# Not so Difficult in the End: Breaking the Lookup Table-based Affine Masking Scheme

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

[1] Radboud University, The Netherlands
[2] Leiden University, The Netherlands

**Abstract.** The lookup table-based masking countermeasure is prevalent in real-world applications due to its potent resistance against side-channel attacks and low computational cost. The ASCADv2 dataset, for instance, ranks among the most secure publicly available datasets today due to two layers of countermeasures: lookup table-based affine masking and shuffling. Current attack approaches rely on strong assumptions. In addition to requiring access to the source code, an adversary would also need prior knowledge of random shares.
This paper forgoes reliance on such knowledge and proposes two attack approaches based on the vulnerabilities of the lookup table-based affine masking implementation. As a result, the first attack can retrieve all secret keys' reliance in less than a minute without knowing mask shares. Although the second attack is not entirely successful in recovering all keys, we believe more traces would help make such an attack fully functional.

**Keywords:** Side-channel analysis · Side-channel collision attack · Correlation.

## 1 Introduction

Side-channel analysis (SCA) on symmetric-key cryptography implementations is typically divided into non-profiling [20,31] and profiling attacks [10,37], depending on the availability of a replica of the device under attack (profiling device). Non-profiling attacks operate without this assumption, and an adversary collects measurements that encode secret information and subsequently perform a statistical analysis to form a guess about the secrets. In contrast, profiling attacks assume that the adversary has unrestricted control over a duplicate of the targeted device. Using this duplicate, the adversary identifies and understands the device's side-channel behavior, subsequently leveraging this knowledge to extract secret information from the device under attack. Recent advancements, especially deep learning-based side-channel analysis, have significantly improved profiling attacks. Today, researchers can compromise various protected targets using a single measurement, underscoring the impressive progress in this field [24]. Such results were achieved on datasets only a few years ago considered difficult to break: ASCAD with fixed key and ASCAD with random keys, both protected with the first-order Boolean masking.

However, one dataset is yet to be broken without prior knowledge about the random shares: ASCADv2 [2]. Indeed, this secure AES-128 implementation published by the Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI) has been protected with multiple layers of hiding (shuffling) and masking countermeasures. Specifically, a lookup table-based masking scheme [9,28,13] is adopted, wherein a masked $\mathsf{Sbox}$ is pre-computed with random shares before the cryptographic operation. Consequently, any intermediate data leakages relating to the non-linear operation of the cipher are effectively eliminated. Although the weaknesses of lookup table-based masking schemes have been discussed in [32], to our knowledge, a direct key-recovery attack without any prior knowledge of mask shares never succeeded. One of the most significant challenges in overcoming this implementation is the masking schemes that incorporate both multiplicative mask computation with finite field multiplication over $\mathsf{GF}(2^8)$ and additive (Boolean) masks. Traditional attacks on the Boolean masked implementation depend on the ability of the profiling model to combine mask shares and masked data to retrieve the sensitive data. For ASCADv2, even though the multiplicative masks significantly leak [23], the additive masked Galois field multiplication is complex for a profiling model to comprehend, even when leveraging deep learning [8]. Therefore, all existing attacks on this dataset are performed in a white-box setting with prior knowledge of the random shares, at least in the profiling phase or in both profiling and attack phases [23,22]. We argue that such assumptions could not be practical even in secure evaluation labs that perform white-box evaluations. Although cryptographic algorithms are evaluated with all implementation details (e.g., source code of the cryptographic library and hardware design details), the random shares would rarely be accessible to an evaluator. The reason is straightforward: the registers that store the random values for the system protection would never be accessible from the outside world unless severe implementation flaws exist. Although it is possible to predict the output of some weak pseudo-random number generators (PRNG) with modeling techniques [1], we consider it difficult considering the unknown random seeds and complexity (e.g., high-order polynomials) of PRNG. Other ways of bypassing such protections are monitoring the random number leakages on data bus with probes or forcing the PRNG stuck at some fixed value with fault injection. However, it highly depends on the implementation, and it is out of the scope of this paper that focus solely on SCA.

This paper presents two vulnerabilities in ASCADv2's affine masking implementation that could lead to successful key recovery without knowledge of the mask shares. With the knowledge of plaintexts, the implementation could be broken down in less than a minute with a CPU only. Note that we disable the shuffling countermeasure and concentrate solely on masking schemes. Although turning on this countermeasure would cause the proposed attack to fail with the number of traces we have, we expect it could be circumvented with, for instance, more leakage measurements [39]. For instance, without further means to overcome the shuffling, one would need approximately 120 times more traces to overcome a true random permutation. Our main contributions are:

1. We conduct an in-depth investigation into two vulnerabilities inherent in implementing the lookup table-based affine masking scheme, substantiating our findings with theoretical analysis.
2. We propose several strategies to execute second-order attacks that leverage the identified vulnerabilities.
3. We demonstrate two attack methodologies that lead to efficient key recovery without the knowledge of the mask shares.
4. We discuss several protection methods that would be resilient to our attack.

