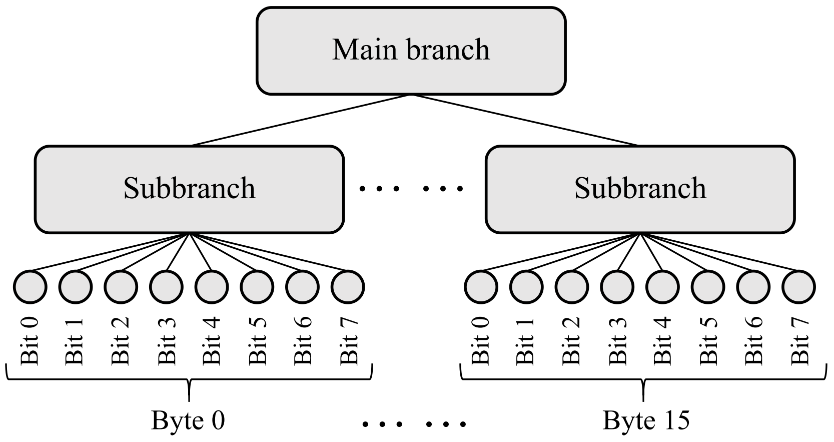


Figure 3: Multibyte multibit DLSCA.

The proposed architecture is shown in Figure 3. The main branch is responsible for leakage processing and extraction of general features. After the main branch, several multibit models construct the subbranch for each target byte. The multibyte multibit DLSCA inherits the advantages of multibit DLSCA, namely, no assumptions on the 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 method reduces the model size, thus increasing the learning efficiency.

The importance of 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 6.3.1) as the crucial element that drives the success of the proposed 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 on 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+18, WJB20]. We present an ablation study on data augmentation in Section 6.3.1.

5 Experimental Results with a Simulated Dataset

To demonstrate the effectiveness of the proposed method, we present attack results with a simulated dataset comprising different leakages following Equation 12.

$$leakage = \begin{cases} Case1 : y \\ Case2 : \mathsf{hw}(y) \\ Case3 : \sum_{i=2}^{5} \mathsf{b}(y)[i] \\ Case4 : \prod_{i=0}^{3} \mathsf{b}(y)[i] + \prod_{i=4}^{7} \mathsf{b}(y)[i] \end{cases} + Z, \tag{12}$$

where y represents the target sensitive data $\mathsf{Sbox}(d_i \oplus k^*)$, $d_i \in \mathcal{D}$; d_i and k^* denote random plaintexts and a fixed key, respectively. 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 due to its non-parametric nature. The multibit model is constructed by building a Gaussian template on each bit separately; the bit predictions form the probability of a byte following Eq. (10). For comparison, we consider byte models for conventional template attacks and assume an attacker does not know the construction of actual leakage. Both the Hamming weight (HW) and Identity (ID) leakage models are considered in the benchmark.

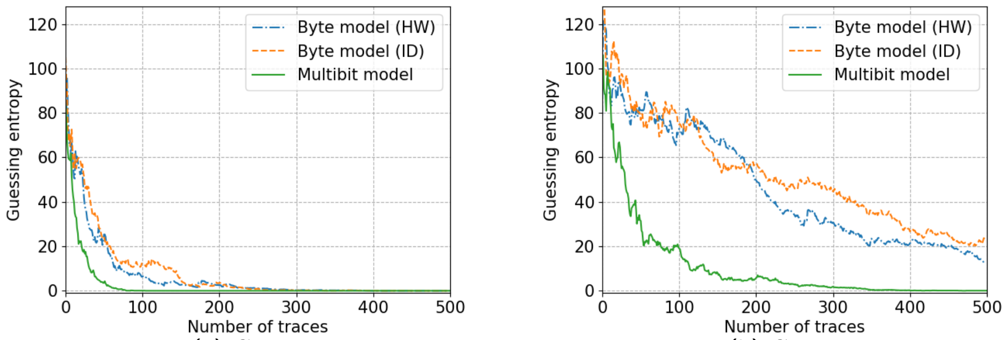
We first test the flexibility of the multibit model in generalizing to leakage that only leaks HW and ID (cases 1 and 2 in Eq. (12)). σ is set to 0.1 for the low noise setting. The accuracy of each bit is shown in Table 1. Aligned with the discussion in Section 4.2, the multibit model perfectly covers each leakage type. HW leakages lead to similar accuracy of each bit, indicating the equal contribution to the actual leakage; bit_7 of the ID leakage model has the highest accuracy; the rest follows descending order. These observations confirm the ability of the multibit model to adjust to different leakage models. Moreover, this result again confirms our claim that the commonly used leakage models, including HW and ID, have assumptions on the leakages of each bit.

