An End-to-end Plaintext-based Side-channel Collision Attack without Trace Segmentation

Lichao Wu[1], Sébastien Tiran[3], Guilherme Perin[2] and Stjepan Picek[1]

[1] Radboud University, The Netherlands
[2] Leiden University, The Netherlands
[3] Independent Researcher

Abstract. Side-channel Collision Attacks (SCCA) constitute a subset of non-profiling attacks that exploit information dependency leaked during cryptographic operations. Unlike traditional collision attacks, which seek instances where two different inputs to a cryptographic algorithm yield identical outputs, SCCAs specifically target the internal state, where identical outputs are more likely. In CHES 2023, Staib et al. presented a Deep Learning-based SCCA (DL-SCCA), which enhanced the attack performance while decreasing the required effort for leakage preprocessing. Nevertheless, this method inherits the conventional SCCA's limitations, as it operates on trace segments reflecting the target operation explicitly, leading to issues such as portability and low tolerance to errors.

This paper introduces an end-to-end plaintext-based SCCA to address these challenges. We leverage the bijective relationship between plaintext and secret data to label the leakage measurement with known information, then learn plaintext-based profiling models to depict leakages from varying operations. By comparing the leakage representations produced by the profiling model, an adversary can reveal the key difference. As an end-to-end approach, we propose an error correction scheme to rectify false predictions. Experimental results indicate our approach significantly surpasses DL-SCCA in terms of attack performance (e.g., success rate increased from 53% to 100%) and computational complexity (training time reduced from approximately 2 hours to 10 minutes). These findings underscore our method's effectiveness and practicality in real-world attack scenarios.

Keywords: Side-channel Analysis · Side-channel Collision Attack · Deep Learning.

1 Introduction

Side-channel analysis (SCA) of symmetric-key cryptography implementations is typically divided into non-profiling and profiling attacks depending on the availability of a second (clone) device identical (similar) to the device under attack (target). Non-profiling attacks operate without a clone device. The adversary gathers measurements encapsulating secret information, conducting statistical analysis to deduce the secrets. Conversely, profiling attacks assume the adversary has unrestricted access to an identical device. Using this device, the adversary studies and comprehends the device's side-channel behavior, utilizing this understanding to extract secret information from the target device.

Side-channel collision attacks (SCCA), a type of non-profiling SCA, were introduced in [SWP03], using SCA data to detect collisions in the internal state of an AES implementation. This strategy benefits from not depending on any leakage model. Recently, Staib et al. incorporated deep learning into SCCA by learning the plaintext [SM23]. The technique was denoted as a deep learning side-channel collision attack (DL-SCCA). Similar to SCCA, DL-SCCA preprocesses the leakage traces into segments representing the processing of each

targeted intermediate data (i.e.,Sbox output), characterizing a trace segment, and tries to recover the key difference by comparing with a different trace segmentation with deep learning. Despite the performance improvement, the limitations of conventional SCCA are inherited by DL-SCCA, constraining its application to practical attack scenarios. The main limitations are:

- **Trace Segmentation.** Trace segmentation is the most intricate procedure for SCCA, including DL-SCCA. Indeed, the traces segment is analyzed to extract secret information; imprecise segmentation could directly impact attack performance. DL-SCCA employs a multi-stage traces segmentation method involving deep learning-based sensitivity analysis and visual inspection. Regrettably, this method is time-consuming and dependent on human expertise (e.g., model tuning and finding leakages). Errors may be introduced at each trace segmentation stage, and the generality of this method is questionable.
- **Portability Problem.** DL-SCCA employs a model trained on one trace segmentation (representing the processing of a single intermediate data) to attack a different trace segment, introducing the well-known SCA portability problem [BCH+20]. Specifically, the method's effectiveness relies on 1) Temporal Consistency: the timing of leakage features between two trace segments must be well aligned for the trained model to extract information from features at the identical location; 2) Spatial Consistency: the leakage features between two traces segments should be alike. Any discrepancy between features could diminish attack performance.
- **Error Tolerance.** The attack performance of DL-SCCA relies solely on the DL model trained on a reference trace segment. In realistic attack scenarios where the adversary does not know the key, blindly depending on one model can be problematic, as there is no mechanism to confirm the correctness of the predicted key relationship.

This paper introduces a novel plaintext-based side-channel collision attack (PSCCA) to circumvent the above-mentioned limitations. Our main contributions are:

1. We propose a novel approach for SCCA based on the learning from plaintexts, which eliminates the need for trace segmentation, operating directly on the raw traces.
2. We present an error correction method that can rectify incorrectly predicted key relationships based on the prediction of the key relationship of other bytes.
3. We employ multi-task learning to expedite the training process while maintaining attack performance.
4. We provide a comprehensive experimental analysis showcasing the exceptional attack performance of the proposed method in terms of key difference rank and success rate. Our method outperforms the state-of-the-art DL-SCCA even without error correction. When applied, the success rate is significantly increased.
5. We investigate several hyperparameters relevant to our attack: data augmentation and batch size. In the meantime, we discuss the efficiency of error correction in different settings.

The rest of this paper is organized as follows. Section 2 provides the necessary background information. Section 3 discusses related works. Section 4 details the threat model, then performs a theoretical analysis of the method and presents the attack scheme. Finally, a case study is given. In Section 6, we offer experimental results. Section 7 presents the evaluation of some hyperparameters. Finally, Section 8 concludes the paper and discusses potential future research directions.

2 Background

This section starts by introducing the notation we follow. Afterward, the relevant information about the side-channel analysis, collision attack, evaluation metrics, and considered datasets are introduced.

2.1 Notation

We denote sets using calligraphic letters such as $\mathcal{X}$, while the related uppercase letters X represent random variables and random vectors $\mathbf{X}$ that are defined over the domain of $\mathcal{X}$. The corresponding lowercase letters x and $\mathbf{x}$ symbolize realizations of X and $\mathbf{X}$, respectively. Furthermore, functions, represented as f, are given in a sans-serif font.

The symbol k represents a candidate key byte from a key space $\mathcal{K}$. Δ denotes the xor key difference between two different key bytes. A dataset $\mathbf{T}$ comprises traces, symbolized as $\mathbf{t}_i$, which have corresponding associations with plaintext/ciphertext pairs $\mathbf{d}_i \in \mathcal{D}$ and keys $\mathbf{k}_i$, or $k_{i,j}$ and $d_{i,j}$ when partial key recovery on byte j is under consideration. Since we focus on byte processing, k_i and d_i represent a byte for simplicity. This work considers a fixed key scenario where $\mathbf{t}_i$ has the same $\mathbf{k}_i$, utilizing byte vector notation exclusively in equations.

2.2 Side-channel Analysis

As briefly mentioned in the introduction section, side-channel analysis (SCA) can be broadly classified into two categories: profiling SCA and non-profiling SCA, based on the availability of a fully-controlled cloned device.

Profiling side-channel attacks aim to connect a set of inputs (such as side-channel traces) to outputs (like a probability vector of key guesses). Profiling attacks have two stages. In the profiling stage, the adversary builds a profiling model, represented as $\mathsf{f}_{\boldsymbol{\theta}}^{M}$, which is determined by a leakage model M and a group of learning parameters $\boldsymbol{\theta}$. In this paper, the terms $\mathsf{f}_{\boldsymbol{\theta}}^{M}$ and $\mathsf{f}_{\boldsymbol{\theta}}$ are used interchangeably. This model maps inputs (side-channel measurements) to outputs (classes obtained by evaluating the leakage model during a sensitive operation) using a set of N profiling traces. Then, in the attack stage, the trained model processes each attack trace $\mathbf{t}_i$ and produces a probability vector $\mathbf{p}_j$, reflecting the likelihood of the related leakage value or label j. The adversary chooses the best key candidate based on this probability vector. If the adversary can build an effective profiling model, only a few measurements from the target device could be enough to break its security. Profiling attacks include methods like the template attack [CRR02], stochastic models [SLP05], and supervised machine learning-based attacks [HGM+11, MPP16, PHJ+17].