The rest of this paper is organized as follows. In Section 2, we provide the necessary background information. Section 3 discusses related works. Section 4 details the identified vulnerabilities. In Section 5, we provide experimental results. Section 6 discusses the identified vulnerability from a higher level, then offer possible protection methods to defend against proposed attacks. Finally, in Section 7, we conclude the paper and discuss potential future research directions.

## 2 Preliminaries

This section introduces the notation we follow. Afterward, the relevant information about the side-channel analysis, collision attack, and the targeted ASCADv2 dataset is discussed.

### 2.1 Notation

We utilize calligraphic letters such as $\mathcal{X}$ to represent sets, while the corresponding uppercase letters $X$ denote random variables and random vectors $\mathbf{X}$ defined over $\mathcal{X}$. Realizations of $X$ and $\mathbf{X}$ are denoted by lowercase letters $x$ and $\mathbf{x}$, respectively. Functions, such as $\mathsf{f}$, are presented in a sans-serif font.

The symbol $k$ represents a candidate key byte in a key space $\mathcal{K}$. The notation $k^*$ refers to the correct key byte or the key byte assumed to be correct by the adversary.[3]

A dataset $\mathbf{T}$ comprises traces $\mathbf{t}_i$, which are associated with plaintext/ciphertext pairs $\mathbf{d}_i$ in plaintext/ciphertext space $\mathcal{D}$ and keys $\mathbf{k}_i$, or $k_{i,j}$ and $d_{i,j}$ when considering partial key recovery on byte $j$. Throughout this work, we focus on a fixed key scenario where $\mathbf{k}_i$ remains constant for each trace $\mathbf{t}_i$, resulting in the utilization of 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 types, profiling SCA and non-profiling SCA, based on the availability of a fully-controlled cloned device. Non-profiling side-channel analysis exploits the correlation between key-related intermediate values

[3] It is important to note that subkey candidates can involve guessing any number of bits. Although we assume the AES cipher here, the concept remains algorithm-independent.

and leakage measurements. An adversary collects a series of traces generated during the encryption of different plaintexts. The adversary can guess a key portion by examining the correlation between the key-related intermediate values and the leakage measurements. The attack strategy typically involves a "divide-and-conquer" approach. First, an adversary divides the traces into groups based on the predicted intermediate value corresponding to the current key guess. If the groups exhibit noticeable differences (the definition of "difference" depends on the attack method), it suggests that the current key guess is likely correct. The non-profiling attacks assume relatively weaker adversaries who do not have access to a cloned device. Consequently, these attacks may require many measurements (potentially millions) to extract confidential information. Examples of non-profiling attacks include simple power analysis (SPA), differential power analysis (DPA) [20]/correlation power analysis (CPA) [7], and some machine learning-based attacks [31,16,38]. Note that side-channel collision attack [29,5] and its deep learning version [30] are also considered a non-profiling SCA but follows a slightly different strategy, discussed in the next section.

Profiling side-channel attacks aim to map a set of inputs (e.g., side-channel traces) to outputs (e.g., a probability vector of key hypotheses). Profiling attacks involve two phases. In the profiling phase, the adversary constructs a profiling model $\mathsf{f}_{\boldsymbol{\theta}}^{\mathsf{M}}$, parameterized by a leakage model $\mathsf{M}$ and a set of learning parameters $\boldsymbol{\theta}$. This model maps the inputs (side-channel measurements) to the outputs (classes obtained by evaluating the leakage model during a sensitive operation) using a set of $N$ profiling traces. The notations $\mathsf{f}_{\boldsymbol{\theta}}^{\mathsf{M}}$ and $\mathsf{f}_{\boldsymbol{\theta}}$ are used interchangeably. Then, in the attack phase, the trained model processes each attack trace $\mathbf{t}_i$ and produces a vector of probabilities $\mathbf{p}_j$, representing the likelihood of the associated leakage value or label $j$. The adversary determines the best key candidate based on this vector of probabilities. If the adversary constructs an effective profiling model, only a few measurements from the target device may be sufficient to break its security. Examples of profiling attacks include the template attack [10], stochastic models [27], and supervised machine learning-based attacks [19,21,25].

### 2.3 Side-channel Collision Attack

Side-channel Collision Attack (SCCA) represents a class of non-profiling attacks that leverage information dependence leaked during cryptographic processes. Traditional collision attacks capitalize on situations where two distinct inputs into a cryptographic algorithm yield an identical output, a circumstance known as a "collision". Since collisions are generally infrequent in robustly designed cryptographic systems, SCCA explicitly targets the internal state, which is more likely to be identical.

In SCCA, an adversary observes the side-channel information as the system processes different inputs. The adversary then scans for patterns or similarities in the side-channel data that indicate a collision has occurred. Upon identifying a collision, the adversary can utilize this knowledge to deduce information about the inter-dependencies of different key portions or the algorithm's internal state, thereby significantly reducing the remaining key space.

