MIDSCAN: Investigating the portability problem for cross-device DL-SCA

Lizzy Grootjen[0009−0007−4305−0285](✉), Zhuoran Liu[0000−0003−0049−7080], and Ileana Buhan[0000−0001−5494−9164]

Digital Security Group, Radboud University, Nijmegen, The Netherlands
{lizzy.grootjen,zhuoran.liu,ileana.buhan}@ru.nl

Abstract. In deep learning side-channel analysis, a neural network is employed to develop a profile of our target device. Data from a similar dummy device is used to construct the profile. However, when the profiling device differs from the target device, the profile may not be accurate enough for a successful attack. This study examines the effect of manufacturing-induced inter-device discrepancies across 14 identical 32-bit 32STMF303 devices. To map the manufacturing discrepancies for these devices, we create a tool called MIDSCAN - Manufacturing-Induced Discrepancies SCAN. Our analysis revealed that the 14 ChipWhisperer 32-bit devices have limited manufacturing discrepancies. In a multilayer perceptron setup, the manufacturing discrepancies between the profiling and target devices only affected the attack performance. In this case, devices that showed more discrepancies based on correlation and difference than the profiling device need additional traces for a successful attack. Manufacturing discrepancies between profiling and attack devices do not affect attack performance for convolutional neural network architectures. No additional attack traces were needed to perform a successful attack. Our findings indicate that statistical metrics, as implemented by MIDSCAN, can estimate inter-device discrepancies for identical devices. Finally, we found that manufacturing discrepancies are limited for 32-bit STM32F303 ChipWhisperer targets, eliminating the need for additional measures for cross-device deep learning attacks.

Keywords: Side-channel analysis · Deep learning · Portability problem · Inter-device variations · Cross-device attacks

1 Introduction

Today, communication is secured with the use of cryptographic algorithms. However, side-channel attacks form a realistic threat against embedded devices. These attacks can obtain keys from mathematically secure cryptographic implementations [14][23]. An attacker utilises leakage omitted by the physical properties of the device, *side-channels*, such as timing, power, and electromagnetic emanation, to obtain secret information. There are two main types of side-channel attacks: non-profiled and profiled attacks. In non-profiled attacks, many traces

are collected from the target device to perform a statistical analysis, while in profiled attacks, the attacker creates a profile from the target device to leverage such an attack [21]. Device profiles can be created in various ways. The most famous attack is the template attack, which assumes the leakage follows a multivariate Gaussian distribution and is considered one of the strongest attacks in an information-theoretic sense [23][6]. Another option is to create a profile based on power traces from a similar device. A greater similarity between the profiling device and the target device simplifies the success of an attack [27].

Even with similar profiling and target devices, discrepancies, such as manufacturing discrepancies, environmental factors, and measurement setup, can result in an inaccurate template, possibly causing the attack to fail. This gap between the profile and the target device is called the *portability problem* [21]. Device and key variations can influence device portability and, therefore, attack performance [4]. These discrepancies occur not only between homogeneous and heterogeneous devices, but also between identical devices [27].

Various works have investigated the portability problem on identical devices for deep learning. Cao et al. explored how unsupervised domain adaptation, a type of transfer learning, facilitates cross-device attacks on 8-bit XMEGA devices [5], which are otherwise unfeasible without transfer learning. Golder et al. reported similar findings as Cao et al., demonstrating that additional measures were necessary to execute a cross-device attack on 30 chipWhisperer ATMEGA 8-bit devices, introducing their DTW-PCA-MLP solution [9]. Wang et al. reported the same results as Golder et al. on two other 8-bit ATMEGA devices where a cross-device attack was not directly possible [25]. Ninan et al. demonstrated that a cross-device attack is feasible on a 32-bit ChipWhisperer using EM traces without additional measures if the location of the probe is consistent [18]. This poses the question whether a cross-device is also possible in 32-bit devices with power traces, which are generally less noisy, easier to capture, and contain more circuit activity compared to EM traces [1]. Additionally, power traces are often used within the SCA community, as they do not require special equipment [23]. Finally, the potential for a cross-device attack varies between 8-bit and 32-bit devices.

The effect of manufacturing discrepancies among identical 32-bit devices on deep learning performance has not been studied, and it is unclear whether such variations should be considered. Therefore, we aim to answer the research question: How do manufacturing discrepancies influence the portability gap of 32-bit identical ChipWhisperer devices for cross-device deep learning attacks? To answer this question, first, we explore the variability arising from manufacturing discrepancies between 14 identical STM32F ChipWhisperer devices equipped with a Cortex-M4 chip. Second, we explore the impact of these discrepancies on the portability issue in deep learning profiling attacks. Lastly, we open-source our MIDSCAN tool to make it usable for any group of identical devices to examine manufacturing discrepancies[1]. In short, our main contributions are as follows.

[1] https://gitlab.science.ru.nl/lgrootjen/midscan

1. Design a framework to compare our 14 identical 32STMF303 ChipWhisperer devices with each other on manufacturing discrepancies;
2. Create a tool (MIDSCAN) which implements the proposed framework;
3. Demonstrate that statistical metrics can assess manufacturing discrepancies in devices and indicate the impact on deep-learning model performance;
4. Show that deep learning architectures can handle identical device discrepancies of these 32-bit devices;

2 Background

2.1 Profiled Side-channel Analysis & leakage

A profiling side-channel attack assumes the attacker has complete access to a profiling device. Here, the attacker controls the plaintext and key input, enabling the collection of device traces to form a profile. Subsequently, the attacker collects traces of the target device containing unknown plaintexts and keys. The constructed profile is used to attack the target device to recover the keys. To analyze the leakage, a leakage model and a sensitive intermediate value y are selected. In AES, the sensitive value is selected as the s-box output.

$$y = Sbox(P \oplus k) \tag{1}$$

This value relies on plaintext and key, and its byte-wise characteristic within the AES algorithm enables a divide-and-conquer attack. Several approaches are available to model the value of y. The Hamming weight or the Hamming distance is a commonly used leakage model, resulting in a 9-class leakage model. Another option is the identity model, where we use a 256-class leakage model, one class for each possible value of y. In the Hamming weight model, the complexity of analysis is reduced because of the reduced classes. However, the Hamming weight leakage model does not guarantee that all classes are evenly balanced. For the identity leakage model, there is a uniform distribution in all 256 classes. Depending on the technique used for analysis, one leakage model might be more suitable than the other.