Non-profiling attacks tend to assume less powerful adversaries who do not have access to a cloned device. An adversary collects a series of traces created during the encryption of different plaintexts. Then, the adversary can guess part of the key by analyzing the correlation between the key-related intermediate values and the leakage measurements. This strategy often follows a 'divide-and-conquer' approach. Initially, the adversary groups the traces based on the predicted intermediate value tied to the current key guess. If the groups show clear differences (how 'difference' is defined depends on the attack method), it suggests that the current key guess is likely correct. Non-profiling attacks may need many measurements (possibly millions) to reveal sensitive information. Examples of non-profiling attacks include simple power analysis (SPA), differential power analysis (DPA) [KJJ99a], correlation power analysis (CPA) [BCO04], and some machine learning-based attacks [Tim19, DLH+22, WPP23a]. It is also worth mentioning that side-channel collision attack [SWP03, Bog07] is considered a non-profiling SCA, but it follows a slightly different strategy, which will be discussed in the following section.

2.3 Side-channel Collision Attack

Side-channel Collision Attacks (SCCA) form a category of non-profiling attacks that exploit data inter-dependence leaked during cryptographic procedures. Unlike traditional collision attacks that leverage instances where two different inputs into a cryptographic algorithm lead to an identical output, SCCA specifically targets the internal state, which is more likely to coincide between two cryptographic operations.

In SCCA, an adversary monitors the side-channel information while the system processes different inputs. The adversary then searches for repeated leakage patterns signifying a collision event. When a collision is detected, the adversary can use this information to infer insights about the inter-dependencies of different key sections or the algorithm's internal state. For instance, let us consider the SubBytes operation of the Advanced Encryption Standard (AES) with the same substitution box (Sbox). The same data has been processed if two different Sbox operations lead to an identical side-channel pattern. Since the Sbox operation is bijective (that is, it establishes a one-to-one correspondence between two sets), we obtain the following equations:

$$\begin{aligned} &\mathsf{Sbox}(k_i \oplus p_i) = \mathsf{Sbox}(k_j \oplus p_j) \\ &=> \ k_i \oplus p_i = k_j \oplus p_j \\ &=> \ k_i \oplus k_j = p_i \oplus p_j. \end{aligned} \tag{1}$$

We represent the result of $k_i \oplus k_j$ as $\Delta_{i,j}$. Unlike other SCA techniques focusing on key recovery, SCCA strives to expose the linear difference between various keys. An adversary guesses one subkey to achieve full key recovery, as the rest of the key can then be calculated based on this linear difference. Since all subkeys can be recovered by correctly guessing one of the subkey bytes, the remaining keyspace shrinks to 2^8.

2.4 Evaluating the Attack Performance

This work employs the maximum log-likelihood distinguisher to obtain a cumulative sum $S(\Delta)$ for each Δ candidate:

$$S(\Delta) = \sum_{i=1}^{Q} \log(\mathbf{p}_{i,j}(\Delta)), \ \Delta \in \mathcal{K}, \tag{2}$$

where $\mathbf{p}_{i,j}(\Delta)$ denotes the probability of each $\Delta \in \mathcal{K}$ being chosen as the correct $\Delta_{i,j}$; Q represents the number of attack traces. The result of an attack is a Δ guessing vector $\mathbf{g} = [g_1, g_2, \ldots, g_{|\mathcal{K}|}]$, which is computed for Q traces in the attack phase. This vector orders the Δ candidates in descending likelihood, with g_1 being the most probable candidate and $g_{|\mathcal{K}|}$ the least probable one. The position of $\Delta_{i,j}$ within the Δ guessing vector $\mathbf{g}$ is utilized to estimate the effort needed to reveal the secret key difference, and the corresponding rank is referred to as the Δ rank. In the experimental section, each attack is conducted ten times, and we calculate the success rate, representing the percentage of successful attacks out of ten.

Since the side-channel collision attack engages multiple sub bytes, we are also interested in the overall efficiency of the attack in retrieving $\Delta_{i,j}$ with different sub bytes. Therefore, we also assess the overall success rate, indicating the percentage of successful recoveries of the key difference among all tested $\Delta_{i,j}$.

2.5 Evaluated Datasets

The experiments in this study involve the following datasets, all of which employ first-order Boolean masking countermeasures. The raw side-channel measurements for each

dataset are as follows. It is worth noting that handling such large intervals of side-channel measurements can be time-consuming. To address this, we adopt the resampling technique with a resampling window 80, aligned with the approach outlined in [PWP22].

ASCAD_F. This dataset comprises measurements from an 8-bit AVR microcontroller executing a masked AES-128 implementation [BPS+20]. It consists of 60,000 leakage traces.

ASCAD_R. Similar to ASCAD_F, this dataset employs the same measurement setup [BPS+20]. However, ASCAD_R also includes traces with random keys. For our experiments, we solely utilize the leakage traces associated with a fixed key, commonly used for testing with DLSCA. We employ 60,000 traces for the attack.

CHES_CTF [1]**.** This dataset comprises the CHES Capture-the-flag (CTF) AES-128 measurements released in 2018 for the Conference on Cryptographic Hardware and Embedded Systems (CHES). The traces correspond to masked AES-128 encryption performed on a 32-bit STM microcontroller. Our experiments consider 40,000 traces for the Δ recovery.

3 Related Works

The Side-channel Analysis community has been conducting research on profiling attacks for over two decades. The first profiling attack was introduced by Chari et al., which established the template attack [CRR02]. This attack is the most powerful one from an information-theoretic standpoint. However, it makes assumptions difficult to satisfy, such as an unlimited number of profiling traces and noise following a Gaussian distribution [LPB+15]. Other "classic" profiling attacks include the stochastic model [SLP05]. The field has also seen a rise in the application of machine learning techniques, initially employing simpler methods like random forest [LMBM13] and support vector machines [HGM+11], as the most common examples. While simple (compared to, e.g., deep learning), these techniques have already shown performance surpassing template attacks or stochastic models. Since 2016, significant attention is given to deep learning techniques for profiling SCA. The first work utilizing convolutional neural networks (CNNs) already established the potential of deep learning in breaking various targets [MPP16]. The following years provided advancements in building methodologies for CNNs [ZBHV19, WAGP20], advanced hyperparameter tuning strategies [WPP22b, RWPP21], data augmentation [CDP17, MBPK22], ensembles [PCP20], or custom loss functions [ZZN+20, KWPP22]. Current research has made considerable advancements, breaking datasets perceived as challenging a few years ago, even with a single measurement [LZC+21, WPP22a, PWP22]. While deep learning-based SCA provides excellent results, multiple challenges still need to be addressed [PPM+23]. One of the main ones is how to move away from the “classical” profiling paradigm.

However, profiling attacks do not apply to black-box adversaries without access to a profiling device. In these cases, non-profiling SCA becomes relevant. The differential power analysis, a classical non-profiling SCA, is widely adopted in academia and industry [KJJ99b]. The concept of using deep learning in non-profiling SCA was first proposed by B. Timon [Tim19], which involved training multiple neural networks corresponding to different key guesses. Later, advancements by Hoang et al. introduced a multi-output classification technique for non-profiling SCA, which significantly improved the efficiency and effectiveness of the attack [HDD22]. Recently, multi-output classification (MOC) and multi-output regression (MOR) models for non-profiling SCA were investigated by Do et al. [DLH+22]. Finally, Wu et al. employed plaintext distribution to perform the key recovery [WPP23a].

Side-channel collision attacks (SCCA) have received less attention than profiling and non-profiling attacks. The concept of SCCA was first introduced to use SCA data to

detect collisions in the internal state of an AES implementation [SWP03]. The advantage of this strategy is its independence from any leakage model. Initially limited to single collision, these attacks were refined to include all possible collisions in the measurement set [MME10, MS16]. Furthermore, SCCAs can bypass countermeasures previously deemed 'impossible' through conventional profiling SCA [WPP23b]. The integration of deep learning into SCCA, known as Deep Learning Side-Channel Collision Attack (DL-SCCA), was recently proposed by Staib et al. [SM23]. This methodology designates a target byte as a reference, employing corresponding trace segments and plaintexts to train a deep learning model. The trained model is subsequently applied to a distinct trace segment to predict the key relationship. However, identifying the trace segment in this method is performed through deep learning-based sensitivity analysis and visual inspection, which could be prone to inaccuracies and potentially introduce new challenges to the attack process, as detailed in the next section.