As an illustration, let us consider the SubBytes operation of the Advanced Encryption Standard (AES) [15]. The same intermediate data (Sbox input or output) could be processed if two different Sbox operations result in an identical side-channel pattern. Since the Sbox operation is bijective (i.e., a one-to-one correspondence between two sets), we have the following equations:

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

Note that a collision of Sbox input would also satisfy $k_i \oplus k_j = p_i \oplus p_j$. Indeed, in contrast to other SCA methods concentrating on key recovery, SCCA aims to uncover the linear difference between various keys. By making guesses on a single subkey, an adversary can leverage this linear difference to compute the remainder of the key. This essentially reduces the remaining key space to the equivalent of a single byte, $2^8$.

### 2.4 The ANSSI's AES Implementation: ASCADv2

ANSSI has published a library implementing a secure AES-128 on an ARM Cortex-M4 architecture [2] together with 800 000 power measurements focusing on the full AES encryption. This implementation is equipped with several layers of countermeasures, such as affine secret-sharing [17] and random shuffling of independent operations [36]. We briefly discuss their implementations in this section. More implementation details can be found on the corresponding GitHub page [2] and paper [8,23].

An overview of generating a mask state $C_i$ with an AES state $X_i$ is shown in Equation 2.

$$C_i = (X_i \otimes \alpha) \oplus \beta, \tag{2}$$

where $i$ stands for byte indices. Two random shares realize the affine masking scheme: the multiplicative share $\alpha$ and additive share $\beta$. Finite field multiplication over $\mathsf{GF}(2^8)$ and xor are denoted by $\otimes$ and $\oplus$, respectively. Note that $\beta$ may denote Sbox's input mask $r_{in}$, Sbox's output mask $r_{out}$, or $r_l$, the mask used in the linear operation of AES [4]. To ensure there is no direct leakage on the AES state, a masked Sbox, denoted as $\mathsf{Sbox}_m$, is pre-computed for all bytes based on $r_{in}$, $r_{out}$ and $\alpha$, enabling the processing of the masked data in the non-linear part of AES, illustrated in Equation 3. Note that $r_l$ is removed after $r_{in}$ is applied and added before $r_{out}$ is canceled. Therefore, sensitive states are masked during the entire AES process.

$$(X_i \otimes \alpha) \oplus r_{in} \xrightarrow{\mathsf{Sbox}_m} (\mathsf{Sbox}(X_i) \otimes \alpha) \oplus r_{out}. \tag{3}$$

The random shares $\alpha$, $r_{in}$, and $r_{out}$ remain the same during the computations of each byte and are refreshed in the next AES operation.

[4] The proposed attack target the intermediate data when $\beta = r_{in}$ and $\beta = r_{out}$.

Random permutations are applied to ShiftRows, MixColumns, and Sbox executions; the permutations indices for each byte are generated based on random seeds.

## 3 Related works

Side-channel analysis has been widely researched and applied to different cryptographic algorithms during past decades. Multiple attack methods have been developed, such as direct (non-profiled) attacks like Simple Power Analysis (SPA), Differential Power Analysis (DPA) [20], and two-stage (profiling) attacks like the template attack [10]. Machine learning-based attacks have been actively researched in recent years and could be used in both profiling [19,25,41,37,24] and non-profiling settings [31,16,38].

ASCAD [3], a first-order masked AES-128 implementation running on an 8-bit AVR microcontroller, is one of the most studied datasets by the side-channel community. While there are two versions of this ASCAD dataset (one with a fixed key and the other one with random keys), there is little difference in attacking those two datasets, see, e.g., [24], which is also discussed in more generic terms of portability difficulty in [4]. During only a few years of active research, the secret key of this dataset managed to be retrieved from around a thousand attack traces [3] to one trace [24]. For an overview of novel attack methodologies based on the publicly available implementations and the corresponding leakage measurements, as well as for the details on those datasets, we refer readers to [26]. Considering that almost all of the available datasets can be "easily" broken, there is a strong demand from the SCA community to have more robust open-source implementations and leakage measurements. Indeed, knowing the complexity of modern devices, we see a large disbalance between the realistic implementations and those studied in academia. The release of new cryptographic implementations implemented with different hardware, software, and protections fills the gap between academics and the real world.

Lookup table-based masking is a common countermeasure against SCA. This strategy, particularly when applied to mask the Sbox, stands out for its computational efficiency. The pre-computed Sbox notably reduces the computational load during operations. The initial provably secure first-order lookup table-based masking scheme was proposed by Chari et al. [9]. A randomized Sbox lookup table undergoes a shift and receives protection with an output mask. Following this seminal work, enhancements have been made in terms of enhancing its secure order [28,11,12] and decreasing its memory requirement [33,34]. In 2019, ANSSI publicly released a hardened AES-128 implementation. This secure variant employs a lookup table-based affine masking scheme in line with [17], incorporating both multiplicative and additive masking. This combination poses a significant challenge to the profiling attack on first-order leakages. Initial security analysis of this implementation was undertaken by Bronchain et al., who proposed several attack strategies given knowledge of the source code, the secret key, and the random shares processed during the profiling phase [8]. Following this, Cristiani

et al. [14] uses non-profiled SCA with Joint Moments Regression to break the ASCADv2 dataset with 100M traces. Masure et al. conducted partial attacks on various shares and permutation indices [23]. The knowledge they garnered from these attacks was subsequently used to orchestrate a global attack on the protected data. Marquet et al. further contributed to the field by highlighting the superiority of multi-task learning over single-task learning when the analysis is focused exclusively on secret data [22]. Recently, Vasselle et al. published an AES implementation that included both masking and artificially implemented shuffling as countermeasures [35]. They successfully breached the target using a spatial dependency analysis. Their research has helped to further our understanding of the strengths and weaknesses of these countermeasures and offers new avenues for exploration in securing AES implementations.