2.2 The portability problem

The portability issue refers to the difference in the leakage distribution between profiling and attacking devices, which is affected by how similar these devices are. In this work, we refer to manufacturing discrepancies *as the difference in the results in the set of metrics used to compare two devices.* The relationship between the profiling device and the target device based on their discrepancies can be sorted into four categories [27]:

1. **Same device**: When the same device is used for both profiling and attack, only variations in the measurement setup impact the leakage distribution. There are no manufacturing discrepancies, as the device used is the same for profiling and attacking. Practically speaking, this attack scenario is unrealistic.

2. **Identical/clone devices**: When the profiling device and target device are the same, they are identical or clone devices. This indicates that the architecture, microarchitecture, configurations, and manufacturer are the same. Differences in measurement setups and PCB manufacturing lead to variations between identical devices, as noted in [25]. This scenario is often used in lab setups.
3. **Homogeneous devices**: In a homogeneous device relationship, the profiling and attacking devices share the same architecture and manufacturer, yet differ in microarchitecture and configurations.
4. **Heterogeneous devices**: Here, the profiling and target devices vary entirely, encompassing both diverse manufacturers and architectures. Leakage through side channels is influenced by the microarchitecture of a device [3], resulting in different leakage distributions between heterogeneous devices.

2.3 Evaluation of power trace signal quality

Collected power traces can contain noise. Manufacturing discrepancies, measurement setup, and countermeasures can cause this noise. Countermeasures reduce the signal-to-noise (SNR) ratio to make side-channel attacks harder. The signal-to-noise ratio can be used to estimate the amount of signal in a trace compared to the noise. The signal-to-noise ratio is based on the variance of the signal and the variance of the noise [17].

$$SNR = \frac{Var(Signal)}{Var(Noise)} \tag{2}$$

2.4 Deep learning

Neural networks have been applied in various domains, such as image classification and speech recognition. There are different architectures available, but they all perform a similar task: try to find the minimum in the landscape of a function. Gradient descent is the most commonly used technique for iteratively finding this minimum. For a supervised learning problem, the dataset consists of data points and their labels. The data used to find this optimum is divided into three parts: the training set, the validation set, and the test set. The most commonly used neural network architectures in deep learning side-channel analysis are the multilayer perceptron and the convolutional neural network [21][22][11].

Multilayer perceptron The multi-layer perceptron is composed of an input layer, one or more hidden layers, and an output layer, with nodes in each layer. Each layer in a fully-connected neural network links every node to the subsequent layer's nodes. The output of a neuron k in a layer is the sum of the weighted inputs n of the prior layer plus a bias b. This is typically represented in a matrix.

$$z_k = \sum_{i=1}^{n} w_i \cdot x_i + b = W \cdot x + b \tag{3}$$

An activation function can be applied to make the multilayer perceptron able to deal with non-linearity. This is usually the ReLU function, as it only keeps positive values within the network.

$$f_{relu}(x) = \max(0, x) \tag{4}$$

In a classification problem, the last layer is the number of classes in the problem. Each node denotes the score for each class. The softmax layer ensures that the output neuron values total 1, representing a probability distribution.

$$f(x)_i = \frac{e_i^x}{\sum_j e^{x_j}} \tag{5}$$

Convolutional neural networks Created initially for image processing, convolutional neural networks, the convolutional layers are inspired by the animal visual cortex [2]. Later, they proved beneficial for 1D data like audio processing [12]. Within the side-channel research field, convolutional neural networks based on the VGG16 architecture are noted for their robustness against desynchronised power traces [11]. Convolutional neural networks share a similar structure to multilayer perceptrons: they consist of an input layer, at least one hidden layer, and an output layer. Hidden layers generally comprise blocks, each containing a convolutional layer and a subsequent pooling layer. CNNs feature a final fully-connected layer for classification. Every convolutional layer includes a kernel determining the window size and a stride specifying how far the window moves. In convolutional layers, the linear layers share weights across a plane. The core convolution between the signal and the weight matrix is as follows:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{6}$$

The convolutional layers are often followed by a pooling layer. Pooling layers decrease dimensionality, thus needing fewer neurons in the subsequent network layers. The pooling layer downsamples a segment of the feature map. Average-Pooling computes the mean of the values within the receptive field of the feature map, while MaxPooling selects the maximum value of the feature map. The following equation represents the output of a MaxPooling layer.

$$MaxPool_k = \max(\{x_{k'} | k' \text{ in the receptive field of } k\}) \tag{7}$$

2.5 Evaluation of a DL-SCA

In deep learning, accuracy is commonly used to assess model performance. For image classification, each predicted label of a new image is correct or incorrect. The label is derived from the probability vector - a vector containing the probability for each class -, selecting the class with the highest probability. For side-channel analysis, accuracy is less suitable as a metric for model performance

[20]. In a probability vector, even if the correct key ranks second, it can still facilitate a successful attack because the attacker can have multiple attack traces available. The community frequently uses *key rank* or *guessing entropy*, which are plotted against the number of required attack traces. Where the key rank is usually used for a known-key analysis, the guessing entropy is used to estimate the remaining workload after a side-channel attack [19]. In this work, the key rank is used as we perform a manufacturing discrepancies comparison for a cross-device attack. The key rank is based on the key guessing vector g on a T_a number of timesamples. The key guessing vector is calculated based on the log-likelihood principle, where $\hat{p_{ij}}$ is the estimated probability of a key candidate i using the sample j.

$$g_i = \sum_{j=1}^{T_a} log(\hat{p}_{ij}) \tag{8}$$

$$rank_{guess} = \text{position of } guess\text{'s score in } g \tag{9}$$

3 Related Work

Kocher et al. are the first to introduce a side-channel attack, where RSA is attacked using Simple Power Analysis (SPA) and Differential Power Analysis (DPA) [13]. After that, various other attacks are developed, such as Correlational Power Analysis (CPA) and Template Attacks [6][23]. In a Template Attack, the attacker creates a profile by assuming that the leakage follows a multivariate Gaussian distribution. Later, deep learning techniques were applied to learn the profile based on a similar device [15][16][10]. In recent work, the multilayer perceptron and the convolutional neural network are the most promising architectures in deep learning for side-channel analysis [22][11].