4 Plaintext-based Side-channel Collision Attack

This section first introduces the threat model we follow. Afterward, the theoretical evaluation and case study of the plaintext-based side-channel collision attack are presented.

4.1 Threat Model

We assume the adversary has access to a device running the target cipher and possesses a fixed yet undisclosed key. The adversary can instruct the device to perform encryption or decryption operations, but it is assumed that they can only observe the employed plaintext or ciphertext and cannot manipulate their values.

The adversary lacks information about the hardware implementation, the countermeasure settings, and the source code. To execute attacks, the adversary captures multiple side-channel leakages using a high-speed oscilloscope and subsequently analyzes these leakage traces in conjunction with plaintexts and/or ciphertexts.

4.2 The Curse of Trace Segmentation

Consider a leaking device using a fixed secret key denoted by k. Cryptographic processes include sub-key k_i and a string of data d_i, considered as an n-bit word. Typically, $n = 8$ due to the byte-oriented nature of AES, which is widely examined in related works [ZBHV19, WPP22a, SM23]. The side-channel leakage can be modeled by a leakage function ψ applied to intermediate data $\mathsf{I}(k_i, d_i)$ plus some additive noise $Z_i \sim \mathcal{N}(0, \sigma^2)$, shown in Eq. (3).

$$\mathbf{t_i} = \psi(\mathsf{I}(k_i,\ d_i)) + Z_i,\ d_i \in \mathcal{D}, \tag{3}$$

where $\mathbf{t_i}$ represent the leakage traces. The purpose of a side-channel collision attack (SCCA) is to enable an adversary to identify a key difference Δ, which results in the smallest difference between two trace segments $\mathbf{t_i}$ and $\mathbf{t_j}$, denoting the leakage traces from two identical operations processing different data within one cryptographic function:

$$\Delta_{i,j} = \arg\min_{\Delta}\ \psi(\mathsf{I}(k_i, d_i)) - \psi(\mathsf{I}(k_j, d_j)) + (Z_i - Z_j),\ d_j = d_i \oplus \Delta. \tag{4}$$

In this context, k_i and k_j are fixed, unknown values representing the secret sub-key of two trace segments. As the collision between $\mathsf{I}(k_i, d_i)$, and $\mathsf{I}(k_j, d_j)$ is not consistently present in a single cryptographic function, an adversary must segment $\mathbf{t_i}$ and $\mathbf{t_j}$ from different traces such that $d_j = d_i \oplus \Delta$, then making comparison of trace segments from two operations. Nonetheless, this process can be problematic from a realistic point of view.

- **Identification of target operations.** SCCA is categorized as a non-profiled attack, as no cloned device for leakage characterization exists. Thus, an adversary needs to detect the leakage, for instance, via simple power analysis. The identification of leakage operation typically necessitates distinct leakage traces. Considering that a hardware AES operation can be completed in microseconds, the corresponding side-channel leakage may only have negligible patterns (e.g., a small power peak representing all AES rounds), significantly complicating the trace segmentation process. Concurrently, identifying the leakage operation could require an extra understanding of the source code or the ability to alter the source code to place separators before and after the target process. Although these requirements might be met in a white-box evaluation, a stronger attack assumption is required, reducing the chance of a successful attack in a black-box setting.
- **Feature alignment and noise.** A fundamental assumption of SCCA is the perfect alignment of leakage features from two trace segments. Ideally, the target operation's pattern should be unique and noise-free, facilitating easy alignment for an adversary. However, even hardware with a naive cryptographic implementation can be subject to environmental noise and clock jitters. One might suggest that leakage randomness due to environmental noise could be compensated with a low-pass filter or averaging techniques. Nevertheless, clock jitter complicates this process and introduces variability in execution time. To counter this, an adversary could apply elastic alignment [vWWB11] or align on a specific feature subset. Still, these methods may ignore features pertinent to the target operations, potentially reducing attack effectiveness.

Even though three limitations are discussed in the introduction, trace segmentation is the primary hurdle that hinders the broader application of SCCA-related methods. Simply operating on raw leakage features, such as the latest DL-SCCA [SM23], may not suffice to overcome this obstacle. An ideal approach should autonomously identify the target operation, extract relevant information, and generate a *representation* of the leakages. Consequently, an adversary could compare these representations with various Δ guesses to determine the correct value.

4.3 Learning from Plaintext

Plaintext learning has demonstrated its effectiveness in recent works [WPP23a, SM23]. Specifically, in the context of profiling SCA, a profiling model $\mathsf{f}_{\boldsymbol{\theta}}$ encapsulates the relationship between the input leakage measurement and output intermediate data, as processed by a leakage model M. When an attack trace is fed into $\mathsf{f}_{\boldsymbol{\theta}}$, the output is a probability vector corresponding to all potential intermediate data:

$$\mathbf{p}_i(y) = \mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t_i}), \tag{5}$$

where $y = \mathsf{M}(\mathsf{I}(k_i, d_i)),\ d_i \in \mathcal{D}$. Since we focus on a specific sub-key byte, k_i and k are used interchangeably. In SCCA, as the adversary is unaware of the key in use, he cannot leverage $\mathsf{I}(k, d_i)$ to estimate $\boldsymbol{\theta}$. However, if k remains constant across all leakage traces for SCCA, the label $\mathsf{I}(k, d_i)$ and d_i would satisfy:

$$d_i \mapsto \mathsf{I}(k, d_i). \tag{6}$$

The bijectivity between the label $\mathsf{I}(k, d_i)$ and d_i depends on the choice of I. For example, Sbox output is often employed as a label function (and intermediate data) for AES attacks. In such a case, given a fixed key k_i, d_i and $\mathsf{Sbox}(d_i \oplus k_i)$ are bijective. If Eq. (6) is valid, d_i and $\mathsf{I}(k, d_i)$ can be interchangeably mapped using mapping functions map parameterized by k:

$$\mathsf{I}(k, d_i) = \mathsf{map}_k(d_i). \tag{7}$$

Let $\mathbf{p}(\mathsf{I}(k, d_i)|\mathbf{t_i})$ be the probability of $\mathsf{I}(k, d_i)$ given $\mathbf{t_i}$. Following Eq. (7), it can be represented by $\mathbf{p}(\mathsf{map}_k(di)|\mathbf{t_i})$. As map is a fixed function that does not influence the profiling process, in the context of profiling models, we obtain:

$$\mathsf{f}_{\boldsymbol{\theta}_d}(\mathbf{t_i}) = \mathsf{map}'_k(\mathsf{f}_{\boldsymbol{\theta}}(\mathbf{t_i})), \tag{8}$$

where $f_{\boldsymbol{\theta}_d}$ refers to the profiling model trained with plaintexts, and map'_k is a fixed function that transforms the probability of input $\mathsf{I}(k, d)$ to its corresponding output d, for all $d \in \mathcal{D}$. Given their bijectivity, the same leakage features are learned when profiling with d_i or $\mathsf{I}(k, di)$. As a result, the models based on these features, $f_{\boldsymbol{\theta}_d}$ and $\mathsf{map}_k(\mathsf{f}_{\boldsymbol{\theta}})$, can be regarded as equivalent.

Eq. (8) indicates that a profiling model can be established based on plaintext to extract leakage information. No trace segmentation is required in this process. The next section will introduce a method that compares the plaintext prediction outputs from two cryptographic operations to reveal the key difference Δ.

4.4 Plaintext-based Collision

Given an input trace $\mathbf{t_i}$, $f_{\boldsymbol{\theta}_d}$ generates a prediction vector $\mathbf{p}(d|\mathbf{t_i}),\ d \in \mathcal{D}$, which represents the probability of all possible plaintexts. We argue that the distribution of the $\mathbf{p}(d|\mathbf{t_i})$ correlates with the key. To demonstrate this, let us consider an ideal case in leakage features, which we assume follow a Gaussian distribution, with the mean value for each class linearly correlated with their actual labels. In this context, a probability density function can represent the conditional probability of a label y given a correct label y_i:

$$\mathbf{p}(y|\mathbf{t_i}) = \mathbf{p}(y|y_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a|y_i - y|}{\sigma^2})},\ y \in \mathcal{Y}, \tag{9}$$

where $|\cdot|$ represents the squared Euclidean distance between two variables, a denotes the linear correlation coefficient, and σ represents the variance of the leakage features corresponding to y_i. Note each $\mathbf{t_i}$ associates with a unique y_i, thus $\mathbf{p}(y|\mathbf{t_i}) = \mathbf{p}(y|y_i)$ applies.