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, σ is increased to 0.4 to simulate the realistic noise effect, increasing the attack difficulties. The attack result is shown in Figure 4. Multibit model outperforms the HW and ID leakage models, with the

Table 1: The accuracy of each bit when side-channel traces only have HW (case 1) or ID (case 2) leakages.

	bit_7	bit_6	bit_5	bit_4	bit_3	bit_2	bit_1	bit_0
HW	0.61	0.60	0.60	0.60	0.60	0.60	0.60	0.61
ID	0.93	0.60	0.55	0.51	0.50	0.51	0.50	0.50

leakage models becoming complicated (close to realistic) in case 3 and case 4 according to Eq. (12). The results thus clearly show the advantage of using a multibit model in adjusting and extracting complex features.



(a) Case 3. **(b)** Case 4.

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

6 Attacks on Publicly Available Datasets

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 multibit model and multibyte multibit model 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 of the multibit model. It is important to note that 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 could enhance attack performance. However, this approach also introduces more variables like model complexity and training effort, which could complicate 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 acceptable.

Therefore, given its excellent performance in various attack environments, we utilize a convolution neural network (CNN) based on [PWP22].[1] The network structure includes two convolution blocks, each with a convolution layer (kernel numbers: 4, 32; size: 40, 8;

[1]The deep learning models were implemented using Python 3.6, with the TensorFlow library version 2.6.0. Training algorithms were executed on an Nvidia GTX 1080TI GPU, managed by Slurm workload manager version 19.05.4.

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 $Selu$ for the layer activation, except for the final layer, which uses $Softmax$. The batch size is set at 512. When the multibyte multibit 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 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 6.3.2. To ensure a fair comparison, we apply data augmentation to all DL-based attack methods, achieved by adding a custom layer after the input layer that randomly shifts the leakage measurement within a pre-defined augmentation level of 5. We provide a detailed analysis of data augmentation in Section 6.3.1.

As outlined in Section 2.3, 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 key rank of zero. 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.

6.1 Datasets

Our experiments consider four datasets, consisting of measurements that are software targets protected with a Boolean masking countermeasure.

ASCAD_F. This dataset contains the measurements from an 8-bit AVR microcontroller running a masked AES-128 implementation [BPS+20].

ASCAD_R. This dataset uses the same measurement setup as ASCAD_F [BPS+20]. The difference is that ASCAD_R also provides traces with random keys (for the profiling phase), and it provides more sample points per trace. There are also more profiling and attack traces in ASCAD_R compared to ASCAD_F. However, 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.[2]

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].

For the eShard dataset, since it has already been trimmed, we forgo any additional pre-processing steps. The specific attack settings for these datasets are detailed in Table 2.

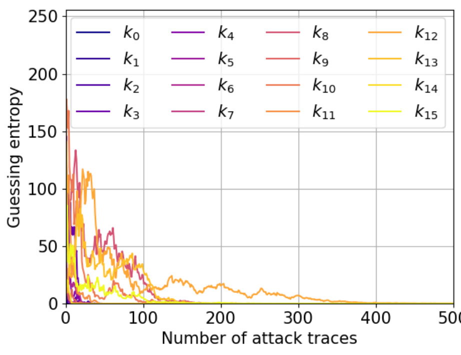
[2] https://chesctf.riscure.com/2018/news

Table 2: Summary of the tested datasets.

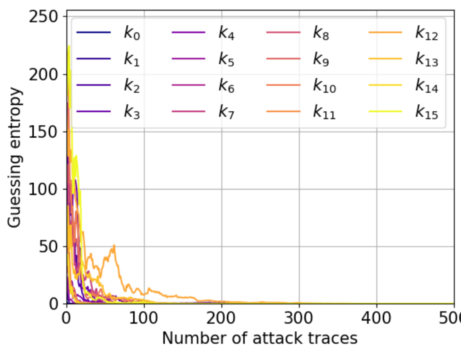
	ASCAD_F	ASCAD_R	CHES_CTF	eShard
Protection	Boolean masking			
Profiling traces	50 000	50 000	80 000	30 000
Attack traces	5 000			

