Main Title

Standard blurb goes here

Series page goes here (if applicable); otherwise blank

Book Title

Subtitle for the book (if Any)

First Edition

Author Name(s)
Edited by Editor Name(s)

Dedication
More than one line text
to break here

Contents

1 Deep Learning Side-Channel Attacks: Challenges and Opportunities

Authors: Logan Reichling (0009-0004-1775-1887), Mabon Ninan (0009-0000-3619-5848), Boyang Wang (0000-0001-8973-2328), and John M. Emmert (0000-0002-6074-535X), *University of Cincinnati*

Abstract: Side-channel attacks can reveal encryption keys on a target (e.g., a microcontroller or FPGA) by analyzing power consumption or electromagnetic (EM) signals. The attack is feasible as correlations exist between power consumption and intermediate results of encryption. It is one of the most critical threats to the security and trust of hardware and embedded systems. Recent research studies have shown that deep learning side-channel attacks can outperform traditional side-channel attacks, such as Template Attacks, especially when countermeasures (including masking and random delay) are applied. Despite the substantial progress in recent research, deep learning side-channel attacks still face several critical challenges, such as portability and explainability. In this chapter, we present the workflow of deep learning side-channel attacks, summarize the latest findings of deep learning side-channel attacks, highlight the challenges in this line of research, and discuss the future research opportunities of deep learning side-channel attacks.

1.1 INTRODUCTION

Side-channel attacks (SCAs) [20, 11, 8] can reveal encryption keys on a target device, e.g., a microcontroller or FPGA, despite the fact that an encryption algorithm is mathematically secure. The attacks examine correlations between power consumption and intermediate outputs of an encryption algorithm, such as Advanced Encryption Standard (AES), over a large amount of traces captured from a device. After intermediate outputs are revealed, the encryption key can be further inferred by known plaintexts/ciphertexts and the encryption algorithm. Once an encryption key is compromised, the communication and data associated with a device are no longer secure. It is one of the most fundamental threats to the security and trust of hardware and embedded systems.

Since the seminal work by Kocher et al. in 1999, traditional side-channel attacks, including Simple Power Analysis (SPA) [10], Differential Power Analysis (DPA)

[21], Correlation Power Analysis (CPA) [8], and Template Attacks [11], have been extensively investigated. To mitigate side-channel leakage and enhance the security of encryption algorithms, two types of countermeasures, *masking* and *blinding*, can be applied. Masking introduces masks, i.e., random values, in the implementation of encryption to hide correlations between power consumption and intermediate results, which mitigates side-channel leakage without affecting the correctness of an encryption algorithm. For instance, a mask can be applied with two exclusive-or operations, one before and one after a leakage function respectively, to hide side leakage but maintain the correctness of the encryption. Blinding randomly shuffles the order of operations or randomly delays operations in the implementation of encryption to desynchronize the samples of the same operations across traces.

Recently, the research community has shown that deep learning side-channel attacks can offer advantages, such as defeating countermeasures and requiring less preprocessing, over traditional side-channel attacks. Maghrebi et al. [27] published the first work evaluating deep neural networks for side-channel attacks against protected and unprotected implementations of AES in 2016. Specifically, the authors investigated multiple neural networks, including Multi-Layer Perceptrons, Convolutional Neural Networks, Autoencoders, Recurrent Neural Networks, and Long Short-Term Memory, and compare their attack performance with Template Attacks. As reported in a survey paper by Picek et al. [35], over 180 papers focusing on deep learning-based side-channel attacks had been published as of 2023.

Despite the substantial progress in deep learning side-channel attacks, there are still many challenges and opportunities. In this chapter, we introduce the typical workflow of deep learning side-channel attacks with concrete examples to help researchers, especially the ones who are new to side-channel attacks, to enhance their understanding and knowledge in this research area. Moreover, we highlight several emerging aspects in deep learning side-channel attacks, including (1) *new datasets*, (2) *new architectures*, (3) *architecture optimization*, (4) *portability*, (5) *explainability*, (6) *non-profiling attacks*, and (7) *pre-silicon side-channel analyses*, and discuss open research opportunities associated with these aspects.

The rest of this chapter is organized as follows. In Section 1.2, we introduce the system model and basic notations of side-channel attacks. In Section 1.3, we illustrate the workflow of deep learning side-channel attacks, from establishing a trace acquisition setup to obtaining attack results, with examples over power traces collected from ARM STM32F3 (a 32-bit microcontroller) running AES-128. In Section 1.4, we discuss existing works and highlight open research opportunities. Finally, we conclude this chapter in Section 1.5.

1.2 BACKGROUND ON SIDE-CHANNEL ATTACKS

Depending the assumptions on an attacker, side-channel attacks consists of two categories, including *profiling attacks* and *non-profiling attacks*. We primarily discuss profiling side-channel attacks in this chapter.

System Model. The system model of a profiling side-channel attack involves two devices: the *profiling device* and the *target device*. We assume that an attacker has

complete control over the profiling device, e.g., a microcontroller or FPGA. Specifically, the attacker controls the plaintexts and associated key on the profiling device and can capture an unlimited number of power or electromagnetic (EM) traces to train a *profile* (i.e., a classifier). This attacker can passively capture the power consumption or EM radiation and plaintexts on the test device. By leveraging the profile, the attacker aims to reveal an *unknown but fixed key* on the test device.

A profiling side-channel attack includes two phases, the *training phase* (or *profiling phase*) and the *test phase* (or *attack phase*). In the profiling phase, the attacker trains a profile with labeled traces from the training device. Labels of traces are intermediate results of encryption (or the Hamming weights of these intermediate results), which are derived based on the known key and plaintexts. In the attack phase, the attacker captures traces and plaintexts from the test device and aims to reveal the unknown key on the test device by leveraging the profile. In deep learning side-channel attacks, *a profile is a trained neural network*.

For a non-profiling attack, an attacker does not have access to a profiling device to build a profile in advance. Instead, it only has *unlabeled traces* from the test device to perform the attack. As a result, it only consists of one phase, i.e., *the attack phase*, which aims to distinguish the correct key from incorrect candidates. For instance, a deep learning non-profiling attack can train 256 neural networks over the same set of traces to identify the correct key.

Notations. A power trace t is one-dimensional time series data represented as a vector $t = (t[1], \ldots, t[l])$, where $t[i]$ is the measurement of power consumption at timestamp i and l is the total number of measured samples. We use $\mathscr{M}$ and $\mathscr{K}$ to denote the plaintext space and the key space respectively. Samples within a trace t are recorded as a device runs encryption with plaintext m and key k, where $m \in \mathscr{M}$ and $k \in \mathscr{K}$. Intermediate values of encryption are denoted as $z = \varphi(m,k)$, where the function $\varphi(\cdot)$ is a leakage step of the encryption algorithm.

AES-128. The description of side-channel attacks in this chapter focuses on AES-128 encryption, where an encryption key has 128 bits (i.e., 16 bytes). The side-channel attacks over AES are performed in a *divide-and-conquer* approach, where one key byte is revealed each time. Following problem descriptions in the literature [36, 31], we can assume key k, plaintext m, or intermediate value z has one byte. We use $k_1^*, k_2^*, ..., k_{256}^*$ to denote all the possible 256 key candidates.

Leakage Model. A leakage model will need to be selected to formulate side-channel leakage and calculate the labels of traces in side-channel attacks. Two popular leakage models, including the *Hamming Weight (HW) model* and the *Identity (ID) model*. The HW model assumes that there are correlations between the power consumption of an intermediate value and the Hamming weight of this intermediate value. The label of a trace t is $\mathtt{HW}(z)$ — the Hamming weight of the intermediate value $z = \varphi(m,k)$ given plaintext m and key k. There are 9 possible Hamming weights (or 9 possible labels) over one byte. The ID model assumes that there are correlations between the power consumption and the intermediate value itself. The label of a trace is the intermediate value z, and there are 256 possible labels.

Evaluation Metrics. Given a trained profile and a trace, the test phase derives a

score for every possible label. Next, each score is assigned to corresponding key candidate(s) based on the label, an associated plaintext, and AES encryption algorithm. Every key candidate's scores across all the test traces are aggregated. Since there are 256 key candidates given one byte, 256 aggregated scores are derived and sorted in a descending order.