As y_i is parameterized by d_i and k_i. Given a fixed k_i, we can extend Eq. (9) to calculate $\mathbf{p}(d|d_i)$:

$$\mathbf{p}(d|d_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a|\mathsf{M}(\mathsf{I}(k_i, d_i)) - \mathsf{M}(\mathsf{I}(k_i, d))|}{\sigma^2})},\ d \in \mathcal{D}. \tag{10}$$

In the SCCA context, an adversary cannot estimate σ as the key k_i is unknown. We thus assume it to be a constant, and then $\mathbf{p}(d|d_i)$ is solely determined by k_i. This connection can be leveraged to compute the key difference $\Delta_{i,j}$. Consider another intermediate data $\mathsf{I}(k_j, d_j)$. If $d_i = d_j$ and $\Delta_{i,j} = k_i \oplus k_j$, then:

$$\mathbf{p}(d|d_j) = \mathbf{p}(d \oplus \Delta_{i,j}|d_i \oplus \Delta_{i,j}),\ d \in \mathcal{D}. \tag{11}$$

Proof. Following Eq. (10), $\mathbf{p}(d|d_j)$ can be represented by:

$$\mathbf{p}(d|d_j) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a\|\mathsf{M}(\mathsf{I}(k_j, d_j)) - \mathsf{M}(\mathsf{I}(k_i, d))\|}{\sigma^2})},\ d \in \mathcal{D}. \tag{12}$$

Since $\Delta = k_i \oplus k_j$ and $d_i = d_j$, $\mathbf{p}(d|d_j)$ can be rewritten to:

$$\begin{aligned}
\mathbf{p}(d|d_j) &= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a\|\mathsf{M}(\mathsf{I}(k_i \oplus \Delta_{i,j}, d_i)) - \mathsf{M}(\mathsf{I}(k_i \oplus \Delta_{i,j}, d))\|}{\sigma^2})} \\
&= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a\|\mathsf{M}(\mathsf{I}(k_i, d_i \oplus \Delta_{i,j})) - \mathsf{M}(\mathsf{I}(k_i, d \oplus \Delta_{i,j}))\|}{\sigma^2})} \\
&= \mathbf{p}(d \oplus \Delta_{i,j}|d_i \oplus \Delta_{i,j}),\ d \in \mathcal{D}.
\end{aligned} \tag{13}$$

□

Eq. (11) encapsulates the core idea of this paper, denoting plaintext-based collision. If the plaintext used in two separate intermediate data is the same, the plaintext prediction vector from one operation can be converted to that of the other using the correct Δ guess. The correct $\Delta_{i,j}$ can be retrieved via Eq. (14).

$$\Delta_{i,j} = \underset{\Delta}{\text{argmax}}\ \mathsf{corr}(\mathbf{p}(d|d_j),\ \mathbf{p}(d \oplus \Delta | d_i \oplus \Delta)),\ d_i = d_j,\ \Delta \in \mathcal{K}, \tag{14}$$

where corr represents the Spearman correlation [2] [HK11] that evaluates the monotonic relationship between two inputs. In this context, we are not solely interested in plaintext prediction (as they are already known) but in the probability distribution of each plaintext. For this reason, the Spearman correlation is best suited for this task.

4.5 Error Correction

During attacks utilizing plaintext-based SCCA, one may encounter incorrect prediction outcomes due to, for instance, complicated leakage features or insufficient attack traces. Fortunately, the prediction differs from profiling and non-profiling methods, which predict each key byte individually. Indeed, SCCA predictions rely on the key difference Δ, enabling an adversary to correct prediction outcomes with the aid of predictions from other bytes. Specifically, considering $\Delta_{i,j} = k_i \oplus k_j$, the following relationship holds:

$$\begin{aligned} \Delta_{i,j} &= (k_i \oplus k_a) \oplus (k_j \oplus k_a) \\ &= \Delta_{i,a} \oplus \Delta_{j,a},\ a \notin \{i, j\}. \end{aligned} \tag{15}$$

Here, k_a denotes an arbitrary key byte, and $\Delta_{i,j}$ can be expressed as the xor of two key differences $\Delta_{i,a}$ and $\Delta_{j,a}$ that involve an additional byte. In cases where $\Delta_{i,j}$ is mispredicted, while $\Delta_{i,a}$ and $\Delta_{j,a}$ are accurate, we can ascertain the true value of $\Delta_{i,j}$ using other predictions. Our error correction technique is inspired by the Low-Density Parity Check (LDPC) decoding method [GS12].

Formally, let the outcome of Eq. (14) denote the probability of each $\Delta \in \mathcal{K}$ being chosen as the correct $\Delta_{i,j}$, symbolized as $\mathbf{p}_{i,j}(\Delta)$. We can then correct $\mathbf{p}_{i,j}(\Delta)$ using the following expression:

$$\mathbf{p}_{i,j}(\Delta) = \mathbf{p}_{i,j}(\Delta) \cdot \prod_{a \notin \{i,j\}} \max_{\beta}(\mathbf{p}_{i,a}(\beta) \cdot \mathbf{p}_{j,a}(\beta \oplus \Delta)),\ \Delta \in \mathcal{K},\ \beta \in [0, 255]. \tag{16}$$

The right-hand side of the equation comprises two components: the original probability $\mathbf{p}_{i,j}(\Delta)$, and the composite probability of two Δ values that include a third key byte. We choose the maximum combined probability for all possible β, diverging from the original formulation, which proposed summation [GS12]. Ideally, when $\beta = \Delta_{i,a}$ and $\beta \oplus \Delta = \Delta_{j,a}$, $\mathbf{p}_{i,a}(\Delta_{i,a}) \cdot \mathbf{p}_{j,a}(\Delta_{j,a})$ will yield the maximum value. The maximum combined probability more accurately reflects the likelihood of each Δ guess, thereby enhancing the effectiveness of the attack, as demonstrated in the following sections.

5 Attack Scheme

Following Section 4, this section describes our attack scheme, which consists of three steps, shown in Figure 1. Note that an adversary will always execute the first two steps. Error correction is only performed when incorrect predictions are detected, detailed in the later paragraphs.

[2] The Spearman correlation offers more numerical stability than the Pearson correlation, as it has high tolerance when the labels and leakage features are not linearly correlated.

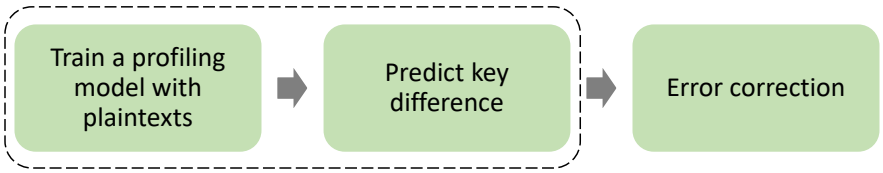


Figure 1: Plaintext-based SCCA.

In the first step, instead of profiling each plaintext byte separately, we employ multi-task learning (MTL) to profile all plaintexts simultaneously. MTL is a well-studied machine learning technique that trains multiple learning tasks in parallel [Car97, Rud17]. The main idea of MTL is to learn multiple tasks together to improve the learning of each task by leveraging information shared among related tasks [Car97, ZY21]. For its side-channel applications, previous works from Maghrebi and Masure et al. use MTL to attack multiple secret shares, leading to an enhanced attack performance compared with the single task learning [Mag20, MS21].

Figure 2 illustrates this paper's deep learning network architecture. The shared layers are responsible for processing the leakage and extracting general features. Then, several task-specific layers are constructed, forming sub-branches for each plaintext byte. First, the shared structure encourages the profiling model to learn the general features of different tasks, potentially leading to improved performance and resilience to overfitting. Second, MTL is computationally efficient as it only requires training a single model.