## 4 Vulnerability Analysis

This section first discusses the constant affine mask shares used in ASCADv2. Afterward, we discuss the zero input to the affine masking scheme.

### 4.1 Constant Affine Mask Shares for an Encryption

As mentioned, the ASCADv2 implementation is protected by an affine masking scheme consisting of independent additive and multiplicative mask shares (see Equation 2). This implementation increases the security level of the implementation [8,23]. Nonetheless, upon analyzing the code, we observe that the same pre-computation table is used for all state bytes, meaning that additive and multiplicative masks remain constant throughout a single AES encryption. Random values are pre-loaded into mask registers before encryption and are retrieved during mask calculations. Such an implementation presents the opportunity to bypass these masking schemes altogether. Formally, assuming $C_i = C_j$ during an AES processing, we have:

$$\begin{aligned} (X_i \otimes \alpha) \oplus \beta = (X_j \otimes \alpha) \oplus \beta \\ \implies \quad X_i = X_j,\ \alpha \neq 0. \end{aligned} \tag{4}$$

**Lemma 1.** *Given $X_i \otimes \alpha \oplus \beta = X_j \otimes \alpha \oplus \beta$, we xor both sides of the equation with $\beta$ to cancel it out*

$$X_i \otimes \alpha = X_j \otimes \alpha. \tag{5}$$

*Since we work with finite field multiplication (in $\mathsf{GF}(2^8)$), each element has a unique inverse (except the element 0). Since $\alpha$ is non-zero (otherwise, both sides of the original equation would equal $\beta$, which would not provide any information), we can multiply both sides of the equation by the multiplicative inverse of $\alpha$, denoted as $\alpha^{-1}$:*

$$X_i \otimes \alpha \otimes \alpha^{-1} = X_j \otimes \alpha \otimes \alpha^{-1},\ \alpha \neq 0. \tag{6}$$

*Applying the associative property of finite field multiplication over* $GF(2^8)$*, we have:*

$$\begin{aligned} X_i \otimes (\alpha \otimes \alpha^{-1}) &= X_j \otimes (\alpha \otimes \alpha^{-1}) \\ \implies X_i \otimes 1 &= X_j \otimes 1 \\ \implies X_i &= X_j. \end{aligned} \tag{7}$$

*Therefore, a collision between* $X_i$ *and* $X_j$ *is created without the knowledge of* $\alpha$ *and* $\beta$*.*

Equation 4 illustrates the vulnerability of this AES implementation. Indeed, a fixed mask can be easily canceled by comparing intermediate data protected by the same mask shares. Note that $X_i$ and $X_j$ could be key-related intermediate data, represented by $\mathsf{Sbox}(k_j \oplus p_j)$. In this case, Equation 1 is satisfied if $X_i$ equals $X_j$. Since the plaintext is known, we adopt a side-channel collision attack to retrieve $k_i \oplus k_j$ for all keys, detailed in Section 5.1.

### 4.2 Zero Input of Affine Masking Scheme

As discussed in Equation 7, the multiplicative mask $\alpha$ is non-zero, so each element has a unique inverse. However, it is also possible that $X_i$ is zero (e.g., $\mathsf{Sbox}(\cdot) = 0$). Formally speaking, when $X_i = 0$, Equation 2 can be rewritten as:

$$\begin{aligned} C_i &= (X_i \otimes \alpha) \oplus \beta \\ &= 0 \oplus \beta \\ &= \beta. \end{aligned} \tag{8}$$

The masked state $C_i$ only relies on $\beta$, and the multiplicative mask $\alpha$ is disabled in this scenario, links to the zero-value $2^{nd}$-order leakage mentioned in [17]. To exploit this attack path, an adversary would try all possible keys to calculate $X_i$ and select the traces that satisfy $X_i = 0$. Then, the chosen traces are correlated with $\beta$. The traces set with the highest correlation would indicate the correct key.

There are two ways to perform such an attack. The first attack path requires the knowledge of $\beta$, indicating that an adversary should, for instance, access the output of a PRNG that provides the mask value. In this case, one could conduct the attack in the profiling SCA setting similar to other researches [8,23,22], namely learning $\beta$ on the cloned and fully controlled device and predict $\beta$ on a victim device, finally performing correlation analysis using the predicted values and leakage measurements. Since this attack path relies on the knowledge of the mask shares, it is less interesting considering the scope of this paper that aims at breaking ASCADv2 with no assumption on prior knowledge about the mask shares.