Specifically, let us assume that an attacker can capture N' test traces $T' = (t'_1, ..., t'_{N'})$ and their N' plaintexts $M' = (m'_1, ..., m'_{N'})$ from a test device running unknown key k', where m'_i is associated with t'_i. Assume the ID model is used as the leakage model. Given a trace t'_i, the trained neural network outputs a *ID score vector* $(s_i[0], ..., s_i[255])$, where $s_i[j]$ is the score for label byte value j (in decimal). Next, an attacker obtains a *key score vector* $(r_i[k^*_1], ..., r_i[k^*_{256}])$ by calculating

$$r_i[k^*_g] = s_i[j], \quad \text{if} \quad \varphi(m'_i, k^*_g) == j \tag{1.1}$$

for $1 \leq g \leq 256$, where $r_i[k^*_g]$ is the score of possible key k^*_g according to trace t'_i and plaintext m'. The *aggregated key score vector* $(r[k^*_1], ..., r[k^*_{256}])$ over N' power traces are computed as

$$r[k^*_j] = \sum_{i=1}^{N'} r_i[k^*_j], \textit{ for } 1 \leq j \leq 256 \tag{1.2}$$

The aggregated key scores $(r[k^*_1], ..., r[k^*_{256}])$ are further sorted in descending order based on the scores.

We utilize *Key Rank* (*or Guessing Entropy*) and *Measurements To Disclosure* (MTD) to measure the effectiveness of side-channel attacks [32, 41]. Key rank r, where $r \in [1, 256]$, is the rank of the aggregated score of the correct key among all the 256 key candidates given a certain number of test traces. For instance, a key rank of 1 given 200 test traces indicates that the correct key has the highest score and the attacker can distinguishe the key correctly within 200 test traces. MTD indicates the minimal number of test traces that is needed for the key rank to converge to 1 (i.e., when it is distinguishable from others). A lower MTD indicates the attack is more effective.

1.3 EXAMPLES OF DEEP LEARNING SIDE-CHANNEL ATTACKS

The complete workflow of deep learning side-channel attacks includes the following steps: (1) establishing a trace acquisition setup; (2) identifying Points of Interest; (3) large-scale trace collection; (4) statistical analysis; (5) deep learning model training; and (6) reporting attack results.

In the following section, we present the workflow of deep learning side-channel attacks over power traces from the ARM STM32F3 microcontroller running unmasked AES-128 as an example. ARM STM32F3 is a 32-bit microcontroller, which is widely used on embedded devices. We utilize TinyAES, a C implementation of unmasked AES-128 that is customized for embedded systems with low memory usage. The power traces are collected using ChipWhisperer [29]. In addition, we demonstrate the advantages of deep learning side-channel attacks over traditional

side-channel attacks given countermeasures (e.g., random delays). Understanding these examples will help readers enhance their capabilities on performing and troubleshooting the evaluations of deep learning side-channel attacks over various targets and datasets in practice. Examples of source code that can be used for statistical analysis, deep learning model training, and reporting attack results can be found [24, 44].

1.3.1 TRACE ACQUISITION SETUP

The first step of side-channel attacks is to establish a trace acquisition setup, which can collect traces from a target running an encryption algorithm. There are no differences between traditional side-channel attacks and deep learning side-channel attacks in terms of a trace acquisition setup. We briefly describe a typical trace acquisition setup for power and EM traces in this section for completeness.

Power Trace Acquisition. A common power trace acquisition setup requires components including (1) a target (e.g., microcontroller or FPGA); (2) an implementation of an encryption algorithm (e.g., a C implementation of AES-128); (3) an oscilloscope (with a high sampling rate); and (4) a computer (with large memory). During data collection, the implementation of an encryption algorithm is first compiled on the computer. Next, the compiled program (e.g., a binary) is loaded to the target with the help of the computer. To collect a trace, the computer passes a plaintext and a key to the target such that the target will perform one execution of the encryption. During this process, the oscilloscope captures a trace and passes it to the computer. The computer saves this trace and its associated plaintext and key.

Ideally, the entire trace, i.e., the samples from one entire execution of the encryption algorithm, is kept. However, due to storage limitation for the later large-scale data collection, one can opt to save a small portion of the samples from the entire execution. This requires identifying the leakage step to ensure the samples from the leakage step are included. We will discuss more details regarding this aspect in the next subsection. The sampling rate typically should be at least 3 or 4 times higher than the clock rate of the target in order to capture samples associated with the executions of the leakage step. Typically, an additional trigger signal is provided to automatically record the start of each encryption execution, which can align traces across multiple executions easily.

Establishing a trace acquisition setup is one of the key challenges for side-channel attacks as it often requires comprehensive engineering experience and extensive hand-on skills. This could be a major barrier, especially for new researchers to this area. Fortunately, thanks to the efforts of the research and industry community in the past decade, there are tools available that can ease the process. For instance, ChipWhisperer is an open-source, affordable, and easy-to-use toolset that can collect traces from microcontrollers and FPGAs for side-channel attacks. It is very popular among the research and industry community. More information regarding ChipWhisperer hardware, its API, and tutorials can be found at [29].

EM Trace Acquisition. Built upon a power trace acquisition setup, a typical EM trace acquisition setup requires additional components, including (5) an EM probe, (6) an amplifier, (7) a probe holder, and (8) circuit board clamps. Specifically, an EM

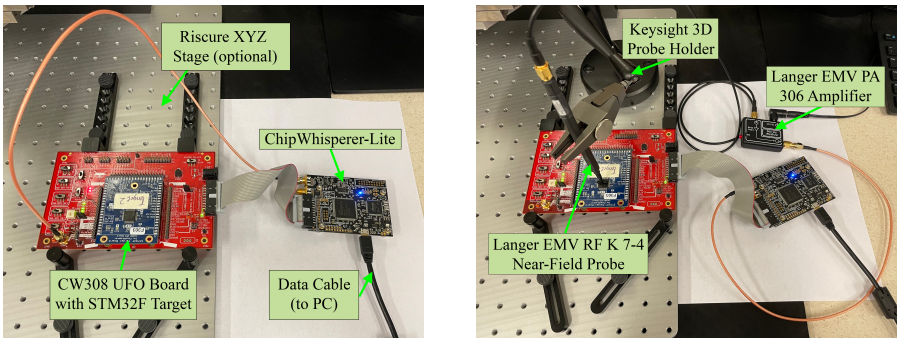

Figure 1.1: A power trace acquisition setup (left) and an EM trace acquisition setup (right) using ChipWhisperper.

probe is positioned over the top of a target to collect EM signals while the target runs the encryption algorithm. Ideally, the probe should be very close to the target (e.g., 2mm) but does not physically touch it. The EM signals are then passed to the amplifier to reduce noise. Next, the amplifier passes the filtered signals to the computer. The probe holder and circuit board clamps help fix the position of the probe and the target to minimize noise during EM trace acquisition. Once connected, the steps for collecting a trace are almost identical to the ones from power trace acquisition. Examples of a power trace acquisition and an EM trace acquisition using ChipWhisperper are illustrated in Fig. 1.1.

Even given the tiny surface of a target (e.g., 8mm $\times$ 8mm), The location of the probe can be critical for high-quality EM traces (i.e., traces with high Signal-to-Noise Ratio). In general, a probe location that is closer to the data bus and ALU (Arithmetic Logic Unit) can result in EM traces with higher Signal-to-Noise Ratio. Multiple probe locations can be scanned and compared to decide which probe location is the best option before conducting a large-scale trace collection.

1.3.2 IDENTIFYING POINTS OF INTEREST

Before collecting traces on a large scale, it is critical to identify Points of Interest, the samples associated with the leakage step (as well as the ones associated with countermeasures, e.g., masking) given a target, an encryption algorithm, and its software/hardware implementation. Identifying Points of Interest before large-scale trace collection will allow us to keep a small portion of the samples from the entire execution, which can significantly save the storage for traces and downstream analyses. This is particularly important for microcontrollers due to the large number of samples from one execution of AES. For instance, one execution of TinyAES running on ARM STM32F3 can lead to around 50,000 samples given a sampling rate of 28 MHz (about 4 times of the clock rate of the microcontroller) using ChipWhisperer. Keeping all the 50,000 samples per trace requires large storage overheads and it is not necessary. For masked AES implementation on microcontrollers, the number of samples per AES execution is even much greater and a higher sampling rate is

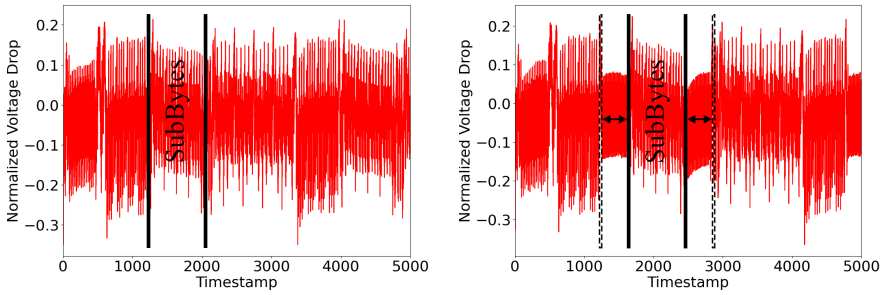

Figure 1.2: A comparison of two power traces from ARM STM32F3 running TinyAES with ChipWhisperper. The left one is a power trace collected from the original C code and the right one is a power trace collected from the modified C code (25 NOP operations added before and after SubBytes function respectively). The C code was compiled with `gcc` 9.2.1 and `O0` optimization.

also preferred, and therefore, identifying Points of Interest is certainly needed before large-scale trace collections. Even when the storage overhead is not an issue and an entire trace from a microcontroller can be kept, identifying Points of Interest in advance will allow us to provide a reasonable size of samples as an input to a neural network, rather than passing an entire trace in the later training step.