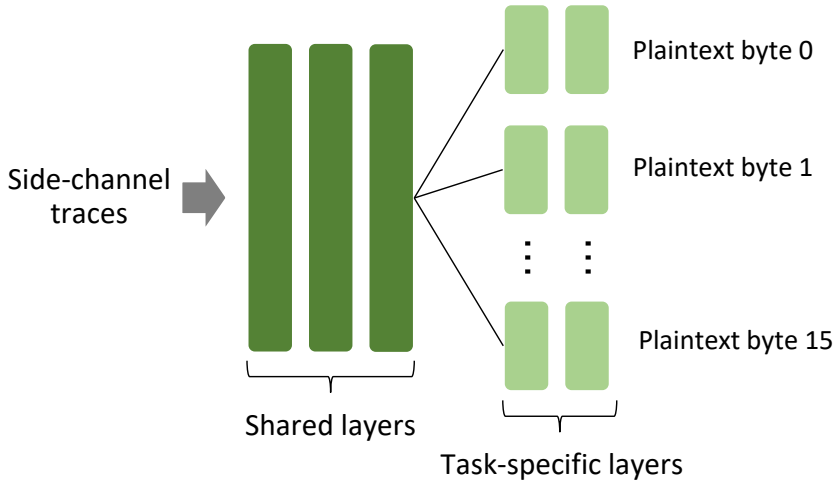


Figure 2: Deep learning architecture of plaintext-based SCCA.

It is worth noting that the proposed plaintext-based side-channel collision attack (PSCCA) is not limited to deep learning and can be applied to other profiling methods, such as Gaussian templates. In such cases, each plaintext would need to be profiled separately. We demonstrate its usage in Section 5.1.

Once the profiling step is completed, we calculate $\mathbf{p}_{i,j}(\Delta)$ for each pair of key bytes k_i and k_j using Eq. (14). Finally, error correction is applied if any of the predicted $\mathbf{p}_{i,j}(\Delta)$ values are incorrect. In the case of SCCA, it is straightforward to detect the presence of incorrect predictions. Assuming an SCCA targeting 16 bytes k_0, k_2, $\cdots$, k_{15}, a successful secret recovery would satisfy Eq. (17). If the equation does not hold, the error correction

method presented in Section 4.5 is applied.

$$\arg\max_{\Delta} \ \mathbf{p}_{0,15}(\Delta) = \arg\max_{\Delta} \ \mathbf{p}_{0,1}(\Delta) \oplus \arg\max_{\Delta} \ \mathbf{p}_{1,2}(\Delta) \oplus \cdots \oplus \arg\max_{\Delta} \ \mathbf{p}_{14,15}(\Delta) \quad (17)$$

To provide a clear formulation of PSCCA, the pseudocode of the attack is shown in Algorithm 1. Line 1 represents the building of the profiling model using plaintexts. The Δ guess is obtained from Line 2 to Line 6. Lines 7 and 8 are dedicated to the error correction step. As mentioned in Section 4.4, the plaintext-based collision requires $d_i = d_j$. Therefore, as shown in Line 4, it is important to reorder the traces to ensure this condition is satisfied for each Δ guess.

Algorithm 1: Plaintext-based DL Collision Attack

Input : Traces $\mathbf{T}$, Plaintext d, byte index i, byte length L, Profiling model PM
Output : Key difference $\Delta_{i,j},\ i,j \in L$

1 Train model $\mathsf{PM} \leftarrow \mathsf{Train}(\mathbf{T}, d)$;
2 **while** $i < L - 1$ **do**
3 **while** $i < j$ and $j < L$ **do**
4 Reorder $\mathbf{T}$ to make $d_i = d_j$;
5 $\mathbf{p}(d|d_i), \mathbf{p}(d|d_j) \leftarrow \mathsf{PM}(\mathbf{T}),\ d \in \mathcal{D}$;
6 $\mathbf{p}_{i,j}(\Delta) \leftarrow \mathsf{corr}(\mathbf{p}(d|d_j),\ \mathbf{p}(d \oplus \Delta | d_i \oplus \Delta)),\ \Delta \in \mathcal{K}$;
7 **if** $\arg\max_{\Delta} \mathbf{p}_{i,j}(\Delta)$ *is incorrect* **then**
8 $\Delta_{i,j} \leftarrow \mathsf{Error\_Correction}(\mathbf{p}_{i,j}(\Delta))$

5.1 Case Study with Simulated Datasets

To demonstrate the effectiveness of PSCCA, we present attack results using simulated datasets as described by Eq. (18).

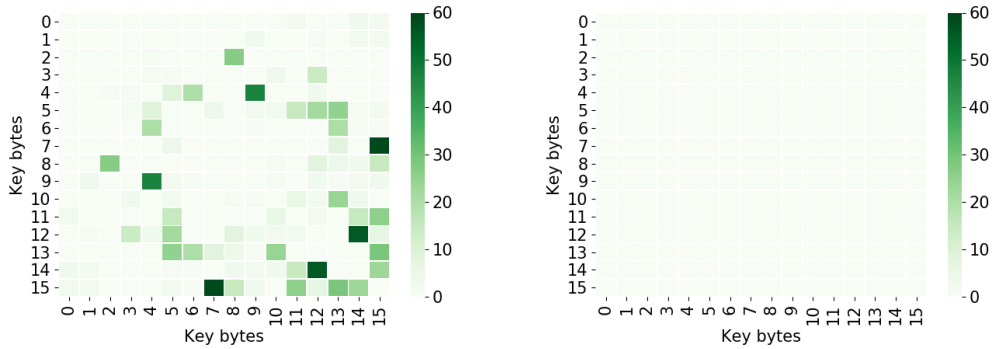
$$leakage = \mathsf{Sbox}(d_i \oplus k_i) + Z, \quad i \in [0, 15], \quad (18)$$

where d_i and k_i denote random plaintexts and a fixed key, respectively. All features are manipuated with noise $Z \sim \mathcal{N}(0, \sigma^2)$. To ensure the noise has the same effect on each test case, the leakage is normalized between 0 and 1. A total of 5 000 traces were simulated. As mentioned in Section 5, PSCCA can be applied to various profiling methods. In this case study, we utilize the Gaussian template as the benchmark due to its non-parametric nature. The profiling of each plaintext byte is performed separately.

Two test cases are considered, with different levels of noise represented by two σ values: 0.05 and 0.3. In the low-noise test case, the key difference between all key bytes is successfully recovered, highlighting the effectiveness of selective recovery with plaintext learning and plaintext-based collision. When the σ value is increased to 0.3, as shown in Figure 3a, we observe some unsuccessful Δ recoveries for certain key bytes using the plaintext-based collision method. Fortunately, by leveraging the Δ dependencies between different key bytes, as illustrated in Figure 3b, all key differences are successfully recovered after applying the error correction. This demonstrates the efficiency of the proposed error correction method in challenging scenarios with higher levels of noise.

6 Experimental Results

In this section, we evaluate the effectiveness of our proposed attack strategy using three publicly available side-channel datasets. We aim to perform a plaintext-based side-channel



(a) Before error correction. **(b)** After error correction.

Figure 3: Rank table of Δ before and after the error correction with the simulated dataset.

collision attack and recover all 16 subkeys. To achieve this, we employ a deep learning model shown in Section 5 that offers the advantage of attacking all sub bytes simultaneously.

The overall architecture of the model is depicted in Figure 2. Specifically, we adapt a Convolutional Neural Network (CNN) as detailed in [PWP22].[3] The network architecture is shown in Table 1. Note that a convolution block consists of a convolution layer, a pooling layer, and a batch normalization layer. The convolution block is used as shared layers; the remaining dense layers are assigned to each sub-branches.

Table 1: Deep learning architectures used in the experiments.

Layer	**Kernel number/size**	**Pooling stride/size**	**Neurons**
Convolution block	40/20	2/2	-
Dense	-	-	200
Dense	-	-	200

Regarding other hyperparameters, the activation function used for each layer is $Selu$, except for the final layer, which employs $Softmax$. The DL model is trained for 100 epochs. Based on our preliminary analysis, varying the training epochs does not significantly influence the attack performance. The batch size is set to 768, and the implications of this hyperparameter choice are discussed in Section 7.2. Data augmentation is implemented using a random translation layer following the input layer, randomly shifting the leakage measurements within a predefined augmentation level of 5. A comprehensive analysis of data augmentation is provided in Section 7.1.