Using a similar device for profiling and attacking implicitly assumes they share the same leakage distribution [21]. This may not always be true due to inter-device differences [4]. Side-channel leakage is greatly influenced by a device's micro-architecture [3]. Current research aims to bridge the portability gap by introducing new solutions. For example, Das et al. [7] investigated how data augmentation from different identical profiling devices improves performance for a deep learning cross-device attack. Bhasin et al. [4] conducted research on the effect of inter-device and inter-key variations for attack performance. They suggested a similar solution as Das et al., the Multiple Device Model, which improves the training dataset by incorporating traces from different devices to address for these variations. Zhang et al. [27] showed in different device families how inter-device variations influence attack performance and proposed a preprocessing method, FL-PA, to deal with device variations of homogeneous and heterogeneous devices. Several studies have explored how transfer learning can help bridge the portability gap. Thapar et al. developed TranSCA, a transfer learning tool to address the portability gap in heterogeneous traces from simulated FPGA implementations [24]. Yu et al. similarly applied Meta-Transfer

learning to enhance a cross-device attack on homogeneous 32STMFx devices and a cross-device attack involving the heterogeneous ATMEGA 8-bit device and 32STMFx 32-bit devices [26]. Genevey-Metat et al. applied transfer learning by retraining the network during the fine-tuning stage to target 32-bit homogeneous devices [8]. Cao et al conducted a cross-device attack on identical 8-bit XMEGA devices utilizing unsupervised domain adaptation, discovering that this transfer learning technique enhances cross-device attack performance [5].

Most of the previous research has focused on covering the portability gap caused by the measurement setup, measurement domain, and architectural differences of devices. Few works have investigated the portability gap on identical devices. Ninan et al. focused on the variability in measurement setup and showed that the variation between devices increases when different probe locations are used compared to the profiling device [18]. They also found that for their 32-bit STM32F3 devices with the same probe location, their deep learning model, a convolutional neural network, was able to recover the keys without any additional pre-processing. For their 8-bit devices, it was feasible but required more attack traces. According to Golder et al. [9], for 30 identical 8-bit devices, inter-device variations affect attack performance. They used a convolutional neural network as a benchmark and compared it to their proposed solution, DTW-PCA-MLP, to account for the inter-device variations. Both authors showed different results on the portability gap between identical devices. The main difference between these two works is the target devices (8-bit vs. 32-bit) and their measurement technique (ChipWhisperer power consumption vs. EM probes). Interestingly, Ninan et al. found that a cross-device attack with EM-probes is possible without adjustments if the probe is positioned in the same area. However, when dealing with 8-bit ChipWhisperer devices, Golder et al. found that adjustments were needed for a cross-device deep learning attack. This suggests that manufacturing discrepancies can have an influence on attack performance, depending on the architecture of the devices and the chosen deep learning architecture. Table 1 provides an overview of related work concerning cross-device deep learning attacks on identical devices. This table shows that no other study has been conducted on a large set of 32-bit devices.

Table 1. Overview of the related work on the portability issue for identical devices for a cross-device deep learning attack.

Reference	Device	Arch.	Nr. devices	Neural network	Side-channel
Das et al. [7]	XMEGA	8-bit	8	MLP	Power
Golder et al. [9]	XMEGA	8-bit	30	MLP & CNN	Power
Cao et al. [5]	XMEGA	8-bit	8	CNN	Power
Wang et al. [25]	XMEGA	8-bit	2	MLP & CNN	Power
Ninan et al. (1/2) [18]	XMEGA	8-bit	2	MLP & CNN	EM
Ninan et al. (2/2) [18]	STM32F3	32-bit	2	MLP & CNN	EM
Our work	STM32F303	32-bit	14	MLP & CNN	Power

4 Capturing manufacturing discrepancies

4.1 Components within a power trace

The device's power trace is connected to the algorithm in use. There are different approaches on how to determine the components of a power trace. In this study, we follow the definition of Mangard et al. [17], which is repeated below:

> "Each point in a power trace can be modelled as the sum of an operation-dependent component P_{op}, a data-dependent component P_{data}, electronic noise $P_{elnoise}$, and a constant component P_{const}."

$$P_{total} = P_{op} + P_{data} + P_{elnoise} + P_{const} \tag{10}$$

P_{const} is considered the constant power supply for a device and remains consistent across identical devices. If the devices are homogeneous instead of identical, P_{const} may vary. P_{data} is the component of the power trace influenced by the data, affecting overall power usage. P_{op} refers to the power required to execute operations on the device. These operations may rely on the data, depending on the op optimizations used within the chip. If clone devices have the same data input, their operations are equal. Bhasin et al. demonstrated that inter-key variations significantly influence the profiling phase[4]. $P_{elnoise}$ refers to the manufacturing process and measurement setup discrepancies.

A profile has to accommodate all of these variables, depending on the relationship between the two devices. Earlier studies have shown that extra steps might be necessary to address inter-device variations caused by differences in measurement setups, manufacturing, and the devices themselves [7][9].

To isolate the impact of device manufacturing discrepancies, we believe it is essential to accurately assess the electrical noise component $P_{elnoise}$. To achieve this, we keep P_{const}, P_{data}, and P_{op} uniform between devices. With identical ChipWhisperer boards used in all setups, P_{const} remains stable. Employing fixed keys and seeded plaintext data ensures P_{op} and P_{data} remain consistent across devices. This yields a single variable, $P_{elnoise}$, which reflects the manufacturing variations among the devices. To our knowledge, this work is the first to model manufacturing discrepancies by isolating $P_{elnoise}$.

4.2 MIDSCAN: Manufacturing-Induced Discrepancies SCAN

To facilitate the evaluation of $P_{elnoise}$, as described in Section 4.1, we developed a tool called MIDSCAN (Manufacturing-Induced Discrepancies SCAN). This tool includes scripts in three categories: data collection, device analysis, and deep learning attack. A schematic overview of MIDSCAN is visualised in Figure 1. The data collection category consists of two scripts to collect data from ChipWhisperer: one script collects traces for the profiling device, and the other collects traces from the target devices. We use seeded random plaintext for our target devices to maintain a consistent P_{data} across all devices. We also collect

500 traces from a fixed plaintext and key to conduct a cross-device analysis of manufacturing-induced variations. The device analysis component includes the analysis of the point of interest between the profiling device and target devices, as well as a cross-device analysis.

The cross-device analysis calculates manufacturing discrepancies based on correlation, difference, standard deviation, time sample distribution, timesample outliers, and signal-to-noise ratio. The deep learning component assists in executing a deep learning side-channel attack. The initial script trains three deep learning models chosen from the literature on our profiling device. The second script executes the cross-device deep learning attack and determines the key rank for each device. This key-rank is then used as an additional metric to assess cross-device attack performance when dealing with device discrepancies. This tool has been designed for use with our 14 available devices, but it is extendable to other (ChipWhisperer) devices.