On the other hand, for FPGAs, keeping all the samples from one execution of AES does not lead to significant storage overheads and it is fine to collect a large number of traces without specifically identifying Points of Interest in advance. In other words, all the samples from one AES execution are considered as Points of Interest. For instance, there are less than 150 samples for one execution of unmasked AES when collecting a power trace from a FPGA target using ChipWhisperer given a sampling rate of 96 MHz.

For AES-128, most of the existing studies consider the first round of SubBytes as the leakage step. There are two main reasons: (1) the side-channel leakage is high as SubBytes is a non-linear function; (2) it is easier to derive the encryption key in the analyses as the round key used in the first round of AES-128 is exactly the same as the encryption key.

One effective approach for identifying Points of Interest is to add a certain number of consecutive No Operations (NOPs) instructions (e.g., 20∼30 NOPs) before and after the leakage function in the source code (C or assembly), and then compare the power trace of this modified code with the power trace of the original code. An example of the comparison of power traces is presented in Fig. 1.2. As we select the first round of SubBytes as the leakage step, the inserted NOPs cause noticeable gaps before and after the first round of SubBytes. These gaps offer indications of the start and end timestamps of SubBytes in the power trace of the original code. In this particular example shown in Fig. 1.2, Points of Interest are set as [1200, 2200], which covers all the samples associated with the first round of SubBytes.

Note that the selections of the start and end of Points of Interest are relatively loose

in this example, and can be further optimized if one prefers. In addition, we highly recommend to keep the version of the compiler and optimization level consistent between the original code and modified code as different software settings can lead to different sets of machine instructions, and therefore, unexpected shifts in terms of Points of Interest as shown in a recent study [43].

1.3.3 LARGE-SCALE TRACE COLLECTION

Once the first two steps are completed, the next step is to collect a large number of traces. Specifically, by leveraging the trace acquisition setup, the computer passes a number of N plaintexts and a key to the target such that the target will execute the encryption N times. The oscilloscope captures N traces, one for each plaintext, and passes these traces to the computer. In addition, the computer also saves the N plaintexts and the key. Finally, a dataset consisting of N traces, N plaintexts, and a key is assembled. In other words, the dataset can be presented as a set of N tuples, in which each tuple is in the form of (`trace`, `plaintext`, `key`). This concludes the large-scale trace collection.

In this example, we collect a dataset of $N = 50,000$ power traces, plaintexts, and a key by running STM32F3 using ChipWhisperer. We denote this dataset as `S1_K1_50k`. In addition, we also generate a dataset by randomly delaying samples in each trace with a number of d samples, where d is uniformly selected from [0,50]. This is a common approach to simulate traces with random delays. We denote this dataset as `S1_K1_50k_RD50`.

Note that a dataset can contain traces from multiple keys as long as a trace, a plaintext, and a key are stored accordingly. One can also keep ciphertexts instead of plaintexts (or even keep both ciphertexts and plaintexts) in a dataset. There are no specific advantages between keeping plaintexts and ciphertexts as long as it is sufficient to generate a label (i.e., an intermediate result of an encryption) given a key and a plaintext (or ciphertext).

Ideally, we should collect traces from two targets (e.g., two ARM STM32F3 microcontrollers with two different keys), where one target serves as the training device and the other serves as the target device. This is critical, especially when we would like to consider the domain shifts between training and test traces. For ease of presentation, we assume that both training and test traces are from the same device in this example.

1.3.4 STATISTICAL ANALYSES

Given a dataset, we highly recommend performing statistical analyses before training a neural network. Specifically, the results of statistical analyses can validate at which timestamps side-channel leakage happen and to what degree leakage may present. Examining the results of statistical analyses enhances the explainability of the entire process and minimizes the cases of blindly training a neural network as a black-box process without understanding the root cause of side-channel leakage. Common statistical analyses include Signal-to-Noise Ratio [32], TVLA (Test Vector Leakage

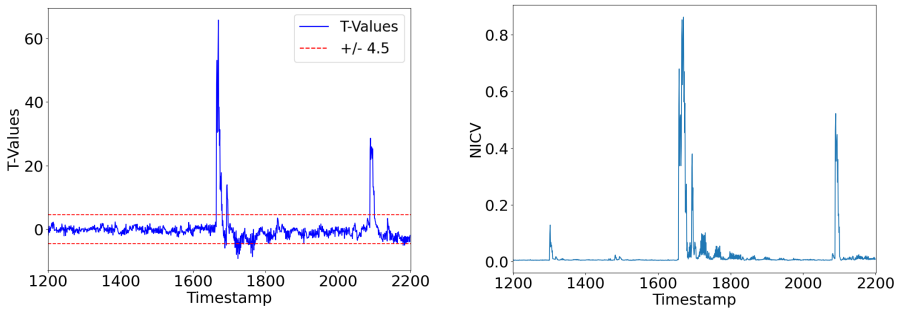

Figure 1.3: Results of statistical analysis (left: TVLA, right: NICV) over dataset `S1_K1_50k` based on the 3rd key byte, where distinguishable high values suggest high side-channel leakage.

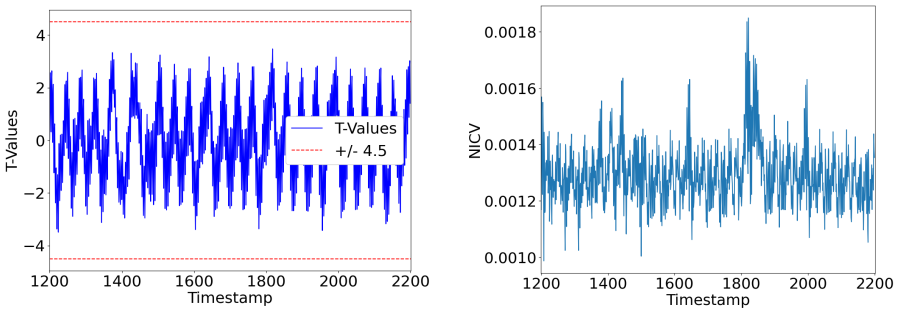

Figure 1.4: Results of statistical analysis (left: TVLA, right: NICV) over dataset `S1_K1_50k_RD50` based on the 3rd key byte, where distinguishable high values are no longer available due to random delays.

Assessment, also referred to as t-test) [1], and NICV (Normalized Inner Class Variance) [7]. The details of each method are different but share similar concepts (i.e., indicating where and to what degree there is side-channel leakage, but does not recover a key directly). Typically, performing one is sufficient. Same as side-channel attacks, statistical analyses over AES are based on each key byte.

In Fig. 1.3, we provide the results of TVLA and NICV given dataset `S1_K1_50k`, where we select the 3rd key byte and all the 50,000 traces for analysis. As we can observe from Fig. 1.3, TVLA shows that significant side-channel leakage associated with the 3rd key byte (in essence, the 3rd byte of the output of 1st round SubBytes) can be observed around timestamp 1,675. In TVLA, a value higher than 4.5 indicates that a key byte can likely be recovered. Similarly, we also observe high NICV values at the same timestamp around 1,675 given the results from the same byte. We can also further expand the analyses on all the 16 key bytes. For instance, we aggregate the results of NICV over `S1_K1_50k` across all the 16 bytes in Fig. 1.5. This also offers another opportunity to validate the selection of Points of Interest covering all the samples associated with the 1st round of SubBytes.