It is important to note that the deep learning model and its hyperparameters remain constant across all attack methods. Although customizing the model and tuning hyperparameters for each dataset and method may enhance attack performance, such variables can significantly complicate our benchmarking conclusions by introducing model complexity and increasing training effort. Moreover, even after customizing the model and conducting hyperparameter tuning, it is impossible to guarantee the model's generality for a specific scenario. Since the main contribution of this paper lies in introducing a new attack method, the impact of selecting optimal solutions can be disregarded by opting for a model with acceptable performance.

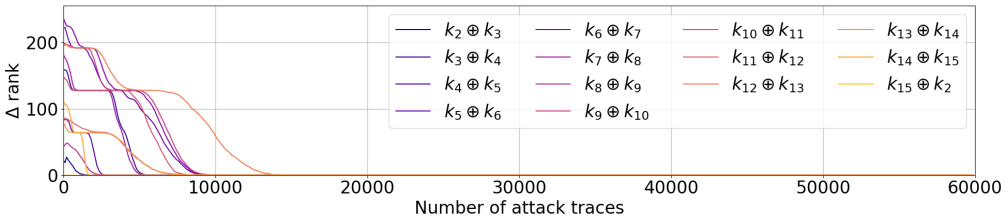
[3]The deep learning models were implemented using Python 3.6 and TensorFlow library version 2.6.0. Training algorithms were executed on an Nvidia GTX 1080TI GPU, managed by Slurm workload manager version 19.05.4.

To ensure reliable and robust evaluation, each attack scenario considered in this section is independently tested ten times to reduce the influence of random factors (e.g., random weight initialization) on attack performance [WPP22c]. The results are then averaged to represent the overall performance of each attack method.

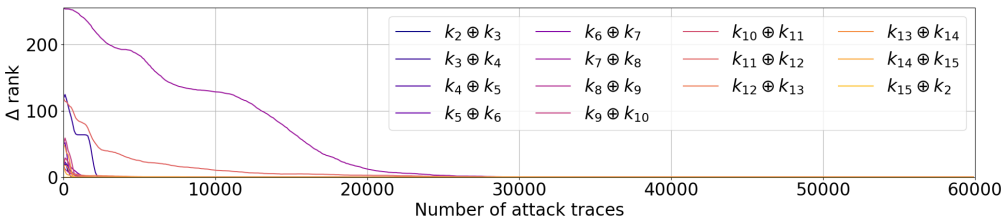
6.1 Attack Performance Analysis

This section provides a detailed analysis of the attack performance on the datasets under consideration. Aligned with Section 5.1, we present the results separately with and without the error correction.

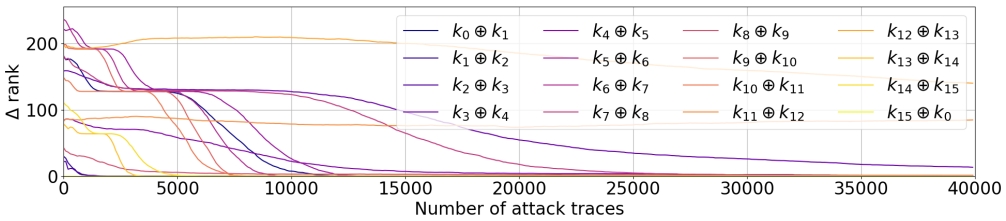
Figure 4 illustrates the Δ ranks for different combinations of key bytes. It is important to note that the ASCAD_F and ASCAD_R datasets' first two bytes are unmasked and therefore excluded from the attack due to their relative simplicity. As a result, all key bytes depicted in Figure 4 are mask-protected. The results show that the proposed plaintext-based collision method successfully recovers all key differences for ASCAD_F and ASCAD_R, thereby reducing the remaining key space to 2^8. For the CHES_CTF dataset, 13 of the 16 tested Δ values are successfully recovered.



(a) ASCAD_F.



(b) ASCAD_R.

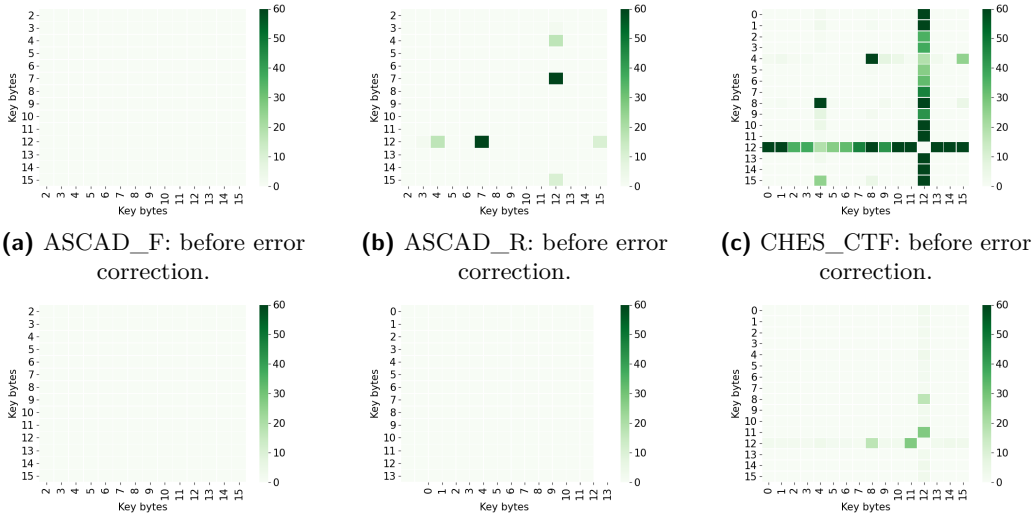


(c) CHES_CTF.

Figure 4: Δ rank for each dataset (no error correction).

A comprehensive overview of the attack performance for each dataset is depicted in Figure 5, which presents the Δ rank for all possible subkey combinations. Our method successfully breaks ASCAD_F with all possible subkey combinations, even without error correction. For ASCAD_R and CHES_CTF, the proposed method recovers a significant portion of Δ. As expected, the attack results improved further after applying the error correction method. It is worth mentioning that the 12th subkey of CHES_CTF consistently

exhibits mediocre performance in the Δ calculation. Fortunately, when error correction is applied, the Δ rank experiences a significant improvement. An overview of the overall success rate, representing the percentage of successfully recovered $\Delta_{i,j}$, is provided in Table 2. The error correction method contributes to a higher success rate for the ASCAD_R and CHES_CTF datasets.



(a) ASCAD_F: before error correction.

(b) ASCAD_R: before error correction.

(c) CHES_CTF: before error correction.

(d) ASCAD_F: after error correction.

(e) ASCAD_R: after error correction.

(f) CHES_CTF: after error correction.

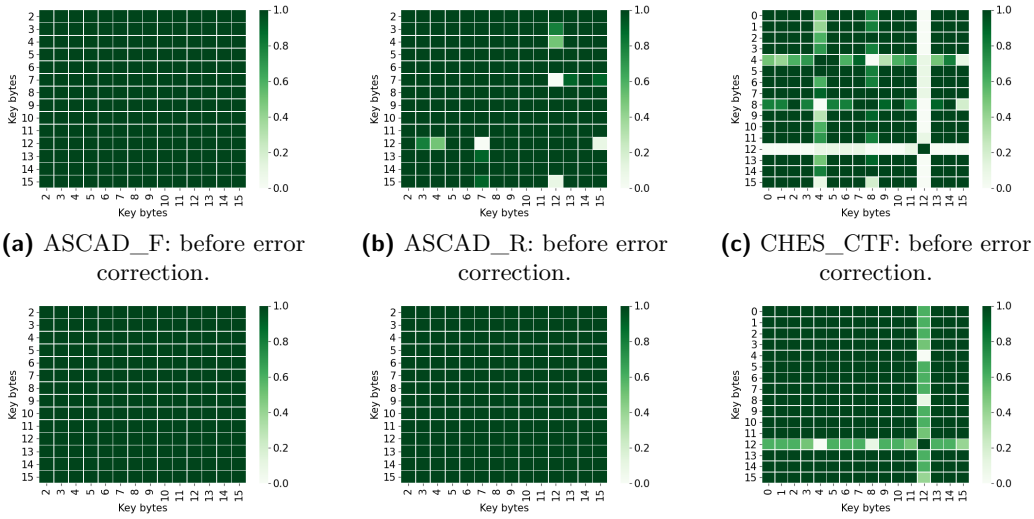
Figure 5: Rank table of Δ before and after error correction.