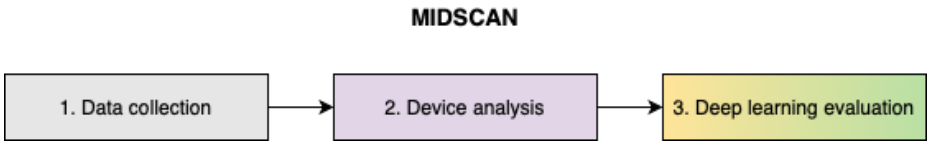

Fig. 1. Schematic overview of MIDSCAN

5 Experimental setup

We gathered data from 14 ChipWhisperer Lite boards, each featuring a 32-bit STM32F303 ARM Cortex M4 target. These chips use NewAE Solutions' tinyAES-128 implementation. Among these 14 devices, one is randomly selected as the profile device; in this study, it is device A. As this work focuses on the effect of manufacturing differences, no countermeasures were implemented to protect against side-channel attacks.

5.1 Dataset generation

The study is divided into two sections: device analysis and deep learning attack, with a dataset collected for each part. All devices share an identical, fixed key in this study. During the device analysis part, we collected 500 traces with fixed plaintext input, each trace consisting of 12,000 timesamples. Throughout the remainder of the paper, this dataset is called CW_{fixed}. In the deep learning part, we collected 10,000 traces with seeded plaintexts, each trace again consisting of 12,000 timesamples. This set of plaintexts is used on all devices, maintaining consistency in P_{op} and P_{data}. For the remainder of the paper, this dataset is referred to as CW_{seeded}. Figure 2 illustrates the structure of the dataset. Due to

the size of the datasets, it is not publicly available. However, MIDSCAN as well as this paper offer all the information needed on the datasets.

	A	A	B	C	D	E	F	G	H	I	J	K	L	N	O
CW_fixed	-	500	500	500	500	500	500	500	500	500	500	500	500	500	500
CW_seeded	50k	10k	10k	10k	10k	10k	10k	10k	10k	10k	10k	10k	10k	10k	10k

Legend: Device analysis | ML_train | ML_test

Fig. 2. Overview of the ChipWhisperer dataset with the number of traces gathered for each device.

5.2 Point of interest

In this work, we focused on attacking target byte 0 of the first AES round. For a comprehensive overview of inter-device discrepancies, we concentrated on three aspects of our power traces: the discrepancies across all 12,000 samples, the discrepancies within the defined point of interest window, and the discrepancies in a 10-sample window zoomed in on our target byte 0. The point of interest window is identified by calculating the correlation between the timesamples and the Hamming weight leakage model. For this, the dataset CW_{seeded} is used.

5.3 Deep learning architectures

The cross-device attack performance is evaluated for each device using three different deep learning architectures. The first model, the multilayer perceptron (MLP), introduced by Das et al. [7], is named `mlp_das` in this work. The second model, a deep convolutional neural network, was proposed by Kim et al. [11] and is called `cnn_kim` in this work. The third model, a shallow convolutional neural network, was proposed by Golder et al. [9] and is named `cnn_golder` in this work. The architectures of these networks are shown in 2.

In these models, no additional measures are taken to facilitate a cross-device attack. For the deep learning attack, the identity leakage model is used, resulting in a 256-bit classifier. To evaluate attack performance, the key rank metric is used as described by [22][19]. The architectures are the same as those proposed by the authors. The epochs, batch size, optimiser, and learning rate are adjusted to our dataset. For the convolutional neural networks, those happen to be the same as the original authors. In table 3, these parameters are highlighted for each of these architectures. The aim is to investigate how discrepancies in manufacturing affect attack performance on identical 32-bit devices.

Table 2. Overview of the deep learning architectures. All Dense and Conv1D layers have a ReLU activation function, unless stated otherwise.

	`mlp_das` [7]	`cnn_kim` [11]	`cnn_golder` [9]
Input Layer		BatchNormalization()	
First block	Dense(200) BatchNormalization() Dropout(0.1)	Conv1D(8,3) MaxPooling1D(2) BatchNormalization()	Conv1D(70,60)
Second block	Dense(200) BatchNormalization()	Conv1D(16,3) MaxPooling1D(2)	Conv1D(70,60) MaxPooling1D(3)
Third block		Conv1D(32,3) MaxPooling1D(2) BatchNormalization()	
Fourth block		Conv1D(64,3) MaxPooling1D(2)	
Fifth block		Conv1D(64,3) MaxPooling1D(2) BatchNormalization()	
Sixth block		Conv1D(128,3) MaxPooling1D(2)	
Seventh block		Conv1D(256,3) MaxPooling1D(2) BatchNormalization()	
Eighth block		Conv1D(256,3) Dropout(0.5)	
Classification block	Dense(256, softmax)	Flatten() Dense(512) Dropout(0.5) Dense(256, softmax)	Flatten() Dropout(0.2) Dense(150) BatchNormalization() Dropout(0.1) Dense(256, softmax)

Table 3. Overview of the parameters for the neural network architectures

	`mlp_das`	`cnn_kim`	`cnn_golder`
epochs	20	75	20
batch size	256	256	256
optimizer	Adam	Adam	Adam
learning rate	0.00001	0.001	0.001

5.4 Evaluation metrics

To investigate manufacturing discrepancies, the dataset CW_{fixed} is used. The metrics are computed on all three intervals as discussed in the previous section. The first two evaluation metrics are the difference and the correlation between the average power trace across the devices. This results in a heatmap showing which devices are more similar based on these metrics and how these similarities vary over different intervals. The hypothesis is that for devices more similar and more correlating to the training device, it is easier to perform a cross-device

attack. Then, the standard deviation and the boxplot distribution for each device is calculated. This metric is used to determine outliers between our devices. Finally, we visualise the number of timesamples that fall in the extremes of a normal distribution, also known as 3σ, for the three intervals. This statistical measure has been taken from the research of Golder et al. and is used to detect outliers on a timesample view. Before performing the cross-device attack, the signal-to-noise ratio is calculated for each device. To perform the signal-to-noise ratio analysis, the dataset CW_{seeded} is used. We aim to explore whether discrepancies in manufacturing can affect the signal-to-noise ratio. Lastly, the key rank as introduced in Section 2.5 is used to determine the attack performance.

6 Results and discussion

6.1 Device analysis

Before starting the device analysis, we determine the point of interest. In line with other studies, we choose a window of 700 samples. We use the Hamming weight leakage model to associate s-box values with power traces, identifying leakage and thus the point of interest. Our POI analysis shows the s-box calculation for target byte 0 at timesample 1326 and for byte 15 at 1973. These results are shown in Figure 3. We investigate whether all test devices share the same leakage points. The results of this experiment are shown in Figure 5. Devices G and O are unique in having a leaky point other than the training device A, differing by only a single timesample. Thus, we conclude that the point-of-interest window for all devices spans from timesample 1300 to 2000.