An another example of statistical analyses over dataset `S1_K1_50k_RD50` is pre-

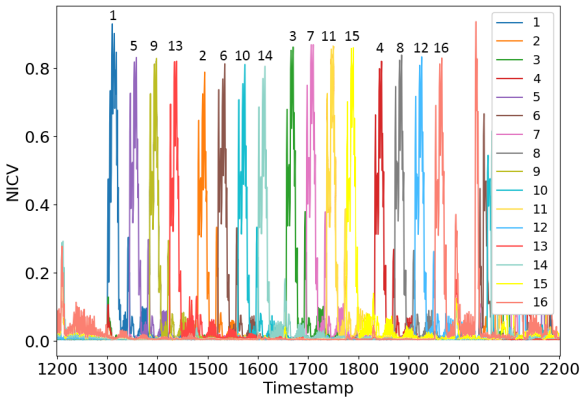

Figure 1.5: Results of NICV over dataset `S1_K1_50k` across all the 16 key bytes, where distinguishable high values suggest high side-channel leakage from every byte.

sented in Fig. 1.4. We still choose all the 50,000 power traces and the 3rd key byte for analysis. Due to random delays, samples from the 3rd byte of the output of the 1st round SubBytes do not happen at the same timestamps across all the traces, which significantly minimizes the side-channel leakage. The TVLA values are lower than 4.5 and the NICV values are almost zero. In other words, the values around 1,675, i.e., the values associated with the 3rd key byte of the output of the 1st round SubBytes, are not sufficiently distinguishable from the values of other timestamps. These results indicate that traditional side-channel attacks such as Correlation Power Analyses or Template Attacks, will fail to recover keys over this dataset.

1.3.5 TRAINING A NEURAL NETWORK

With the steps describe above, we are now ready to move on to the training of a neural network for side-channel attacks. Given dataset `S1_K1_50k`, we use 40,000 traces for training, 5,000 traces for validation, and 5,000 traces for testing. Based on our experience for power traces from microcontrollers running unmasked AES, typically it is sufficient to train a neural network with 10k∼50k training traces such that the neural network can recover keys later in the attack phase. For more challenging datasets, such as EM traces, FPGAs, or masked AES, a much greater number of training traces (e.g., greater than 100k training traces) is suggested.

When performing the training, we pass each training trace (samples from Points of Interest only as shown in Fig. 1.6) and its label to a neural network. The label of each trace is calculated as the output of 1st round SubBytes given a chosen byte, where this output can be generated by the plaintext, key, and the encryption algorithm. For instance, when we choose the 3rd key byte as the byte we would like to attack, then the label of a trace can be assigned as the 3rd byte of the output of the 1st round SubBytes computed from the 3rd key byte, the 3rd plaintext byte, and AES encryption algorithm.

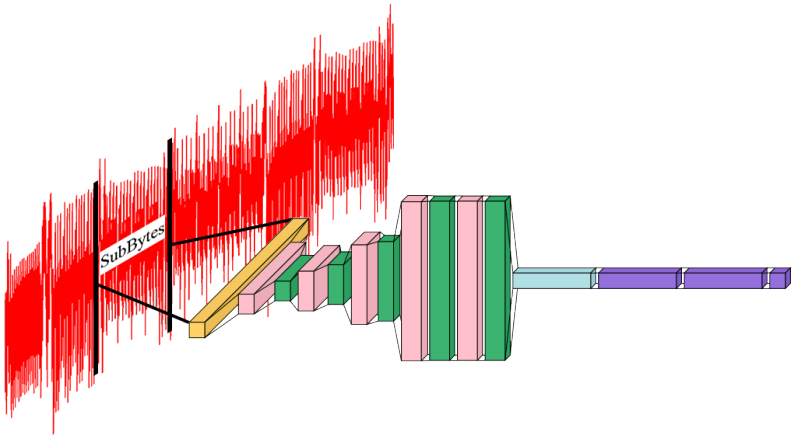

Figure 1.6: 3D Representation of power trace fed to a CNN. Only samples from Points of Interest are passed to the input layer. Layer legend: [Yellow: Input, Pink: Convolutional, Green: Average Pooling, Blue: Flatten, Purple: Dense]

Convolutional Neural Networks (CNNs) are commonly used for side-channel attacks and often derive promising results. We leverage a CNN used over the ASCAD dataset as an example. This CNN contains 5 convolutional blocks, two dense layers, and 1 output layer. Each convolutional block consists of a convolutional layer and an average pooling layer. Hyperparameters of this CNN can be found in [4].

To perform training, we need to choose several important parameters, including Points of Interest, the number of training traces, leakage model, target byte, and the number of training epochs. In this example, we choose [1200, 2200] as Points of Interest, 40,000 as the number of training traces, Identity (ID) model as the leakage model, 3rd key byte as the target byte, and 150 epochs. Through the training process, the neural network calculates the loss and update weights accordingly through back-propagation to minimize the loss over training data. In other words, the training process aims to train a neural network such that it can predict encryption intermediate results, e.g., the output of 1st round SubBytes. An example of the loss changes during the training process is presented in Fig. 1.7. The training in this example can be completed with about 40∼60 minutes given a machine equipped with NVIDIA GeForce RTX 4080 GPU.

Compared to machine learning in other domains, it is critical to keep in mind that *a well-trained neural network for side-channel attacks typically does not derive extremely high accuracy as the ones in other domains* (e.g., over 95% accuracy in image recognition). For instance, the training accuracy of a CNN trained from dataset `S1_K1_50k` only achieves an accuracy around 4% given the ID model. On the other hand, this accuracy is already way above the random guess of the ID model (1/256=0.39%) and is sufficient to recover keys successfully. This difference, in essence, is because successfully recovering keys relying on the aggregated scores across a certain number of traces rather than examining scores from each trace in-

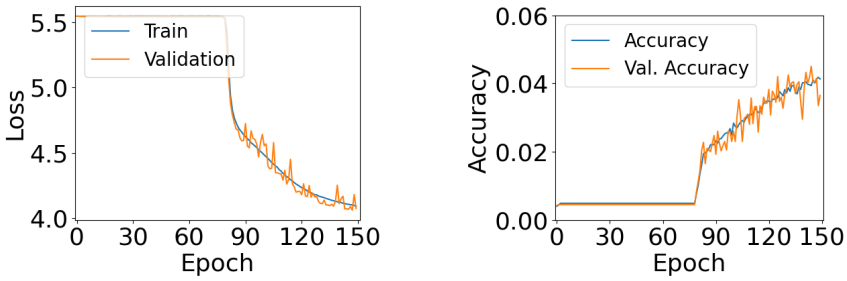

Figure 1.7: Loss changes (left) and accuracy (right) during the training process of the CNN given training traces from dataset `S1_K1_50k` (ID model, 3rd key byte, 150 epochs).

dependently. It is worth mentioning that if a different leakage model is utilized, then the expectation of the accuracy after training should be adjusted accordingly. For instance, if the HW model is used as the leakage model, then the accuracy of a neural network after the training should be sufficiently higher than the random guess of the HW model (1/9=11.11%) in order to (potentially) recover keys correctly later in the test phase. Our experience over various datasets show that, typically, choosing the ID model would derive better results when training a profile with a neural network, especially given challenging datasets, such as EM traces and traces from FPGAs.

1.3.6 REPORTING ATTACK RESULTS

Once the training is completed, we can perform the attack/test phase. Given a trained neural network, we pass each test trace to the neural network, obtain a score for each label among all the 256 possible labels, and assign each score to a candidate key byte based on each label and the plaintext of this trace. This step is repeated across all the test traces. The scores on each candidate key byte from all the test traces are aggregated, and the aggregated scores of all the 256 candidate key bytes are compared and ranked. It is also critical to keep in mind that two scores of a label (e.g., 0x01) obtained from two test traces do not necessarily assign to the same key candidate as the plaintexts of these two traces can be different. *Assigning and aggregating scores of labels to key candidates correctly is a unique but significant step for side-channel attacks, which other machine learning applications typically do not perform.*

As mentioned, once obtaining the aggregated scores on all the candidate key bytes, the results of side-channel attacks are reported with key rank and Measurements To Disclosure (MTD) in the test phase. Specifically, given a number of N' test traces and a trained neural network, typically, the testing should perform multiple rounds (e.g., 20 rounds) where the number of N' test traces are passed to the trained neural network in each round, but the order of the test traces per each round is randomly shuffled. This is because the order of test traces can affect the results

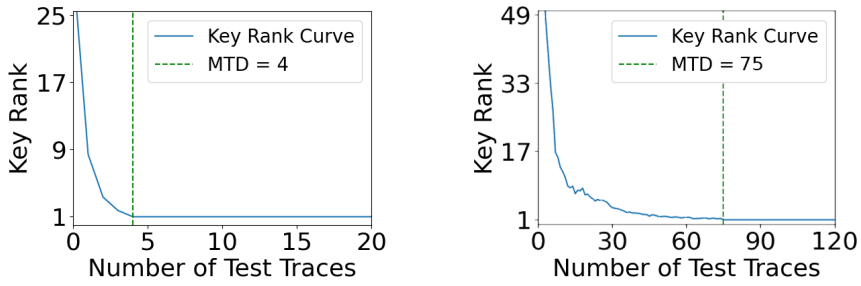

Figure 1.8: Average key rank and MTD results from side-channel attacks on 3rd key byte (left, MTD=4, deep learning; right, MTD=67, Template attacks) given 5,000 test traces from dataset `S1_K1_50k` with 20 rounds of testing.