The second attack path is similar to a side-channel collision attack in which an adversary compares two trace segments. Instead of correlating with the $\beta$ value, an adversary could correlate with the leakage segments that process $\beta$.

According to the source code, since $\beta$ is handled in plaintext (which makes sense as there is no need to protect a random value from side-channel leakages), we expect significant leakages of the $\beta$ processing. The relevant features would correlate well with the trace segments that process SubBytes with zero Sbox inputs. The attack results are shown in subsection 5.2.

## 5 Attack Results

This section provides experimental results, first the collision attack on canceling mask shares, followed by correlation attack on $\mathsf{GF}(0)$. Instead of regenerating leakage traces [8,23], the original traces provided by ANSSI are used for attacks [5].

### 5.1 Side-channel Collision Attack on Canceling Mask Shares

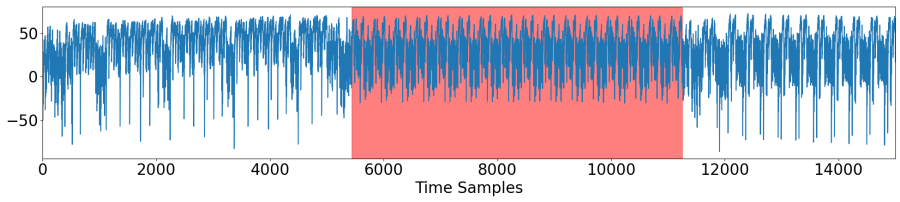
The collision attacks require the trace segments of each intermediate data processing. Therefore, the leakage analysis is crucial for the success of such an attack. Figure 1a presents an averaged trace representing the first round of the AES. Y-axis stands for the leakage amplitude. The sixteen SubBytes operations are highlighted in red. Repeated patterns can be observed when zooming in on each SubBytes operation, as shown in Figure 1b. The trace segments for each operation ($T^0$, $T^1$, $\cdots$ $T^{15}$) are selected based on the lowest value of each repetitive pattern (e.g., the end of $T^0$, $T^1$, and $T^2$ interval in Figure 1b). Note that the selection of the trace segment is neither restricted to the highlighted ranges nor requires any additional knowledge regarding the data being processed or random shares. For instance, one could include intervals between $T^0$, $T^1$, and $T^2$ (according to the source code, these intervals could represent operations such as register writing). Based on our preliminary experiments, such a setting would also break the target.

**Algorithm 1** Side-channel collision attack on ASCADv2

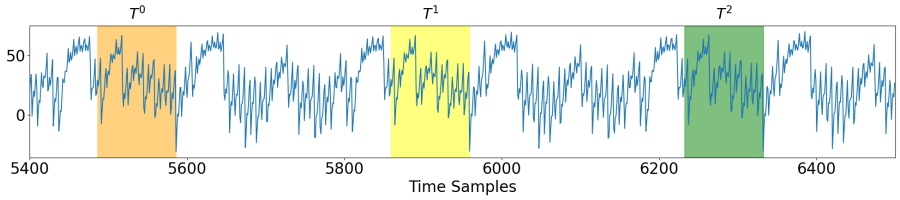
**Input:** trace segments $\mathbf{T}^i$ and $\mathbf{T}^j$ , plaintext bytes $\mathbf{d}_i$ and $\mathbf{d}_j$
**Output:** most-likely key difference $\delta^*$
1: **for** $\delta$ **in** $\mathcal{K}$ **do**
2: **indices** = $\mathsf{arg\,where}(\mathbf{d}_i \oplus \delta == \mathbf{d}_j)$
3: $diff_\delta = \mathsf{E}(\|\mathbf{T}^i_{\mathbf{indices}} - \mathbf{T}^j_{\mathbf{indices}}\|)$
4: **end for**
5: $\delta^* = \mathsf{arg\,min}_\delta\ diff$

Following Algorithm 1, we perform a side-channel collision attack with the selected trace segments. Given trace segments $\mathbf{T}^i$ and $\mathbf{T}^j$ and plaintext bytes $\mathbf{d}_i$ and $\mathbf{d}_j$, we first find the trace indices that satisfies $k_i \oplus k_j = p_i \oplus p_j$ (Equation 1) with the current $k_i \oplus k_j$ guess in an AES encryption, denoted as $\delta$. Then,

[5] https://github.com/ANSSI-FR/ASCAD/tree/master/STM32_AES_v2




(a) Leakage trace and SubBytes operations.




(b) Zoom-in view of the leakage trace and selected time intervals.

Fig. 1: An overview of the leakage trace and the target time interval.

the similarity of the two trace segments is measured with squared Euclidean distance [6] and averaged (represented by E in Algorithm 1) over indices. After looping through all possible $\delta$ candidates, the $\delta$ guess that leads to the lowest averaging difference would be the most likely candidate $\delta^*$.