Table 2: Overall success rate of Δ recovery.

	ASCAD_F	ASCAD_R	CHES_CTF
Before error correction	100%	96%	78%
After error correction	100%	100%	88%

Our results outperform DL-SCCA [SM23] even without employing the error correction method. To demonstrate this, the success rate for all possible subkey combinations is presented in Figure 6. For the ASCAD_F and ASCAD_R datasets, DL-SCCA achieves an average success rate (across all guessed Δ) of 53% and 85%, respectively. In contrast, our method increases these rates to 100% and 97% before error correction and 100% and 100% after error correction. In the case of CHES_CTF, which is considered more challenging than ASCAD_F and ASCAD_R, with a higher number of attack traces required for key recovery in a profiling setting [PWP22], the average success rate is 82%, which improves to 94% after error correction. More importantly, our method is designed to work directly on the raw traces. Since the trace segmentation step is excluded from the attack, our method is more practical and could potentially handle complicated attack scenarios (i.e., hardware crypto implementation) where precise trace segmentation is impossible to be retrieved.

Our method is also more computational efficiency than DL-SCCA. With 100 training epochs, the average training time per dataset is approximately ten minutes. Knowing that a black-box attack with DL-SCCA requires around two hours [SM23], our method is significantly more efficient. Furthermore, attacking both sets of 16 bytes simultaneously allows each byte's attack to be completed in less than a minute, which is comparable with the state-of-the-art method with careful hyperparameter tuning to reduce its size [ZBHV19].



(a) ASCAD_F: before error correction.

(b) ASCAD_R: before error correction.

(c) CHES_CTF: before error correction.

(d) ASCAD_F: after error correction.

(e) ASCAD_R: after error correction.

(f) CHES_CTF: after error correction.

Figure 6: Success rate table of Δ before and after error correction.

It is worth noting that achieving a success rate of 100% is not always necessary for recovering Δ. One could count the occurrences of each Δ candidate being the most likely Δ over multiple attacks (we conducted ten attacks in this paper), and the one with the highest count number can be considered the correct key. Accordingly, the overall success rate for CHES_CTF can reach 100% as well. For instance, as shown in Figure 6f, the success rate of $\Delta_{7,12}$ is 60%, meaning that the correct candidate has been selected six times over ten attacks. In this case, an adversary is confident that the correct $\Delta_{7,12}$ is retrieved by choosing the Δ candidate that happens the most frequently.

7 Hyperparameter Evaluation

This section investigates various hyperparameters on the proposed plaintext-based side-channel collision attack. The overall success rate of all possible Δ is used to understand the influence of hyperparameter changes better. In the meantime, the influence of the error correction method is also analyzed in this section.

7.1 Data Augmentation

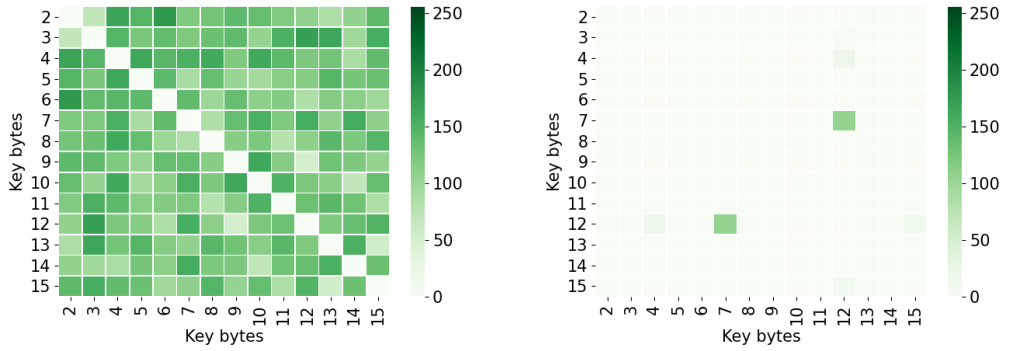
Preprocessing of leakage measurements is essential for an efficient attack. For this work, in addition to normalizing the data, data augmentation plays a pivotal role in the success of the proposed attack. To demonstrate this, we study the impact of different data augmentation levels on attack performance, and the comprehensive findings are presented in Table 3.

When the data augmentation level is set to zero, PSCCA achieves a 100% success rate only with the ASCAD_F dataset. However, a significant performance improvement is observed when introducing random shifts in the datasets. For instance, as shown in Figure 7, PSCCA has a mediocre performance when the data augmentation level is zero; when increasing to 5, most of the key differences can be recovered. Indeed, optimal results consistently emerge within the data augmentation range of 5 across all test scenarios. These observations underscore the critical role of data augmentation in enhancing the

Table 3: Hyperparameter evaluation on data augmentation.

	0	5	10	20
ASCAD_F	100%	100%	100%	28%
ASCAD_R	7%	96%	99%	94%
CHES_CTF	16%	78%	48%	15%

effectiveness of the proposed method. As a regularization technique, data augmentation helps prevent the profiling model from fixating on specific features, enabling it to focus on global features instead. In the context of side-channel attacks, where data leakages exist in only a few features, such techniques prevent the model from overfitting irrelevant features. Note that various strategies can be employed to mitigate overfitting. For instance, early stopping techniques can be applied, halting model training if a monitored metric fails to improve after a certain number of epochs.



(a) Data augmentation: 0. **(b)** Data augmentation: 5.

Figure 7: Rank table of Δ for ASCAD_R dataset with different data augmentation levels.

Nevertheless, it is important to note that employing an excessively high level of data augmentation can have adverse effects, leading to a decline in attack performance. As the data augmentation level reaches 20, there is a noticeable deterioration in attack performance across various configurations. This high augmentation level increases the complexity of fitting the model to the leakage as the timing of the leakages becomes more random. Consequently, longer training periods or larger models may be required, increasing computational effort.

Considering our objective of providing an end-to-end solution for the side-channel collision attack, we aim to reduce the hyperparameters' tuning effort while achieving commendable attack performance, and error correction would be a suitable solution. Table 4 presents the effect of data augmentation variation with error correction enabled. A noticeable increase in the success rate is observed across all test settings, except when the guessing entropy is zero. This limitation arises because the error correction method relies on key differences involving a third byte. The error correction method becomes non-functional if these key differences are also incorrect. In such scenarios, as outlined at the end of Section 6.1, a potential workaround could be applying an ensemble method. This approach would involve repeating the attack multiple times and choosing the most frequently occurring Δ candidate.

Table 4: The influence of error correction on the data augmentation variation.

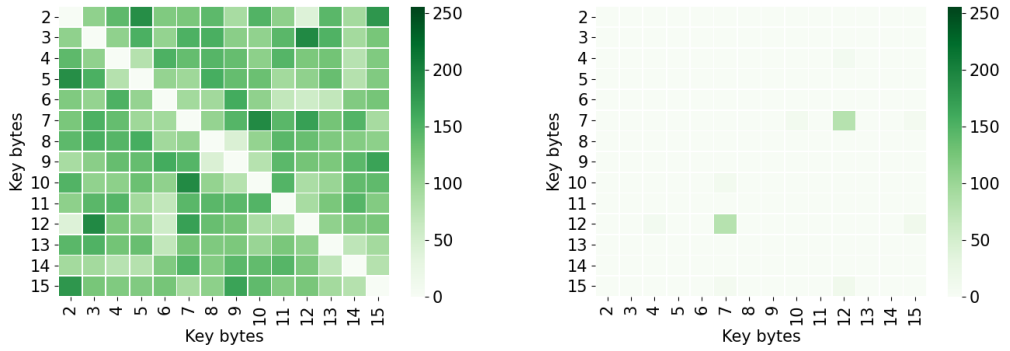
	0	5	10	20
ASCAD_F	100%	100%	100%	49%
ASCAD_R	7%	100%	100%	100%
CHES_CTF	16%	88%	67%	31%

7.2 Batch Size

Batch size in a deep learning model refers to the number of examples (input-output pairs) used in a single training iteration. This parameter plays a crucial role in shaping the behavior and overall performance of the deep learning model. Table 5 illustrates that larger batch sizes generally improve attack performance in the proposed method. This can be attributed to larger batch sizes providing a more accurate gradient estimation. As a concert example, Figure 8 shows the influence of the batch size variation with the ASCAD_R dataset. One could observe a significant performance leap with larger batch size. Indeed, in deep learning, the gradients guide the model's learning process by indicating the direction in which the model parameters should be updated. With a larger batch size, the gradient estimation becomes more reliable as it incorporates information from more data points, potentially leading to better learning and performance.