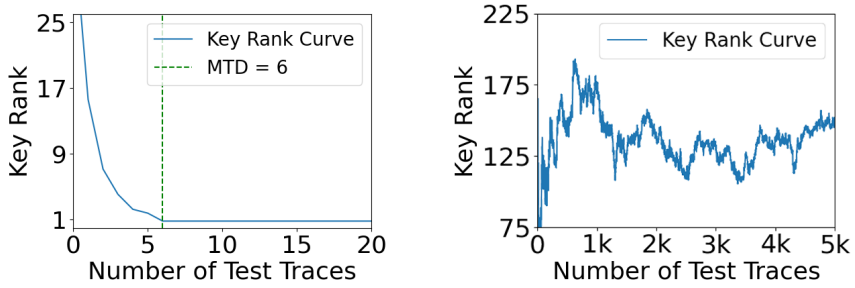

Figure 1.9: Average key rank and MTD results from side-channel attacks on 3rd key byte (left, MTD=6, deep learning; right, attack is unsuccessful, Template attacks) given 5,000 test traces from dataset `S1_K1_50k_RD50` with 20 rounds of testing.

of key rank and MTD (e.g., one specific trace carries extremely high leakage and passing this trace to the trained neural network before others can lead to much lower MTD). This could be extremely important for the evaluation of EM traces, where the variance of samples is higher across traces. Calculating the average results of MTDs from multiple rounds is more accurate to reflect the side-channel leakage. *Running multiple rounds of testing is also a unique step for side-channel attacks compared to machine learning in other domains.*

An example of key rank and MTD from the 3rd key byte is presented in Fig. 1.8, where the neural network is trained and tested based on dataset `S1_K1_50k`. We can observe that the key rank decreases when the number of test traces increases and it converges to 1 when the number of test traces is 4. This suggests that an attacker can reveal the 3rd key byte correctly with 4 test traces given the well-trained CNN. To obtain results for other bytes, we can simply retrain a CNN with each specific key byte and repeat the test phase accordingly. For comparison purpose, we also provide the attack results over dataset `S1_K1_50k` by using a Template attack, where we use the same training traces and test traces. It is easy to see that Template Attacks can

also recover keys over this dataset, but with a greater MTD. In other words, both methods can recover keys, but deep learning side-channel attacks offer better attack results in this case.

In addition to the attack results from dataset `S1_K1_50k`, we also demonstrate the attack results from dataset `S1_K1_50k_RD50` in Fig. 1.9. We still use the same number of training traces, test traces, and CNN architecture. We observe that deep learning side-channel attacks can still effectively recover keys. On the other hand, Template attacks fail to recover keys due to random delays across all the traces. This is a classic example where deep learning side-channel attacks can offer obvious advantages over traditional attacks.

1.4 STATE OF THE ART, CHALLENGES, AND OPPORTUNITIES

Deep learning side-channel attacks have been extensively studied in the past ten years. Comprehensive surveys can be found in [36, 31]. Rather than illustrating the details of every existing study, we primarily discuss recent work related to several emerging aspects, including (1) new datasets, (2) new architectures, (3) architecture optimizations, (4) portability, (5) explainability, (6) non-profiling deep learning side-channel attacks, and (7) pre-silicon side-channel analyses. We believe that these aspects bring unique challenges and new opportunities for the research community.

1.4.1 NEW DATASETS

New datasets, especially datasets acquired from implementations with countermeasures, are critical for expanding research findings in deep learning side-channel attacks. The ASCAD dataset released in [4] has been prominently used for benchmarking and comparison in the majority of recent studies. This dataset consists of two versions, referred to as ASCADv1 and ASCADv2 respectively. ASCADv1 consists of 100,000 power traces collected from an ATMEGA8515 (a 8-bit microcontroller) running a first-order boolean masked AES-128 implementation. The microcontroller was running at 4MHz. The traces were collected with a sampling rate of 2 GHz and each trace contains 100,000 samples, where Points of Interest (POIs), e.g., samples from the first round of SubBytes, are also suggested. ASCADv1 offers traces from fixed keys and variable keys. Traces with random delays (simulated) are also provided. The masked assembly implementation was also made publicly available for the research community [3]. The authors perform comprehensive analyses over this dataset using multiple CNNs and MLPs, and demonstrate the advantages of deep learning side-channel attacks over this dataset. The CNNs and MLPs are also utilized in many subsequent studies in side-channel attacks over other datasets.

Compared to ASCADv1, ASCADv2 is relatively new. It contains 800,000 power traces collected from an ARM STM32F3 microcontroller running a third-order masked AES-128 implementation using ChipWhisperer. This masked implementation offers two countermeasures, including affine masking and shuffling. The microcontroller was under-clocked to 4MHz. The traces were collected with a sampling

rate of 50 MHz and each trace contains 1 million samples. Like ASCADv1, suggested POIs are provided (e.g. mask calculations and first-round SubBytes). 800,000 random keys and plaintexts were utilized. The masked implementation was also made publicly available.

Other datasets, including AES RD [12], DPAcontestV4 [5], and CHES2020 CTF [2] have also been examined under deep learning side-channel attacks, but are less popular than the ASCAD dataset. Two recent studies release unique datasets, referred to as SoftPower [43] and CrossEM [30] respectively, for the evaluation of portability of deep learning side-channel attacks. Both datasets were acquired using ChipWhisperer. SoftPower contains 3.2 million power traces from AVG XMEGA and ARM STM32F3 microcontrollers running AES, where the binaries are not always the same due to discrepancies in software settings, such as compiler optimization, instruction rewriting, and code obfuscation. CrossEM consists of 3 million EM traces from AVG XMEGA and ARM STM32F3 microcontrollers running AES, where EM traces were collected from multiple EM probe locations over the surface of a microcontroller. Unfortunately, the traces from these two datasets are from unmasked AES only.

Opportunities. Continuing to release large-scale datasets of power traces, EM traces, and even simulated traces at the Register Transfer Level (RTL) level, from software and hardware implementation of encryption algorithms with countermeasures, will offer enormous opportunities for the research community to further understand the capabilities of deep learning side-channel attacks. In addition, detailed documents and public repositories associated with new datasets will help improving reproducibility of research and facilitating comparisons between existing and new methods.

1.4.2 NEW ARCHITECTURES

While the majority of existing studies focus on relatively common neural network architectures, such as CNNs (Convolutional Neural Networks), MLPs (Multi-Layer Perceptrons), and LSTM (Long-Short Term Memory), several recent studies introduce more comprehensive architectures, including Residual Networks (ResNets), Transformer Networks, Generative Adversarial Networks (GANs), and Large Language Models (LLMs), which demonstrate the strength of modern techniques in deep learning side-channel attacks.

Masure et al. [28] introduce a new ResNet architecture, referred to as ResNetSCA, which is customized for side-channel attacks. It contains 27 layers and incorporates Multi-Task Learning to perform side-channel attacks successfully over the ASCADv2 dataset. Multi-task learning is a deep learning technique which trains a neural network to perform multiple associated tasks simultaneously to benefit the learning of all the tasks. Specifically, the authors show that the ResNetSCA with Multi-Task Learning can predict all the 16 key bytes, 16 shuffling indices, and 2 masking bytes at the same time with an average of 60 test traces, where the POI of each trace includes 15,000 samples. On the other hand, the authors demonstrate that CNNs and MLPs fail to recover keys over this dataset. A work in parallel by Wu et al. [47] also utilizes Multi-Task Learning to predict the full AES-128 key simultaneously over

the ASCADv1 dataset. The paper also demonstrates that better performance can be achieved by predicting each bit individually instead of using one-hot encoding over a byte.

Hajra et al. [18] design EstraNet, a Transformer-style architecture, for side-channel attacks. It implements a customized modification of the self-attention architecture, which notably includes a linear-time complexity attention head kernel, allowing for a much greater number of utilized POI. The study reports successful attacks with a POI length of 60,000, which is four times greater compared to the one used in ResNetSCA. Furthermore, its custom attention layer includes relative positional encoding parameters that are learned during training. This allows EstraNet to associate specific points in time during the encryption operations with others, which is significant for tackling masking (e.g., calculations of mask shares with intermediate values after SubBytes). The relative positional encoding scheme also allows the architecture to automatically overcome random delays. The study presents low MTDs across several public datasets, including ASCADv1 and CHES2020, and outperform CNNs and LSTMs over traces with the POI length ranging from 10,000 to 60,000.