The experimental result is shown in Figure 2. When attacking with 30 000 traces, only $k_1 \oplus k_2$ and $k_{13} \oplus k_{14}$ cannot be successfully recovered ($\delta$ rank are 3 and 12, respectively). In this case, one could adopt error correction methods [18,40] to recover the true key differences. With around 70 000 attack traces, all $\delta^*$ that represents the correct subkey difference can be recovered. Given this information, the entropy of the key is reduced to 256 and can be easily brute-forced.

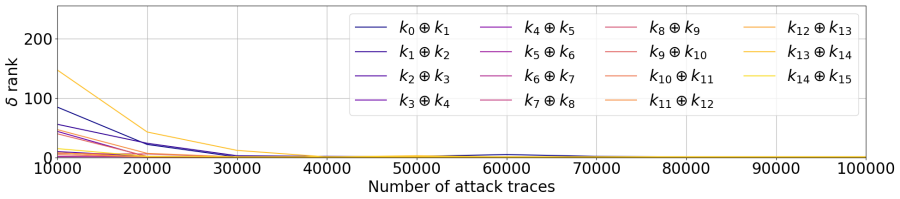



Fig. 2: Side-channel collision attack on all bytes.

### 5.2 Correlation Attack on GF(0)

The same trace segments used in the previous section, namely $\mathbf{T}^0$ to $\mathbf{T}^{15}$, are adopted for the attack presented in this section. Based on the source code, Sbox's output mask $r_{out}$ is loaded right after the SubBytes operation is finished. Therefore, the time interval of $\beta$ is selected similarly to the selection of SubBytes operations with the same pattern gap, for instance, $\mathbf{T}^{14}$ and $\mathbf{T}^{15}$ shown in Figure 3.

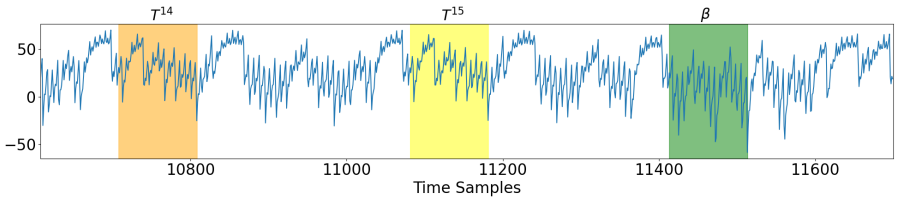



Fig. 3: The selected time intervals including the additive mask ($\beta$).

The attack method is presented in Algorithm 2. Since the goal is to correlate the $\beta$ leakages with the trace segments of SubBytes, the pairwise correlation corr is performed column-wise. Note that each column in trace segments $\mathbf{T}^i$ represents a leakage feature at a specific time location; the pairwise correlation ensures the dissimilarity of traces segments, due to different operation steps and data handling methods, less influence the correlation results. After averaging the output correlation matrix with E, the $k$ guess that leads to the highest correlation value would be the most likely candidate $k^*$.

**Algorithm 2** Correlation attack on ASCADv2

**Input:** trace segments $\mathbf{T}^i$ and $\mathbf{T}^\beta$ , plaintext bytes $\mathbf{d}_i$
**Output:** most-likely key $k^*$

**for** $k$ **in** $\mathcal{K}$ **do**
  **indices** = arg where(Sbox($\mathbf{d}_i \oplus k$) == 0)
  $\mathbf{corr}_k$ = E(corr($\mathbf{T}^i_{\mathbf{indices}}, \mathbf{T}^\beta_{\mathbf{indices}}$))
**end for**
$k^*$ = arg max **corr**

The experimental result is shown in Figure 4. Although most of the correct key does not reach a key rank of zero (the most-likely key), we see a clear convergence of the key rank with increased attack traces. Table 1 shows the detailed key rank of each subkey with 500 000 attack traces. Two subkeys are successfully recovered, and the rest (except $k_5$ and $k_6$) reach low values of the

key rank. As a rough estimation, eight times more traces would lead to successful $\delta$ recovery of all subkeys.

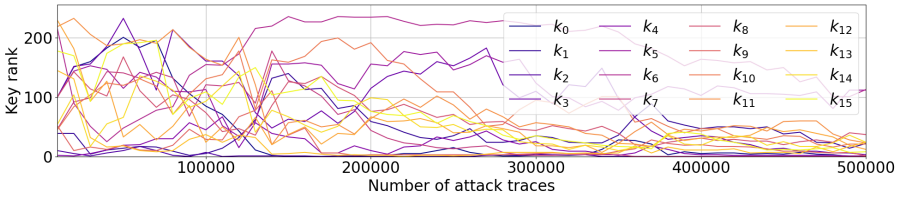



Fig. 4: Correlation attack results on all bytes.