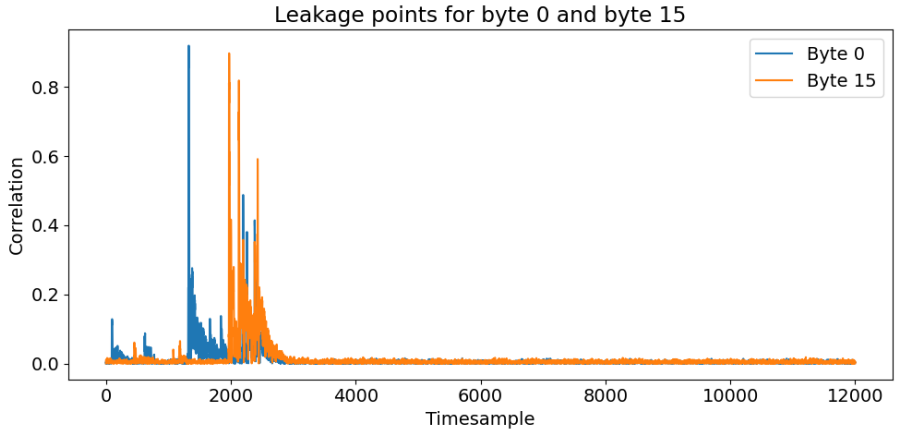

Fig. 3. Graph of leaky points on byte 0 and byte 15 for training device A

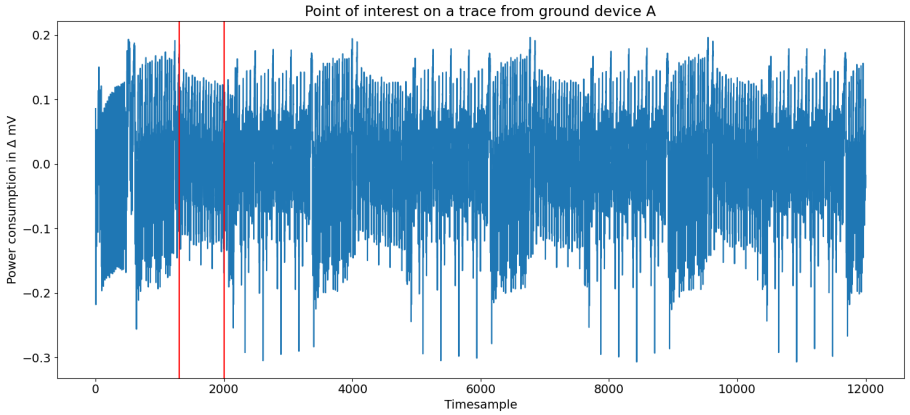

Fig. 4. The chosen POI based on the location of the leakage of byte 0 and byte 15

Determining point of interest for target byte 0: correlating timesamples

	A	B	C	D	E	F	G	H	I	J	K	L	N	O
Highest correlation	1326	1326	1326	1326	1326	1326	1325	1326	1326	1326	1326	1326	1326	1325
Second highest correlation	1327	1325	1325	1325	1325	1325	1326	1325	1325	1327	1325	1325	1325	1326
Third highest correlation	1325	1327	1327	1327	1327	1327	1327	1327	1327	1325	1327	1327	1327	1327

Fig. 5. The timesample showing the most leakage correlated to byte 0 for each device

For each combination of devices, the correlation and differences are computed. The results are shown in Figures 6 and 7. Across the whole trace, device E shows the weakest correlation with our training device A, followed by devices C and O. These results remain the same when taking our POI into account. Zooming in on the target byte 0, device O shows more correlation with our training device. When evaluating the differences across devices, again device E shows to have the biggest difference compared to device A across the whole trace. Device H also shows a larger difference compared to device A. When zooming in on our POI, these findings remain roughly the same compared to our entire trace. Finally, zooming in at the byte level, device E shows less difference compared to the other two timesample windows. The heatmap for correlation varies only 0.2%, indicating that the devices are very similar. Without any feedback loop to the manufacturing process, it is not possible to investigate why these devices are so similar. A possible hypothesis could be that the variations within the data, which are kept constant in this study, have a greater influence on the device similarities compared to the manufacturing discrepancies.

For each device, we calculated the standard deviation. The results of this experiment are shown in Figure 8. The standard deviation is relatively small, and all the devices have a similar standard deviation. The pattern remains the same across different timesample windows. Figure 9 shows that each device has

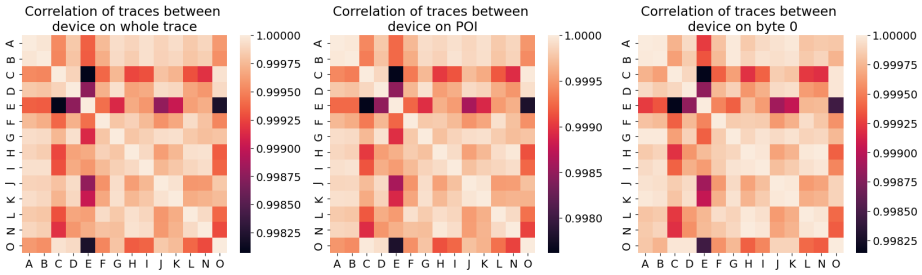

Fig. 6. Heatmap displaying the correlation between the devices on different timesample intervals

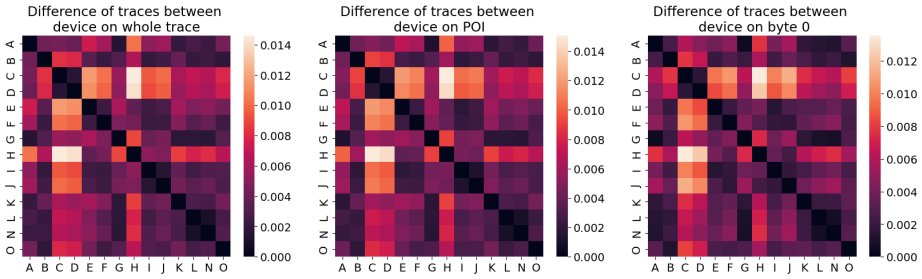

Fig. 7. Heatmap displaying the difference between the devices on different timesample intervals

a similar power distribution. Across the whole trace, the points below the value -0.2 catch attention. These appear to be outliers, but this behaviour is the same across all devices. Interestingly, timesamples that are outliers do not occur in the windows of POI and byte 0, parts where the s-box computation takes place. From these two figures, we can conclude that there are no devices that have a different behaviour based on the power distribution or standard deviation.