Kulkarni et al. [22] demonstrate the potential of formulating side-channel attacks as Natural Language Processing problems and address it with Large Language Models. Specifically, the authors form an *n-cryptogram* consisting of a trace, a key byte, a SubBytes output, a plaintext, a ciphertext, and a mask value. They train a LLM to recognize and later predict the difference between *ordered* and *chaotic* n-cryptograms. An ordered n-cryptogram has the correct key byte guess for the associated trace, plaintext, ciphertext, and mask value. On the other hand, a chaotic n-cryptogram has the incorrect key byte guess. By leveraging a `sigmoid` classification, the proposed Order v.s. Chaos classifier predicts the probability that a n-cryptogram is ordered. Their evaluation shows that the proposed neural network can recover key bytes correctly within 1 test trace over the ASCADv1 dataset (fixed key) and 43 test traces over the ASCADv2 dataset respectively, which are better results than previous architectures. Wang et al. [46] leverage conditional Generative Adversarial Networks (cGANs) to produce synthetic traces for training when there is a small number of traces available.

Opportunities. One emerging question to answer related to this aspect is, given a new dataset (or target), *when do we know a more comprehensive architecture is needed* rather than using a baseline architecture (e.g., a CNN). In addition, rather than experimentally enumerating different comprehensive architectures, *is there a way for researchers to measure (or at least estimate) in advance which architecture will likely succeed in recovering keys given a new dataset*.

1.4.3 ARCHITECTURE OPTIMIZATION

Architecture optimization aims to find best hyperparameters and/or reduce the number of parameters in a neural network for side-channel attacks. Rijsdijk et al. [37] leverage reinforcement learning to perform neural network optimization using a reward-based system. Their evaluations show that it is feasible to obtain a com-

pressed CNN with only 1,282 trainable parameters but still reveal a key byte correctly with 242 traces given the ASCADv1 dataset (fixed key). This is significantly compressed since the baseline CNN examined over the ASCADv1 dataset consists of over 53 million parameters. The main downside of this method is the long search time due to the large architecture search space. Specifically, given one dataset, the study reports that it takes over 100 hours to search over 2,500 combinations of hyperparameters to derive the smallest neural network. An adjacent work [49] on hyperparameter selection for side-channel attacks utilizes multi-fidelity bayesian optimization and hyperband (BOHB) search. It bounds the maximum search space with a budget parameter, where a greater budget size (i.e., a greater search space) leads to more performant neural networks given a dataset. Their experimental results indicate that the proposed approach is effective over multiple public datasets, including ASCADv1, AES HD, and CHES2018. It also requires a long search time (e.g., 2 days and 6 hours).

In addition to hyperparameter search, a couple of recent studies investigate neural network compression using pruning. Pruning reduces the size of a neural network by removing less important parameters, e.g., filters and weights. Perin et al. [33] are the first to explore pruning for side-channel attacks, where they utilize unstructured pruning (i.e., less important weights are set as zeros but not removed). Specifically, the authors make use of the Lottery Ticket Hypothesis, which states that random initialization of weights often contain ideal sub-networks for classification. The experimental results show that it is feasible to reduce over 90% of parameters of a neural network trained on the ASCADv1 dataset. A more recent work in [25] investigate *structured pruning* to reduce the size of neural networks for side-channel attacks. The authors develop an automatic pruning algorithm, named MiniDrop, which can automatically decide the best pruning rate and remove less important filters at each layer. The experimental results show that the proposed pruning is highly effective over multiple datasets, including the ASCADv1 dataset, EM traces from STM32F3, and power traces from Artix-7 FPGAs, and can achieve up to a 98% reduction rate. In addition, the authors demonstrate their pruned neural networks can be operated on a Raspberry Pi.

Opportunities. Being able to optimize neural networks for side-channel attacks and having the capabilities to deploy the networks on embedded devices may facilitate the vision of continuous, local, and real-time analyses of side-channel leakage and ensure embedded devices are monitored and secure through their lifetime.

1.4.4 PORTABILITY

In deep learning side-channel attacks, a trained neural network is considered *portable* when this network can still effectively recover keys given discrepancies (or domain shifts) between training traces and test traces. Many existing studies [15, 14, 17, 45, 13, 53, 6, 38, 51, 16, 52, 43] have investigated the portability of side-channel attacks when there are domain shifts between training and test data. Many factors, such as distinct keys, differences in software settings, hardware manufacturing variances, inconsistent acquisition setups, and EM probe positions can lead to domain shifts

between training and test data in the context of side-channel attacks. Three common existing approaches to address domain shifts for side-channel attacks are (1) multi-domain training, (2) pre-processing, and (3) domain adaptation.

Bhasin et al. [6] investigate the portability of deep learning side-channel attacks given discrepancies caused by keys and devices. The authors propose to train a neural network with traces from multiple devices (in essence, multiple domain training) to tackle domain shifts. Danial et al. [13] evaluated the portability of MLPs over EM traces from distinct devices and different EM probe locations. Their findings suggest that significant cross-device variations and inherent low SNR of EM traces bring challenges to portability. The study also shows that pre-processing methods, including Discrete Fourier Transforms (DFT), Principle Component Analysis (PCA), and Linear Discriminant Analysis (LDA), can overcome domain shifts.

Yu et al. [51] propose to utilize meta-transfer learning to address domain shifts caused by chip model variations. Similarly, Genevey-Metat et al. [16] leverage fine tuning to address domain shifts over EM traces and power traces from STM32 microcontrollers. However, both of the two methods require labeled traces from the test device, which is impractical for real-world attackers. *Knowing labels of test traces before attacks indicates that an attacker knows the key byte already.*

A recent study [43] examines domain shifts caused by software settings, which have not been widely investigated in the literature. Specifically, the authors show that multiple software factors, including compiler optimization levels, instruction rewriting, and code obfuscation, can lead to distinct binaries given the same AES implementation. Different binaries, in essence, different sets of machine instructions, lead to domain shifts between training traces and test traces. While the study demonstrates multi-domain training, training a neural network with traces from multiple software settings, can improve portability. The authors also mention that multi-domain training is not scalable for a real-world attacker in this scenario as the number of potential software settings is extremely large and difficult to enumerate in practice.

Opportunities. Despite the extensive discussions on portability, the domain shifts are complicated and can be easily triggered by multiple factors in side-channel attacks. A recent study [30] indicates that none of the existing approaches can consistently outperform others given domain shifts caused by EM probe locations. It is critical for the research community to further explore new methods that can better address domain shifts in the context of side-channel attacks, especially when traces are collected from implementations with countermeasures.

1.4.5 EXPLAINABILITY

Since neural networks are often considered as black-box approaches, the explainability of deep learning side-channel attacks is fundamental to understanding the root cause of the leakage. Hettwer et al. [19] explores multiple attribution-based methods for enhancing the explainability of a CNN in the context of side-channel attacks. Specifically, each sample of a trace is sequentially set to zero to measure its effect on a given output to determine which sample points matter the most. For instance, the study reports the results of attribution analyses over the ASCADv1 dataset by using

two methods, Saliency and Layer-wise Relevance Propagation. The results indicate that samples associated with the processing of mask values and the masked SubByte operations are most critical to the output of a CNN.

Perin et al. [34] develop a new method, referred to as ExDL-SCA, based on information flow through multiple layers of a neural network, such as a MLPs or CNNs, in side-channel attacks. Specifically, internal hidden layer representations of a neural network are fed through a small, specialized MLP at each epoch to estimate the Perceived Information index. In other words, this specialized MLP estimates to what degree the labels (and masks) can be predicted using internal data. The inner-neural outputs are training data. The original labels are outputs of this specialized MLP. This method relies on Information Bottleneck Theory — deeper layers carry more compressed representations of correct output labels. Evaluations over the ASCADv1 and DPAcontest4.2 dataset show that intermediate layers learn to separate less important features from features related to SubBytes output and masking values. In addition, the study concludes that random delay is typically tackled in earlier layers while masking is addressed in deeper layers.

Opportunities. Despite the findings on the explainability of MLPs and CNNs, there are opportunities for the research community to investigate and enhance the explainability of recent comprehensive architectures, such as ResNets [28] and Transformers [18], in the context of side-channel attacks.