Table 1: The rank of each subkey with 500 000 attack traces.

| | $k_0$ | $k_1$ | $k_2$ | $k_3$ | $k_4$ | $k_5$ | $k_6$ | $k_7$ | $k_8$ | $k_9$ | $k_{10}$ | $k_{11}$ | $k_{12}$ | $k_{13}$ | $k_{14}$ | $k_{15}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Key rank | 1 | 22 | 0 | 3 | 1 | 112 | 113 | 0 | 37 | 1 | 21 | 23 | 12 | 24 | 24 | 8 |

## 6 Discussion and Protection Methods

The implementation of AES by ANSSI provides an excellent example of a secure AES execution. It employs masking schemes that protect the entire AES process while shuffling serves to minimize potential leakages further. From the viewpoint of a first-order attack focusing on the leakage of a single intermediate data, this implementation exhibits robust security, only breakable under strong attack assumptions. However, this masking scheme could be easily compromised with straightforward techniques when examining second-order leakages. A solitary shuffling countermeasure could be defeated by employing more traces.

Analyzing its design reveals that the reliance on a single $\mathsf{Sbox}_m$ facilitates the attacks discussed in this paper. Despite all AES states being masked and unknown to an adversary, the deterministic association between the Sbox input and output leaves the computation of each byte susceptible to second-order attacks. One might propose the generation of sixteen distinct $\mathsf{Sbox}_m$ to facilitate byte substitution. However, in the specific case of the affine masking scheme outlined in [17], it's crucial that the multiplicative share *must* remain consistent across every byte in the state. Deviating from this principle would result in linear operations, such as AddRoundKey and MixColumns, losing their homomorphic property with the affine encoding. This, in turn, implies that each operation would necessitate 16 pre-computed lookup tables, each with a size of $256^2$, which

makes the cryptographic implementation prohibitively resource-intensive. Alternatively, implementing hiding countermeasures may be a simple and effective strategy against the proposed attacks. The proposed attacks require the comparison of trace segments. The original implementation's random shuffling significantly increases the required attack traces. Adding to this, countermeasures introducing temporal randomnesses, such as clock jitters and random branch insertion, could further complicate the process of identifying and comparing the target operations, enhancing the security of the implementation.

Addressing the second vulnerability would involve carefully redesigning the implementation, ensuring that $\mathsf{GF}(0)$ results in a random output. A more straightforward solution would involve randomizing the timing $\beta$ process, thereby reducing the correlation between the $\mathsf{Sbox}$ operation and $\beta$ leakages.

## 7 Conclusions and Future Work

In this paper, we evaluate two vulnerabilities in lookup table-based affine masking implementation, then leverage them to perform efficient second-order attacks on the ASCADv2 dataset. Specifically, we notice that some mask shares remain constant during an AES encryption, which leads to an easy cancellation of masks with a side-channel collision attack. Another vulnerability relies on implementing the Galois field multiplication, which always outputs zero when one input is zero. In this case, an adversary could choose specific traces that generate zero input. In this case, the affine masking scheme is significantly weakened, as only additive mask shares remain as the output.

Multiple aspects would be interesting to consider in future research. First, the proposed attacks rely on the single masked $\mathsf{SBox}$ used during encryption. It will be interesting to investigate the applicability of the proposed attack when two or more masked $\mathsf{SBox}_m$ are used in a cryptographic operation. Next, the proposed attacks are grounded on the squared Euclidean distance and Pearson correlation coefficient for similarity assessment. It would be interesting to explore deep learning in initiating attacks under more noisy circumstances, such as those involving desynchronization. Further, it would be compelling to study and augment the attack performance hinging on our second identified vulnerability: the zero output of the finite field multiplication. Finally, an optimal objective would be to devise innovative techniques to overcome the complexity inherent in finite field multiplication, enabling direct attacks on this dataset's intermediate data.

## References

1. Amigo, G., Dong, L., Ii, R.J.M.: Forecasting pseudo random numbers using deep learning. In: 2021 15th International Conference on Signal Processing and Communication Systems (ICSPCS). pp. 1–7. IEEE (2021)
2. Benadjila, R., Khati, L., Prouff, E., Thillard, A.: Hardened library for AES-128 encryption/decryption on ARM Cortex M4 achitecture. `https://github.com/ANSSI-FR/SecAESSTM32` (2019)