Table 5: Hyperparameter evaluation on batch size.

	128	256	512	768	1 024
ASCAD_F	31%	7%	100%	100%	100%
ASCAD_R	7%	95%	97%	96%	97%
CHES_CTF	6%	72%	73%	78%	81%



(a) Batch size: 128. **(b)** Batch size: 256.

Figure 8: Rank table of Δ for ASCAD_R dataset with different batch sizes.

Moreover, larger batch sizes enable more efficient utilization of computational resources, particularly GPUs, which tend to exhibit optimal performance when processing computations in larger blocks. In our experiments, for example, a batch size of 1 024 completed tests on all datasets four times faster than a batch size of 128, highlighting the advantage of utilizing larger batch sizes in terms of computational efficiency.

Consistent with the previous section, we also investigated the impact of batch size on attack performance when error correction is enabled. Similarly, error correction contributes

to a success rate increase in most test cases except in test cases where the success rate is already low. Notably, the CHES_CTF dataset consistently achieves an 88% success rate when the batch size exceeds 256. As detailed in Section 6.1, the performance is mediocre when the key difference calculation involves the byte 12 of the CHES_CTF key, compared to other scenarios. For error correction, the correction of, for example, $\Delta_{1,12}$, would involve $\Delta_{1,x}$ and $\Delta_{x,12}$, where x represents a random byte that does not equal 1 or 12. Since $\Delta_{x,12}$ is also incorrect, the resultant corrections would likewise be erroneous. In this case, the ensemble method proposed at the end of Section 6 would be a potential solution to recover the correct key.

Table 6: The influence of error correction on the batch size variation.

	128	256	512	768	1 024
ASCAD_F	42%	7%	100%	100%	100%
ASCAD_R	7%	100%	100%	100%	100%
CHES_CTF	6%	88%	88%	88%	88%

8 Conclusions and Future Work

This paper presents a novel end-to-end approach for side-channel collision attacks. We introduce a plaintext-based side-channel collision attack, eliminating the need for trace segmentation during an attack. Additionally, we propose an error correction scheme to enhance the accuracy of the correction key difference prediction, thereby improving the overall attack performance. Furthermore, we employ multi-task learning to attack all sub-key bytes simultaneously, resulting in efficient key difference recovery without targeting each byte individually. By applying our framework to three publicly available masked AES datasets, we achieve profiling attack results that significantly surpass the current state-of-the-art deep learning-based side-channel collision attack [SM23] regarding attack performance and computation effeciency. Importantly, our approach demonstrates its generality across various attack scenarios, as minimal effort is required for hyperparameter tuning.

Building upon this work, there are several promising directions for further investigation. Firstly, exploring improved methods for retrieving the correct key difference would be interesting, such as refining the Spearman correlation technique. This could lead to a significant reduction in the number of attack traces required. Secondly, we plan to continue enhancing the error correction method, potentially incorporating ensembles or other techniques. Developing a more robust deep learning model could also strengthen the relationship between the input and each task. For instance, direct connections between the input layer and the model's subbranch may reduce reliance on the main branch and improve feature extraction capabilities. Lastly, given that our proposed method eliminates the need for trace segmentation, it would be worthwhile to investigate its effectiveness in hardware crypto implementations. Such an exploration would contribute to a deeper understanding of the method's effectiveness in challenging cryptographic scenarios.

References

[BCH+20] Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan Shrivastwa. Mind the portability: A warriors guide through realistic profiled side-channel analysis. In *27th Annual*

Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020. The Internet Society, 2020.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–29. Springer, 2004.

[Bog07] Andrey Bogdanov. Improved side-channel collision attacks on aes. In *Selected Areas in Cryptography: 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers 14*, pages 84–95. Springer, 2007.

[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.

[CRR02] 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.

[DLH+22] Ngoc-Tuan Do, Phu-Cuong Le, Van-Phuc Hoang, Van-Sang Doan, Hoai Giang Nguyen, and Cong-Kha Pham. Mo-dlsca: Deep learning based non-profiled side channel analysis using multi-output neural networks. In *2022 International Conference on Advanced Technologies for Communications (ATC)*, pages 245–250, 2022.

[GS12] Benoît Gérard and François-Xavier Standaert. Unified and optimized linear collision attacks and their application in a non-profiled setting. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 175–192. Springer, 2012.

[HDD22] Van-Phuc Hoang, Ngoc-Tuan Do, and Van Sang Doan. Efficient non-profiled side channel attack using multi-output classification neural network. *IEEE Embedded Systems Letters*, pages 1–1, 2022.

[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.

[HK11] Jan Hauke and Tomasz Kossowski. Comparison of values of pearson's and spearman's correlation coefficients on the same sets of data. *Quaestiones geographicae*, 30(2):87, 2011.

[KJJ99a] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, UK, 1999. Springer-Verlag.

[KJJ99b] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KWPP22] Maikel Kerkhof, Lichao Wu, Guilherme Perin, and Stjepan Picek. Focus is key to success: A focal loss function for deep learning-based side-channel analysis. In *Constructive Side-Channel Analysis and Secure Design*, pages 29–48. Springer International Publishing, 2022.

[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.

[MBPK22] Naila Mukhtar, Lejla Batina, Stjepan Picek, and Yinan Kong. Fake it till you make it: Data augmentation using generative adversarial networks for all the crypto you need on small devices. In Steven D. Galbraith, editor, *Topics in Cryptology – CT-RSA 2022*, pages 297–321, Cham, 2022. Springer International Publishing.

[MME10] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth. Correlation-enhanced power analysis collision attack. In *Cryptographic Hardware and Embedded Systems, CHES 2010: 12th International Workshop, Santa Barbara, USA, August 17-20, 2010. Proceedings 12*, pages 125–139. Springer, 2010.

[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.

[MS16] Amir Moradi and François-Xavier Standaert. Moments-correlating dpa. In *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, pages 5–15, 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.

[PHJ+17] Stjepan Picek, Annelie Heuser, Alan Jovic, Simone A. Ludwig, Sylvain Guilley, Domagoj Jakobovic, and Nele Mentens. Side-channel analysis and machine learning: A practical perspective. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 4095–4102, 2017.

[PPM+23] Stjepan Picek, Guilherme Perin, Luca Mariot, Lichao Wu, and Lejla Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Computing Surveys*, 55(11):1–35, 2023.

[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*, 2022(4):828–861, Aug. 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.

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

[SM23] Marvin Staib and Amir Moradi. Deep learning side-channel collision attack. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):422–444, Jun. 2023.

[SWP03] Kai Schramm, Thomas Wollinger, and Christof Paar. A new class of collision attacks and its application to des. In *Fast Software Encryption: 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003. Revised Papers 10*, pages 206–222. Springer, 2003.

[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.

[vWWB11] Jasper GJ van Woudenberg, Marc F Witteman, and Bram Bakker. Improving differential power analysis by elastic alignment. In *Topics in Cryptology–CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 104–119. Springer, 2011.

[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.

[WPP22a] Lichao Wu, Guilherme Perin, and Stjepan Picek. The best of two worlds: Deep learning-assisted template attack. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 413–437, 2022.

[WPP22b] 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*, pages 1–12, 2022.

[WPP22c] Lichao Wu, Guilherme Perin, and Stjepan Picek. On the evaluation of deep learning-based side-channel analysis. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 49–71. Springer, 2022.

[WPP23a] Lichao Wu, Guilherme Perin, and Stjepan Picek. Hiding in plain sight: Non-profiling deep learning-based side-channel analysis with plaintext/ciphertext. *Cryptology ePrint Archive*, 2023.

[WPP23b] Lichao Wu, Guilherme Perin, and Stjepan Picek. Not so difficult in the end: Breaking the ascadv2 dataset. *Cryptology ePrint Archive*, 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.

[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*, 2020(3):73–96, Jun. 2020.