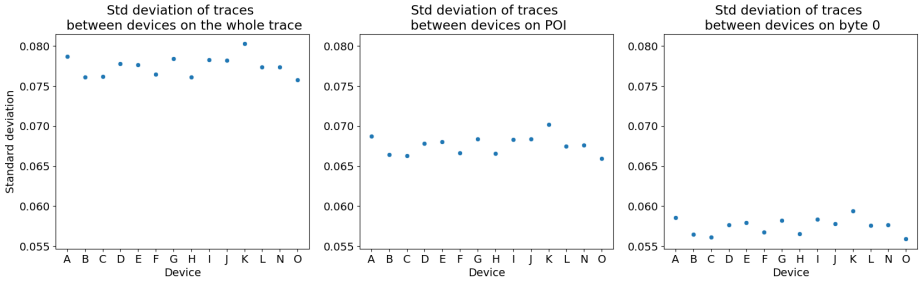

Fig. 8. Scatter plots showing the standard deviation of the devices on different timesample intervals

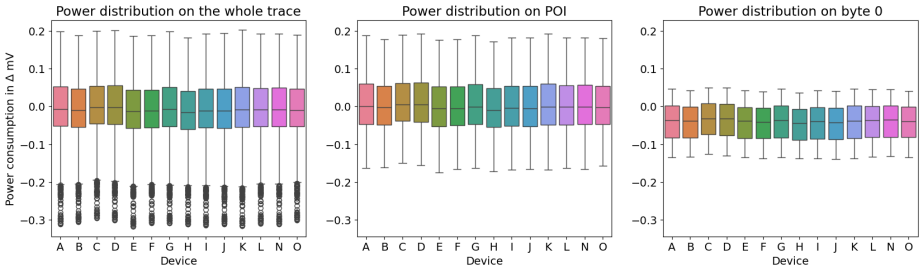

Fig. 9. Boxplot of the power consumption of the devices

Figure 10 shows the count of timesamples in the power traces that lie in the extremes of a normal distribution, that is, 3σ. This measure is adapted from Golder et al [9]. As can be seen, the outliers in the power trace do not lie within our point of interest. This is in line with the observations in Figure 9. Again, from this plot, we conclude that there is no device that is an apparent outlier.

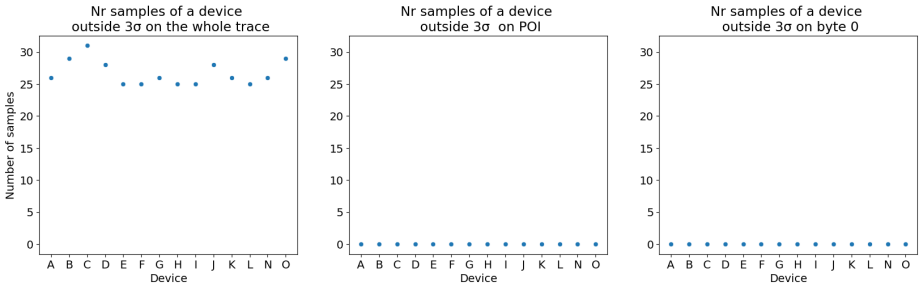

Fig. 10. Scatterplot displaying the number of timesamples outside 3σ on different timesample intervals

Finally, Figure 11 shows the signal-to-noise ratio of each device. Device G has the highest signal-to-noise ratio, and devices H and I have the lowest signal-to-noise ratio. The pattern in the signal-to-noise ratio is roughly the same for all devices. From this section, we conclude that the manufacturing discrepancies of these 14 ChipWhisperers are limited.

6.2 Deep learning attack

This experiment was performed with the CW_{seeded} dataset. Firstly, three neural network architectures were trained on device A. Then, all devices were attacked using the three trained models. The results of this cross-device attack are shown in Figure 12. Both convolutional neural networks, `cnn_kim` and `cnn_golder`, effectively handled the manufacturing discrepancies among the 14 devices. For the

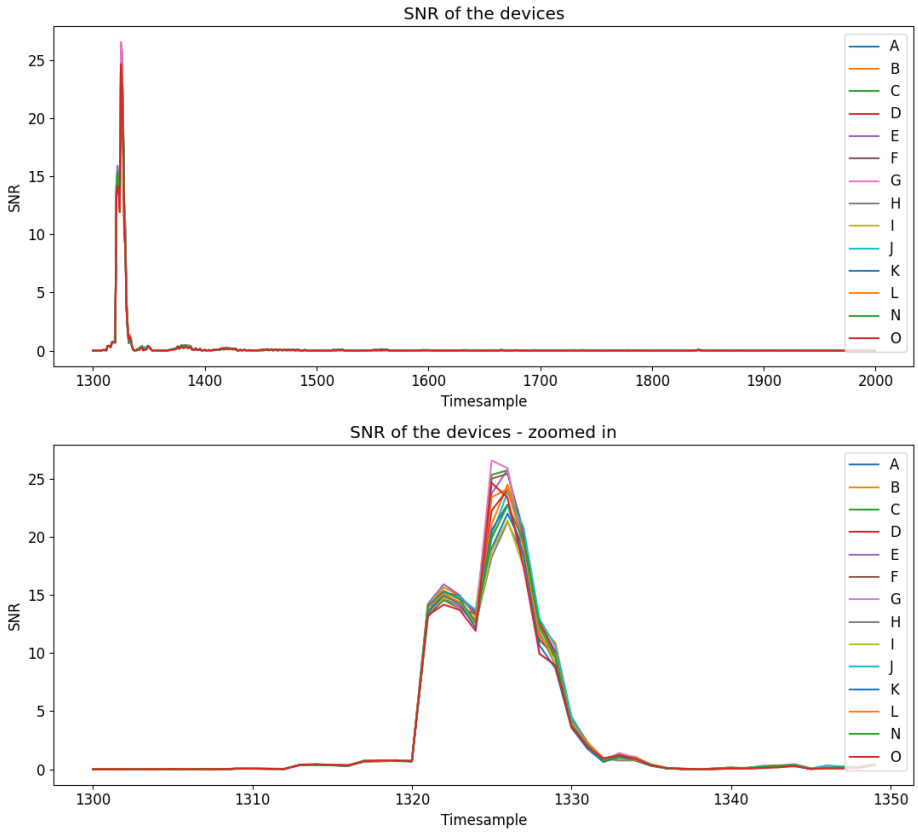

Fig. 11. Signal-to-noise ratio for each device

`mlp_das` multilayer perceptron, additional attack traces are required to successfully recover the key. In particular, devices E and H needed more attack traces for a successful key recovery.