3. Benadjila, R., Prouff, E., Strullu, R., Cagli, E., Dumas, C.: Deep learning for side-channel analysis and introduction to ASCAD database. Journal of Cryptographic Engineering **10**(2), 163–188 (2020)
4. Bhasin, S., Chattopadhyay, A., Heuser, A., Jap, D., Picek, S., Ranjan, R.: Mind the portability: A warriors guide through realistic profiled side-channel analysis. In: NDSS 2020-Network and Distributed System Security Symposium. pp. 1–14 (2020)
5. Bogdanov, A.: 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. pp. 84–95. Springer (2007)
6. Bogdanov, A., Kizhvatov, I.: Beyond the limits of DPA: combined side-channel collision attacks. IEEE Transactions on Computers **61**(8), 1153–1164 (2011)
7. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 16–29. Springer (2004)
8. Bronchain, O., Standaert, F.X.: Side-channel countermeasures' dissection and the limits of closed source security evaluations. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 1–25 (2020)
9. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. pp. 398–412. Springer (1999)
10. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4. pp. 13–28. Springer (2003)
11. Coron, J.S.: Higher order masking of look-up tables. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 441–458. Springer (2014)
12. Coron, J.S., Rondepierre, F., Zeitoun, R.: High order masking of look-up tables with common shares. Cryptology ePrint Archive (2017)
13. Coron, J.S., Rondepierre, F., Zeitoun, R.: High order masking of look-up tables with common shares. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 40–72 (2018)
14. Cristiani, V., Lecomte, M., Hiscock, T., Maurine, P.: Fit the joint moments: How to attack any masking scheme. IEEE Access **10**, 127412–127427 (2022)
15. Daemen, J., Rijmen, V.: AES proposal: Rijndael (1999)
16. Dol, N.T., Le, P.C., Hoang, V.P., Doan, V.S., Nguyen, H.G., Pham, C.K.: 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). pp. 245–250. IEEE (2022)
17. Fumaroli, G., Martinelli, A., Prouff, E., Rivain, M.: Affine masking against higher-order side channel analysis. In: Selected Areas in Cryptography: 17th International Workshop, SAC 2010, Waterloo, Ontario, Canada, August 12-13, 2010, Revised Selected Papers 17. pp. 262–280. Springer (2011)
18. Gérard, B., Standaert, F.X.: Unified and optimized linear collision attacks and their application in a non-profiled setting. In: International Workshop on Cryptographic Hardware and Embedded Systems. pp. 175–192. Springer (2012)
19. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. Journal of Cryptographic Engineering **1**(4), 293–302 (2011)

20. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings 19. pp. 388–397. Springer (1999)
21. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: International Conference on Security, Privacy, and Applied Cryptography Engineering. pp. 3–26. Springer (2016)
22. Marquet, T., Oswald, E.: A comparison of multi-task learning and single-task learning approaches. Cryptology ePrint Archive (2023)
23. Masure, L., Strullu, R.: Side-channel analysis against ANSSI's protected AES implementation on ARM: end-to-end attacks with multi-task learning. Journal of Cryptographic Engineering pp. 1–19 (2023)
24. Perin, G., Wu, L., Picek, S.: Exploring feature selection scenarios for deep learning-based side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 828–861 (2022)
25. Picek, S., Heuser, A., Jovic, A., Ludwig, S.A., Guilley, S., Jakobovic, D., Mentens, N.: 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. pp. 4095–4102 (2017)
26. Picek, S., Perin, G., Mariot, L., Wu, L., Batina, L.: Sok: Deep learning-based physical side-channel analysis. ACM Computing Surveys **55**(11), 1–35 (2023)
27. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Cryptographic Hardware and Embedded Systems–CHES 2005: 7th International Workshop, Edinburgh, UK, August 29–September 1, 2005. Proceedings 7. pp. 30–46. Springer (2005)
28. Schramm, K., Paar, C.: Higher order masking of the aes. In: Topics in Cryptology–CT-RSA 2006: The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2005. Proceedings. pp. 208–225. Springer (2006)
29. Schramm, K., Wollinger, T., Paar, C.: 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. pp. 206–222. Springer (2003)
30. Staib, M., Moradi, A.: Deep learning side-channel collision attack. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 422–444 (2023)
31. Timon, B.: Non-profiled deep learning-based side-channel attacks with sensitivity analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 107–131 (2019)
32. Tunstall, M., Whitnall, C., Oswald, E.: Masking tables—an underestimated security risk. In: Fast Software Encryption: 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers 20. pp. 425–444. Springer (2014)
33. Vadnala, P.K.: Time-memory trade-offs for side-channel resistant implementations of block ciphers. In: Cryptographers' Track at the RSA Conference. pp. 115–130. Springer (2017)
34. Valiveti, A., Vivek, S.: Second-order masked lookup table compression scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 129–153 (2020)
35. Vasselle, A., Thiebeauld, H., Maurine, P.: Spatial dependency analysis to extract information from side-channel mixtures: extended version. Journal of Cryptographic Engineering pp. 1–17 (2023)

36. Veyrat-Charvillon, N., Medwed, M., Kerckhof, S., Standaert, F.X.: Shuffling against side-channel attacks: A comprehensive study with cautionary note. In: Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18. pp. 740–757. Springer (2012)
37. Wu, L., Perin, G., Picek, S.: The best of two worlds: Deep learning-assisted template attack. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 413–437 (2022)
38. Wu, L., Perin, G., Picek, S.: Hiding in plain sight: Non-profiling deep learning-based side-channel analysis with plaintext/ciphertext. Cryptology ePrint Archive (2023)
39. Wu, L., Picek, S.: Remove some noise: On pre-processing of side-channel measurements with autoencoders. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 389–415 (2020)
40. Wu, L., Tiran, S., Perin, G., Picek, S.: An end-to-end plaintext-based side-channel collision attack without trace segmentation. Cryptology ePrint Archive (2023)
41. Zaid, G., Bossuet, L., Habrard, A., Venelli, A.: Methodology for efficient CNN architectures in profiling attacks. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 1–36 (2020)