6.2 Performance Evaluation

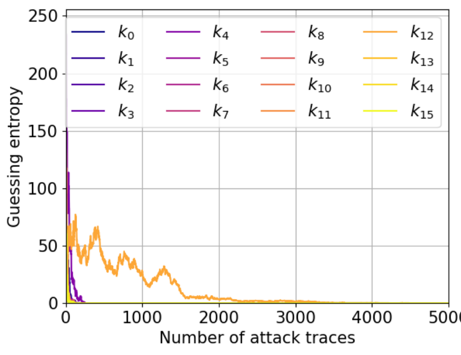
This section comprehensively examines the attack performance on the datasets under consideration. The guessing entropy (GE) using the multibyte-multibit model is depicted in Figure 5. To better illustrate the convergence of GE, different x-axis scales are employed. The results indicate that the multibyte-multibit model can successfully retrieve all key bytes with limited attack traces. According to our preliminary analysis, only the HW leakage model can break the CHES_CTF and eShard datasets. However, in our case, with a fixed deep learning architecture, the multibyte-multibit model efficiently breaks these datasets.



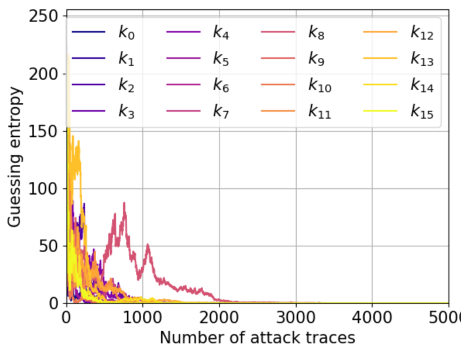
(a) ASCAD_F.



(b) ASCAD_R.



(c) CHES_CTF.



(d) eShard.

Figure 5: Guessing entropy for each dataset with multibyte multibit DLSCA.

Next, we benchmark the proposed methods with different pre-defined leakage models. The multibit model and multibyte multibit model are denoted by MMB and MB, respectively. Note that the MMB model attacks 16 sub-bytes simultaneously, while the rest targets each sub-byte in sequence. The attack performance is represented by T_{GE0}, 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 218 and 58 traces, respectively. 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 176 and 174 attack traces are required to reach a guessing entropy of zero. These observations confirm our earlier assertion in Section 5 that the proposed can effectively handle complex leakage models and masked intermediate data.

However, it should be noted that retrieving these key bytes might be accomplished through refined deep learning architectures or increased training effort [PWP22]. While hyperparameter tuning might yield improved results, it is time-consuming and often necessitates knowledge of the target key to assess the attack performance, making it more suitable for white-box evaluation [RWPP21, WPP22]. When considering real-world scenarios where a fixed model is used to attack different datasets (a common practice in the industry), MMB and MB prove to be superior choices due to their adaptability 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.

Table 3: T_{GE0} 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	0	3	6	19	8	3	21	9	134	14	75	17	218	122	8	68
MB	0	3	4	18	8	6	22	16	184	5	36	34	58	66	8	50
HW	3	3	493	281	418	371	534	589	x	627	1 401	791	x	x	201	x
ID	1	1	x	289	65	73	2 414	110	x	125	1 271	x	x	x	22	x
LSB	16	16	1 551	42	62	80	207	56	2 682	48	137	x	x	1 083	43	60
MSB	26	32	x	1 350	1 078	982	983	865	x	622	x	516	x	x	3 543	x

Table 4: T_{GE0} 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	2	4	22	75	27	8	14	34	93	13	73	14	176	109	12	47
MB	1	4	22	88	22	9	16	41	79	26	70	15	144	64	14	25
HW	4	5	1 522	1 087	356	438	1 383	1 159	3 481	807	1 080	823	x	x	414	x
ID	2	1	x	x	308	58	x	x	x	x	x	x	x	x	66	x
LSB	15	10	287	117	42	54	191	97	519	38	88	115	x	390	23	53
MSB	38	59	x	x	4 127	x	1 103	x	x	1 697	x	295	x	x	1 022	x

Table 5 and Table 6 show the attack results on the CHES_CTF and eShard datasets. Aligned with the previous two datasets, MMB and MB maintain outstanding performance on these two datasets. It is worth noting that MB 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), we conclude that multi-task learning helps generalize each task, leading to robust attack performance.

Table 5: T_{GE0} 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	88	95	57	99	226	75	47	42	113	54	49	45	3 210	49	58	34
MB	71	56	42	52	77	50	54	55	52	72	39	53	581	39	52	25
HW	2057	x	803	1 156	x	1 368	1 714	1 120	x	3 012	1 329	4 557	x	x	1 534	1 472
ID	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
LSB	508	811	139	394	386	500	575	227	265	481	438	266	x	532	580	370
MSB	606	485	1 359	x	2 039	3 163	757	x	x	812	2 256	326	x	x	498	x

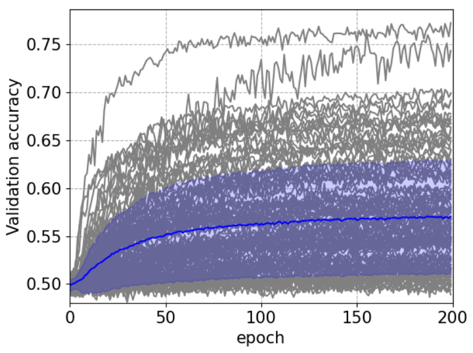
Table 6: T_{GE0} of each subkey for the eShard dataset.