These are the same devices that had larger interdevice discrepancies compared to training device A, based on the difference and correlation. There is no visible effect on the differences between the signal-to-noise ratio for a cross-device attack. This is most likely because the signal-to-noise ratio is already high, indicating more signal than noise. This also shows that the traces captured by the ChipWhisperer board are very clean.

7 Conclusions and future work

This research presented a tool with evaluation metrics to assess manufacturing discrepancies. From the experiments, we conclude that manufacturing discrepancies hardly influence attack performance for 32-bit 32STM303 ChipWhisperer

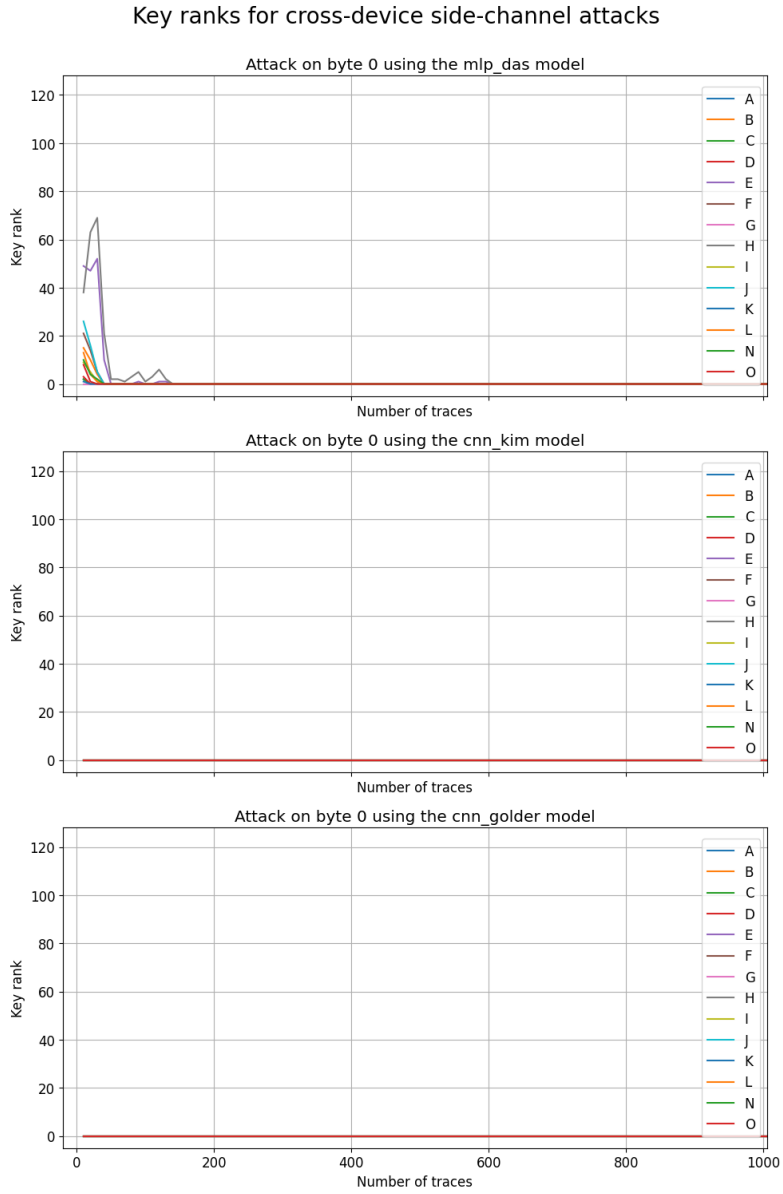

Fig. 12. The key ranks for three different neural network architectures, cross-device attacked

devices. For MLP, more attack traces were needed for a successful key recovery. However, the amount of additional traces needed was only roughly 75 traces in a worst-case scenario. Devices with larger discrepancies compared to the training

device were also devices that required more attack traces. This indicates that a cross-device attack using a multilayer perceptron without additional measures is still possible, at the cost of a few extra attack traces. The high signal-to-noise ratio of the ChipWhisperer devices did not make up for the inter-device variations. Using a convolutional neural network, one attack trace is required for identical 32-bit 32STMF303 devices in a cross-device attack, eliminating the need to consider additional measures to account for inter-device variations.

Our results align with those of Ninan et al., who demonstrated that a 32-bit cross-device attack is feasible with the same probe location. Research by Golder et al., Wang et al., and Cao et al. demonstrated that direct cross-device attacks are not successful for 8-bit identical devices. The analysis by Golder et al. revealed higher inter-device variations and wider power distributions between devices, indicating that a cross-device attack might not be feasible. Based on our work and similar previous work, we conclude that a cross-device deep-learning attack without additional measures to cover the portability gap is possible for 32-bit devices but not necessarily for 8-bit devices. In conclusion, our analysis reveals that manufacturing discrepancies detected by MIDSCAN minimally impact the deep learning attack performance of the multilayer perceptron.

For future work, a side-by-side comparison between 8-bit and 32-bit devices for a cross-device attack using the power side-channel can verify this hypothesis. In addition, transformer models can be evaluated for the effectiveness of cross-device attacks. Another aspect to investigate is whether manufacturing discrepancies between identical devices play a greater role if the signal-to-noise ratio of the devices is lower. Finally, the effect of data variations compared to manufacturing discrepancies can be investigated when doing a cross-device attack.

Acknowledgments. This work was (in part) supported by Dutch Research Council (NWO) through the PROACT project (NWA.1215.18.014), TTW PREDATOR project 19782, TTW BASES project 20858 and the CiCS project of the research programme Gravitation under the grant 024.006.037. We thank the reviewers for their suggestions and ideas to improve the paper.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Amar, M., Navanesan, L., Sayakkara, A.P., Oren, Y.: Waves of Knowledge: A Comparative Study of Electromagnetic and Power Side-Channel Monitoring in Embedded Systems. In: Chen, Y., Lin, C.W., Chen, B., Zhu, Q. (eds.) Security and Privacy in Cyber-Physical Systems and Smart Vehicles. pp. 158–170. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-51630-6_11
2. Arbib, M.A. (ed.): The Handbook of Brain Theory and Neural Networks. The MIT Press (Nov 2002). https://doi.org/10.7551/mitpress/3413.001.0001