1.4.6 NON-PROFILING DEEP-LEARNING SIDE-CHANNEL ATTACKS.

A recent work in [42] shows that it is feasible to perform non-profiling attacks with deep neural networks. To recover one key byte of AES, the proposed method assigns traces with guessed labels based on each possible key byte among the 256 candidates and then trains 256 neural networks, where each neural network is trained based on traces and guessed labels calculated from each possible key byte candidate. In essence, it produces 256 partitions of traces based on all the 256 possible key byte values. The distinguisher, either training accuracy or loss, is utilized to select the correct key byte candidate from incorrect ones. In other words, training of 256 neural network is the attack phase of this attack. The intuition is that the correct key byte results in correct labels, and therefore, lead to better training results while incorrect ones only lead to an accuracy similar to random guess. It is worth mentioning that the Identity model cannot be used as the leakage model in this approach as the 256 partitions would be the same, and there would be no statistical difference across these partitions, i.e., the distinguisher would fail to distinguish the correct key from others regardless how the distinguisher is selected.

Following the study in [42], several works [23, 50, 48, 39] further examine non-profiling deep learning side-channel attacks. Specifically, Kwon et al. [23] successfully attacks AES implementations with random delay and masking with a single unsupervised autoencoder, but utilizes more training traces. Wu et al. [50] successfully recovered a key byte by training a model to predict the differences in a trace given plaintext collisions. Wu et al. [48] also propose another method of determining intermediary encryption information given the plaintext, ciphertext, and trace in a

non-profiling scenario. Savu et al. [39] leverage multi-output regression models for side-channel attacks. It offers modest improvements on attack results, but is more computationally efficient than previous methods.

Opportunities. One major challenge for deep learning non-profiling side-channel attacks is the extremely long time needed to train 256 neural networks per key byte. Fully recovering all the 16 key bytes of AES-128 requires training 256×16 neural networks, which is even more time consuming. New methods are needed to optimize the overall training time for non-profiling side-channel attacks. In addition, how to perform non-profiling side-channel attacks with the recent complex neural network architectures, such as such as ResNets [28] and Transformers [18], remains unknown.

1.4.7 PRE-SILICON SIDE-CHANNEL ANALYSES.

There is an emerging trend to apply side-channel analyses at the pre-silicon phase, e.g., the design stage at the Register Transfer Level (RTL). Pre-silicon side-channel analyses can help identifying the root cause of side leakage early during the design stage and updating the design to mitigate leakage. This can save significant costs in chip design. Typically, simulated power or EM traces of a hardware design of an encryption (e.g., AES) can be generated at the RTL level or netlist level, then these simulated traces can be leveraged to perform side-channel analyses.

The survey paper by Buhan et al. in [9] identifies a range of current offerings for pre-silicon side-channel analyses, including targeted devices/implementations, tools, and capabilities. One major challenge is that a very limited number of existing tools for pre-silicon side-channel analyses are hardware independent or open-source. This is mainly because the hardware design toolchains are complicated and vary across different vendors and designers. For instance, an open-source tool, named TOFU, can produce simulated power traces for pre-silicon side-channel analyses, where these traces are produced based on Value-Change Dump (VCD) files generated by RTL designs. Electromagnetic simulation for pre-silicon side-channel leakage detection proves to be a challenging task. Sehatbakhsh et al. developed an EM emulator named EMSIM [40], which utilizes the hardware design to generate an EM signal and an accompanying physical RISC-V processor as a reference point for signal comparison. The emulator then replicates the analog EM signal by assigning scaling factors to the different categories of potential instructions (ALU, memory storage, etc.) based on recorded amplitudes from the companion RISC-V core. The study further reinforces their simulation by showing similar TVLA results from a real EM trace compared to the synthetic one. A later simulator created by Lin et al. in [26] aims to provide locality-based simulation of EM leakage in addition to analog trace synthesis. A recent study [41] investigates transfer learning for enhancing post-silicon side-channel attacks with simulated traces from the pre-silicon stage. Specifically, a neural network is initially trained with a large number of simulated traces and then fine-tuned with a small number of traces from the post-silicon stage for side-channel attacks.

Opportunities. It remains largely unknown what advantages as well as disadvantages that deep learning can offer in the context of pre-silicon side-channel analyses

compared to traditional side-channel analyses, such as TVLA and CPA. It would be interesting for the research community to investigate this aspect to enhance the implementation of countermeasures for encryption algorithms, such as AES and post-quantum encryption algorithms, at the design stage.

1.5 CONCLUSION

In this chapter, we illustrate the typical workflow of deep learning side-channel attacks and highlight the unique differences of deep learning between side-channel attacks and other applications. In addition, we extensively discuss several emerging aspects of deep learning side-channel attacks and underline multiple open research opportunities. We hope that the content of this chapter provides a solid anchor point for researchers, especially newer ones, to further investigate open problems in this area and contribute new datasets, methods, tools, and insights to the research community.

REFERENCES

1. G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi, and S. Saab. Test vector leakage assessment (tvla) methodology in practice. In *2013 International Cryptographic Module Conference*, 2013.
2. D. Bellizia, O. Bronchain, G. Cassiers, C. Momin, F. Standaert, and B. Udvarhelyi. Spook sca ctf. `https://doi.org/10.14428/DVN/W2SV5G`, 2021. DOI: 10.14428/DVN/W2SV5G.
3. R. Benadjila, V. Lomne, E. Prouff, and T. Roche. Secure aes128 for atmega8515. GitHub repository, 2018.
4. R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *Journal of Cryptographic Engineering*, 10(2), 2020.
5. S. Bhasin, N. Bruneau, J. Danger, S. Guilley, and Z. Najm. Analysis and improvements of the dpa contest v4 implementation. In S. Chakraborty, V. Matyas, and P. Schaumont, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 201–218, Cham, 2014. Springer International Publishing.
6. S. Bhasin, A. Chattopadhyay, A. Heuser, D. Jap, S. Picek, and R. R. Shrivastwa. Mind the Portability: A Warriors Guide through Realistic Profiled Side-channel Analysis. In *Proc. of NDSS'20*, 2020.
7. S. Bhasin, J. Danger, S. Guilley, and Z. Najm. NICV: Normalized inter-class variance for detection of side-channel leakage. In *2014 International Symposium on Electromagnetic Compatibility*, 2014.
8. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *Proc. of CHES'04*, 2004.
9. I. Buhan, L. Batina, Y. Yarom, and P. Schaumont. Sok: Design tools for side-channel-aware implementations, 2021.
10. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 398–412, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

11. S. Chari, J. R. Rao, and P. Rohatgi. Template Attacks. In *Proc. of Cryptographic Hardware and Embeeded Systems (CHES 2002)*, 2002.
12. "J. Coron and I. Kizhvatov". Analysis and improvement of the random delay countermeasure of ches 2009. In S. Mangard and F. Standaert, editors, *"Cryptographic Hardware and Embedded Systems, CHES 2010"*, pages 95–109, Berlin, Heidelberg, "2010". Springer Berlin Heidelberg.
13. J. Danial, D. Das, A. Golder, S. Ghosh, A. Raychowdhury, and S. Sen. EM-X-DL: Efficient Cross-device Deep Learning Side-channel Attack with Noisy EM Signatures. *ACM Journal on Emerging Technologies in Computing Systems*, 18(1):1–17, 2022.
14. D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen. X-DeepSCA: Cross-Device Deep Learning Side Channel Attack. In *Proc. of 56th ACM/IEEE Design Automation Conference (DAC'19)*, 2019.
15. M. Elaabid and S. Guilley. Portability of templates. *Journal of Cryptographic Engineering*, 2:63–74, 2012.
16. C. Genevey-Metat, A. Heuser, and B. Gerard. Train or Adapt a Deeply Learned Profile? In *Proc. of International Conference on Cryptology and Information Security in Latin America (Latin Crypt'21)*, 2021.
17. A. Golder, D. Das, J. Danial, S. Ghosh, S. Sen, and A. Raychowdhury. Practical Approaches Towards Deep-Learning Based Cross-Device Power Side Channel Attack. *IEEE. Trans. on Very Large-Scale Integration (VLSI) Systems*, 27(12), 2019.
18. S. Hajra, S. Chowdhury, and D. Mukhopadhyay. EstraNet: An Efficient Shift-Invariant Transformer Network for Side-Channel Analysis. *TCHES*, 2024.
19. B. Hettwer, S. Gehrer, and T. Guneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. Cryptology ePrint Archive, Paper 2019/143, 2019. `https://eprint.iacr.org/2019/143`.
20. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proc. of CRYPTO'99*, 1999.
21. P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1):5–27, Apr 2011.
22. P. Kulkarni, V. Verneuil, S. Picek, and L. Batina. Order vs. Chaos: A Language Model Approach for Side-channel Attacks. `https://eprint.iacr.org/2023/1615.pdf`.
23. D. Kwon, H. Kim, and S. Hong. Non-Profiled Deep Learning-based Side-Channel Preprocessing with Autoencoders. *IEEE Access*, 2021.
24. H. Li, M. Ninan, B. Wang, and J. Emmert. Tinypower github repository, 2023. `https://github.com/UCdasec/TinyPower`.
25. H. Li, M. Ninan, B. Wang, and J. M. Emmert. TinyPower: Side-Channel Attacks with Tiny Neural Networks. In *Proc. of IEEE HOST'24*, 2024.
26. L. Lin, D. Zhu, J. Wen, H. Chen, Y. Lu, N. Cheng, C. Chow, H. Shrivastav, C. W. Chen, K. Monta, and M. Nagata. Multiphysics Simulation of EM Side-Channels from Silicon Backside with ML-based Auto-POI Identification. In *IEEE HOST'21*, 2021.
27. H. Maghrebi, T. Portigliatti, and E. Proff. Breaking cryptographic implementations using deep learning techniques. In *Proc. of International Conference on Security, Privacy and Applied Cryptography Engineering (SPACE'16)*, 2016.
28. L. Masure and R. Strullu. Side channel analysis against the ANSSI's protected AES implementation on ARM. Cryptology ePrint Archive, Paper 2021/592, 2021. `https://eprint.iacr.org/2021/592`.
29. NewAE. Chipwhisperer. `https://www.newae.com/chipwhisperer`.
30. M. Ninan, E. Nimmo, S. Reilly, C. Smith, W. Sun, B. Wang, and J. Emmert. A second