	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	118	503	857	247	1102	690	487	948	2161	339	105	952	1502	540	416	469
MB	1 172	941	907	298	x	1 705	1 893	1 084	x	875	2 419	1 458	x	1 268	1 879	1 303
HW	1 689	2 625	1 278	921	x	975	972	782	x	1 741	1 369	1 050	x	1 207	1 556	1 175
ID	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
LSB	x	x	4 591	2 320	x	x	x	4 639	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

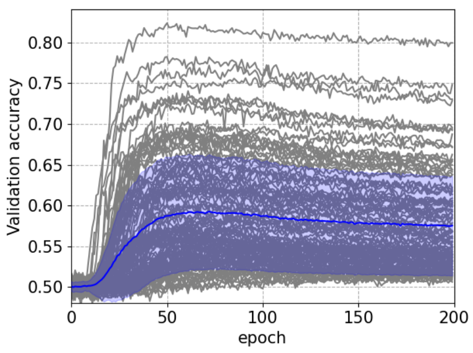
Besides the key recovery, the proposed method provides insight into leakage assessment. Previous results [WPP22, PWP22] show that the CHES_CTF dataset is only breakable via the HW leakage model, which our results confirm. None of the key bytes can be retrieved via the ID leakage model. Interestingly, the MSB leakage model also leads to mediocre performances. Recall in Section 5, when the simulated data has only ID leakage, MSB is the most significant contributor to actual leakages (see Table 1). Based on the attack results, MSB seems to contain limited leakage information for these two datasets, potentially leading to the failure of the ID leakage model.

To provide a more comprehensive view of the learning process, we illustrate the validation accuracy for each bit (256 in total) of the multibyte multibit model in Figure 6. The mean and standard deviation are represented by the blue line and shaded blue areas, respectively.

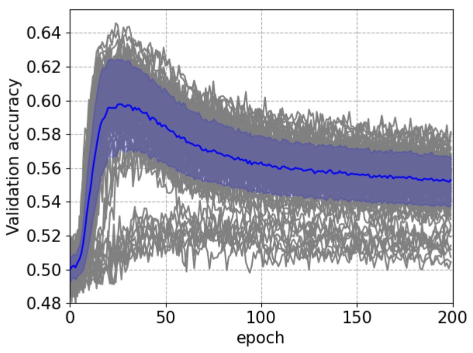
Initially, for ASCAD_F and ASCAD_R, there is a sharp rise in the standard deviation, indicating a broader distribution of bit validation accuracy. The top five bits yielding the highest accuracy are either b_7 (MSB) or b_6 of a byte. This observation aligns with the results displayed in Tables 3 and 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 5 and also corroborates the earlier discussion about the mediocre performance of the ID model.



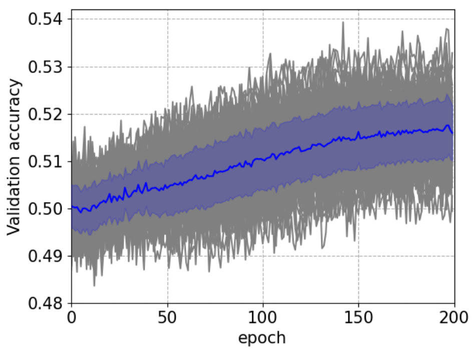
(a) ASCAD_F.



(b) ASCAD_R.



(c) CHES_CTF.



(d) eShard.

Figure 6: Validation bit accuracy for each dataset.

It is clear that the ASCAD_R and CHES_CTF datasets are prone to overfitting, a phenomenon we explore in detail in Section 6.3.2. 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 multibyte multibit model is generalized to features that all tasks can share.

6.3 Ablation Studies

In this section, we explore the influence of various hyperparameters on the multibit model performance. Instead of focusing on one sub-byte, the multibyte multibit model is used for benchmarking, and all sub-bytes are considered. For a fair comparison, the attack performance on 16 sub-bytes is averaged to understand the influence of hyperparameter changes better.