3. Arora, V., Buhan, I., Perin, G., Picek, S.: A Tale of Two Boards: On the Influence of Microarchitecture on Side-Channel Leakage. In: Grosso, V., Pöppelmann, T. (eds.) Smart Card Research and Advanced Applications, vol. 13173, pp. 80–96. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-97348-3_5
4. Bhasin, S., Chattopadhyay, A., Heuser, A., Jap, D., Picek, S., Shrivastwa, R.R.: Mind the Portability: A Warriors Guide through Realistic Profiled Side-channel Analysis. In: Proceedings 2020 Network and Distributed System Security Symposium. Internet Society, San Diego, CA (2020). https://doi.org/10.14722/ndss.2020.24390
5. Cao, P., Zhang, C., Lu, X., Gu, D.: Cross-Device Profiled Side-Channel Attack with Unsupervised Domain Adaptation. TCHES pp. 27–56 (Aug 2021). https://doi.org/10.46586/tches.v2021.i4.27-56
6. Chari, S., Rao, J.R., Rohatgi, P.: Template Attacks. In: Goos, G., Hartmanis, J., Van Leeuwen, J., Kaliski, B.S., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2002, vol. 2523, pp. 13–28. Springer Berlin Heidelberg, Berlin, Heidelberg (2003). https://doi.org/10.1007/3-540-36400-5_3
7. Das, D., Golder, A., Danial, J., Ghosh, S., Raychowdhury, A., Sen, S.: X-DeepSCA: Cross-Device Deep Learning Side Channel Attack. In: Proceedings of the 56th Annual Design Automation Conference 2019. pp. 1–6. ACM, Las Vegas NV USA (Jun 2019). https://doi.org/10.1145/3316781.3317934
8. Genevey-Metat, C., Gérard, B., Heuser, A.: On What to Learn: Train or Adapt a Deeply Learned Profile? (2020), https://eprint.iacr.org/2020/952
9. Golder, A., Das, D., Danial, J., Ghosh, S., Sen, S., Raychowdhury, A.: Practical Approaches Toward Deep-Learning-Based Cross-Device Power Side-Channel Attack. IEEE Trans. VLSI Syst. **27**(12), 2720–2733 (Dec 2019). https://doi.org/10.1109/TVLSI.2019.2926324
10. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: A first study. J Cryptogr Eng **1**(4), 293–302 (Dec 2011). https://doi.org/10.1007/s13389-011-0023-x
11. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make Some Noise. Unleashing the Power of Convolutional Neural Networks for Profiled Side-channel Analysis. TCHES pp. 148–179 (May 2019). https://doi.org/10.46586/tches.v2019.i3.148-179
12. Kim, T., Lee, J., Nam, J.: Sample-level CNN Architectures for Music Auto-tagging Using Raw Waveforms (Feb 2018). https://doi.org/10.48550/arXiv.1710.10451
13. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99. pp. 388–397. Springer, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_25
14. Le, T.H., Canovas, C., Clédière, J.: An overview of side channel analysis attacks. In: Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security. pp. 33–43. ACM, Tokyo Japan (Mar 2008). https://doi.org/10.1145/1368310.1368319
15. Lerman, L., Bontempi, G., Markowitch, O.: Power analysis attack: An approach based on machine learning. IJACT **3**(2), 97 (2014). https://doi.org/10.1504/IJACT.2014.062722
16. Lerman, L., Poussier, R., Markowitch, O., Standaert, F.X.: Template attacks versus machine learning revisited and the curse of dimensionality in side-channel analysis: Extended version. J Cryptogr Eng **8**(4), 301–313 (Nov 2018). https://doi.org/10.1007/s13389-017-0162-9

17. Mangard, S.: Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness. In: Goos, G., Hartmanis, J., Van Leeuwen, J., Okamoto, T. (eds.) Topics in Cryptology – CT-RSA 2004, vol. 2964, pp. 222–235. Springer Berlin Heidelberg, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24660-2_18
18. Ninan, M., Nimmo, E., Reilly, S., Smith, C., Sun, W., Wang, B., Emmert, J.M.: A Second Look at the Portability of Deep Learning Side-Channel Attacks over EM Traces. In: The 27th International Symposium on Research in Attacks, Intrusions and Defenses. pp. 630–643. ACM, Padua Italy (Sep 2024). https://doi.org/10.1145/3678890.3678900
19. Papagiannopoulos, K., Glamočanin, O., Azouaoui, M., Ros, D., Regazzoni, F., Stojilović, M.: The Side-channel Metrics Cheat Sheet. ACM Comput. Surv. **55**(10), 1–38 (Oct 2023). https://doi.org/10.1145/3565571
20. Picek, S., Heuser, A., Jovic, A., Bhasin, S., Regazzoni, F.: The Curse of Class Imbalance and Conflicting Metrics with Machine Learning for Side-channel Evaluations. IACR Transactions on Cryptographic Hardware and Embedded Systems pp. 209–237 (2019). https://doi.org/10.13154/tches.v2019.i1.209-237
21. Picek, S., Perin, G., Mariot, L., Wu, L., Batina, L.: SoK: Deep Learning-based Physical Side-channel Analysis. ACM Comput. Surv. **55**(11), 1–35 (Nov 2023). https://doi.org/10.1145/3569577
22. Prouff, E., Strullu, R., Benadjila, R., Cagli, E., Dumas, C.: Study of Deep Learning Techniques for Side-Channel Analysis and Introduction to ASCAD Database (2018), https://eprint.iacr.org/2018/053
23. Randolph, M., Diehl, W.: Power Side-Channel Attack Analysis: A Review of 20 Years of Study for the Layman. Cryptography **4**(2), 15 (May 2020). https://doi.org/10.3390/cryptography4020015
24. Thapar, D., Alam, M., Mukhopadhyay, D.: TranSCA: Cross-Family Profiled Side-Channel Attacks using Transfer Learning on Deep Neural Networks (2020), https://eprint.iacr.org/2020/1258
25. Wang, H., Brisfors, M., Forsmark, S., Dubrova, E.: How Diversity Affects Deep-Learning Side-Channel Attacks. In: 2019 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC). pp. 1–7. IEEE, Helsinki, Finland (Oct 2019). https://doi.org/10.1109/NORCHIP.2019.8906945
26. Yu, H., Shan, H., Panoff, M., Jin, Y.: Cross-Device Profiled Side-Channel Attacks using Meta-Transfer Learning. In: 2021 58th ACM/IEEE Design Automation Conference (DAC). pp. 703–708. IEEE, San Francisco, CA, USA (Dec 2021). https://doi.org/10.1109/DAC18074.2021.9586100
27. Zhang, F., Shao, B., Xu, G., Yang, B., Yang, Z., Qin, Z., Ren, K.: From Homogeneous to Heterogeneous: Leveraging Deep Learning based Power Analysis across Devices. In: 2020 57th ACM/IEEE Design Automation Conference (DAC). pp. 1–6. IEEE, San Francisco, CA, USA (Jul 2020). https://doi.org/10.1109/DAC18072.2020.9218693