6.3.1 Data Augmentation

As discussed in Section 4.3, the role of data augmentation is crucial for the multibit model. Consequently, we conduct an ablation 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 Table 7.

When the level of data augmentation is set to zero, the multibit model fails to break the eShard dataset. However, a significant performance improvement is noted with the

Table 7: Ablation study on data augmentation (DA).

	DA-0	DA-5	DA-10	DA-20
ASCAD_F	50	45	242	1 653
ASCAD_R	136	45	87	937
CHES_CTF	273	269	427	327
eShard	x	715	394	449

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 multibit model. 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.

6.3.2 Training Epoch

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 6 clearly shows that ASCAD_R and CHES_CTF suffer from overfitting when training with 200 epochs. Table 8 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. As for the eShard dataset, there is a consistent decrease in key rank value, in line with the observation in Figure 6d.

Table 8: Study on the influence of the training epoch.

	EP-50	EP-100	EP-200	EP-300
ASCAD_F	136	43	45	49
ASCAD_R	46	39	44	45
CHES_CTF	118	136	269	285
eShard	3 888	1 080	714	760

Several strategies can be employed to mitigate overfitting. Data augmentation, as discussed in Section 6.3.1, 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. Some studies [KPH$^+$19, WPP22] suggest that validation accuracy may be unreliable when working with pre-defined leakage models. However, our results demonstrate that it is a robust metric for assessing a model's performance for the multibit model.

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 multibyte multibit approach, making it a reliable profiling solution across varying attack scenarios.

7 Conclusions and Future Work

This paper introduces a novel multibit model that learns each bit separately. Unlike conventional profiling attacks, our method does not assume 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 to build on this work. 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, the mainstream microcontroller adopts a 32-bit or even wider bus width. Since the proposed method could prevent a byte from learning each bit, examining the effectiveness of this method in 32-bit implementations is of interest, i.e., learning 32-bit simultaneously with a multibit model.

References

[BPS$^+$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.

[Car97] Rich Caruana. Multitask learning. *Machine learning*, 28:41–75, 1997.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems – CHES 2017*, pages 45–68, Cham, 2017. Springer International Publishing.

[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.

[CRR02a] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In *CHES*, volume 2523 of *LNCS*, pages 13–28. Springer, August 2002. San Francisco Bay (Redwood City), USA.

[CRR02b] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[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.

[HGM+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.

[HK18] Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *CoRR*, abs/1806.03852, 2018.

[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.

[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.

[KPH+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.

[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.

[LPB+15] Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, and François-Xavier Standaert. Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 20–33. Springer, 2015.

[LZC+21] Xiangjun Lu, Chi Zhang, Pei Cao, Dawu Gu, and Haining Lu. Pay attention to raw traces: A deep learning architecture for end-to-end profiling attacks. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 235–274, 2021.

[Mag20] Houssem Maghrebi. Deep learning based side-channel attack: a new profiling methodology based on multi-label classification. *Cryptology ePrint Archive*, 2020.

[MPP16] Houssem 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.

[MS21] Loïc Masure and Rémi Strullu. Side channel analysis against the anssi's protected aes implementation on arm. *Cryptology ePrint Archive*, 2021.

[PCP20] Guilherme Perin, Lukasz Chmielewski, and Stjepan Picek. Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2020(4):337–364, Aug. 2020.

[PHG17] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. Template attack versus bayes classifier. *Journal of Cryptographic Engineering*, 7(4):343–351, September 2017.

[PPM$^+$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.

[PSK$^+$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.

[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.

[RB22] Azade Rezaeezade and Lejla Batina. Regularizers to the rescue: Fighting overfitting in deep learning-based side-channel analysis. *IACR Cryptol. ePrint Arch.*, page 1737, 2022.

[Rud17] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[RWPP21] Jorai Rijsdijk, 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.

[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.

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A stochastic model for differential side channel cryptanalysis. In Josyula R. Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2005*, pages 30–46, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[SMY09] François-Xavier Standaert, Tal G. Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, pages 443–461, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[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.

[VTM23] Aurélien Vasselle, Hugues Thiebeauld, and Philippe Maurine. Spatial dependency analysis to extract information from side-channel mixtures: extended version. *Journal of Cryptographic Engineering*, pages 1–17, 2023.

[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.

[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.

[WPP20] Lichao Wu, Guilherme Perin, and Stjepan Picek. I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis. Cryptology ePrint Archive, Report 2020/1293, 2020. https://eprint.iacr.org/2020/1293.

[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.

[WWK+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.

[ZBC+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.

[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.

[ZXF+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.

[ZY21] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.

[ZZN+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.