look at the portability of deep learning side-channel attacks over em traces. In *2024 International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2024.
31. M. Panoff, H. Yu, H. Shan, and Y. Jin. A Review and Comparison of AI-enhanced Side Channel Analysis. *J. Emerg. Technol. Comput. Syst.*, 2022.
32. K. Papagiannopoulos, O. Glamocanin, M. Azouaoui, D. Ros, F. Regazzoni, and M. Stojilovic. The Side-channel Metrics Cheat Sheet. *ACM Computing Surveys*, 55(10), 2023.
33. G. Perin, L. Wu, and S. Picek. Gambling for success: The lottery ticket hypothesis in deep learning-based SCA. Cryptology ePrint Archive, Paper 2021/197, 2021. `https://eprint.iacr.org/2021/197`.
34. G. Perin, L. Wu, and S. Picek. I know what your layers did: Layer-wise explainability of deep learning side-channel analysis. Cryptology ePrint Archive, Paper 2022/1087, 2022. `https://eprint.iacr.org/2022/1087`.
35. S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11), feb 2023.
36. S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina. SoK: Deep Learning-based Physical Side-channel Analysis. *ACM Computing Surveys*, 55(11), 2023.
37. J. Rijsdijk, L. Wu, G. Perin, and S. Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021.
38. U. Rioja, L. Batina, and I. Armendariz. When Similarities Among Devices are Taken for Granted: Another Look at Portability. In *Proc. of AFRICACRYPT 2020*, pages 337 – 357, 2020.
39. I. Savu, M. Krcek, G. Perin, L. Wu, and S. Picek. The Need for MORE: Unsupervised Side-channel Analysis with Single Network Training and Multi-output Regression. `https://eprint.iacr.org/2023/1681.pdf`.
40. N. Sehatbakhsh, B. Yilmaz, A. Zajic, and M. Prvulovic. Emsim: A microarchitecture-level simulation tool for modeling electromagnetic side-channel signals. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 71–85, 2020.
41. D. Shanmugam and P. Schaumont. Improving Side-Channel Leakage Assessment Using Pre-Silicon Leakage. In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*, 2023.
42. B. Timon. Non-Profiled Deep Learning-based Side-Channel Attacks with Sensitivity Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):107–131, 2019.
43. C. Wang, M. Ninan, S. Reilly, J. Ward, W. Hawkins, B. Wang, and J. M. Emmert. Portability of Deep-Learning Side-Channel Attacks against Software Discrepancies. In *Proc. ACM WiSec'23*, 2023.
44. C. Wang, M. Ninan, S. Reilly, J. Ward, W. Hawkins, B. Wang, and J. M. Emmert. Softpower github repository, 2023. `https://github.com/UCdasec/SoftPower`.
45. H. Wang, M. Brisfors, S. Forsmark, and E. Dubrova. 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)*, 2019.
46. P. Wang, P. Chen, Z. Luo, G. Dong, M. Zheng, N. Yu, and H. Hu. Enhancing the Performance of Practical Profiling Side-Chanel Attacks Using Conditional Generative Adversarial Networks. https://arxiv.org/abs/2007.05285.
47. L. Wu, A. Ali-Pour, A. Rezaeezade, G. Perin, and S. Picek. Breaking Free: Leakage Model-free Deep Learning-based Side-channel Analysis.

https://eprint.iacr.org/2023/1110.pdf.
48. L. Wu, G. Perin, and S. Picek. Hiding in Plain Sight: Non-profiling Deep Learning-based Side-channel Analysis with Plantext/Ciphertext. https://eprint.iacr.org/2023/209.pdf.
49. L. Wu, G. Perin, and S. Picek. I Choose You: Automated Hyperparameter Tuning for Deep Learning-based Side-Channel Analysis. *IEEE Transactions on Emerging Topics in Computing*, 2022.
50. L. Wu, S. Tiran, G. Perin, and S. Picek. An End-to-end Plaintext-based Side-channel Collision Attack without Trace Segmentation. `https://eprint.iacr.org/2023/1109.pdf`.
51. H. Yu, H. Shan, M. Panoff, and Y. Jin. Cross-Device Profiled Side-Channel Attacks using Meta-Transfer Learning. In *Proc. of the 58th ACM/IEEE Design Automation Conference (DAC'21)*, 2021.
52. H. Yu, S. Wang, H. Shan, M. Panoff, M. Lee, K. Yang, and Y. Jin. Dual-Leak: Deep Unsupervised Active Learning for Cross-Device Profiled Side-Channel Leakage Analysis. In *Proc. of IEEE HOST'23*, 2023.
53. F. Zhang, B. Shao, G. Xu, B. Yang, Z. Yang, Z. Qin, and K. Ren. From Homogeneous to Heterogeneous: Leveraging Deep Learning based Power Anlysis across Devices. In *Proc. of 57th ACM/IEEE Design Automation Conference (DAC'20)*, 2020.

References

1. G. Becker, J. Cooper, E. DeMulder, G. Goodwill, J. Jaffe, G. Kenworthy, T. Kouzminov, A. Leiserson, M. Marson, P. Rohatgi, and S. Saab. Test vector leakage assessment (tvla) methodology in practice. In *2013 International Cryptographic Module Conference*, 2013.
2. D. Bellizia, O. Bronchain, G. Cassiers, C. Momin, F. Standaert, and B. Udvarhelyi. Spook sca ctf. `https://doi.org/10.14428/DVN/W2SV5G`, 2021. DOI: 10.14428/DVN/W2SV5G.
3. R. Benadjila, V. Lomne, E. Prouff, and T. Roche. Secure aes128 for atmega8515. GitHub repository, 2018.
4. R. Benadjila, E. Prouff, R. Strullu, E. Cagli, and C. Dumas. Deep learning for side-channel analysis and introduction to ASCAD database. *Journal of Cryptographic Engineering*, 10(2), 2020.
5. S. Bhasin, N. Bruneau, J. Danger, S. Guilley, and Z. Najm. Analysis and improvements of the dpa contest v4 implementation. In S. Chakraborty, V. Matyas, and P. Schaumont, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 201–218, Cham, 2014. Springer International Publishing.
6. S. Bhasin, A. Chattopadhyay, A. Heuser, D. Jap, S. Picek, and R. R. Shrivastwa. Mind the Portability: A Warriors Guide through Realistic Profiled Side-channel Analysis. In *Proc. of NDSS'20*, 2020.
7. S. Bhasin, J. Danger, S. Guilley, and Z. Najm. NICV: Normalized inter-class variance for detection of side-channel leakage. In *2014 International Symposium on Electromagnetic Compatibility*, 2014.
8. E. Brier, C. Clavier, and F. Olivier. Correlation Power Analysis with a Leakage Model. In *Proc. of CHES'04*, 2004.
9. I. Buhan, L. Batina, Y. Yarom, and P. Schaumont. Sok: Design tools for side-channel-aware implementations, 2021.
10. S. Chari, C. Jutla, J. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M. Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 398–412, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.
11. S. Chari, J. R. Rao, and P. Rohatgi. Template Attacks. In *Proc. of Cryptographic Hardware and Embeeded Systems (CHES 2002)*, 2002.
12. "J. Coron and I. Kizhvatov". Analysis and improvement of the random delay countermeasure of ches 2009. In S. Mangard and F. Standaert, editors, *"Cryptographic Hardware and Embedded Systems, CHES 2010"*, pages 95–109, Berlin, Heidelberg, "2010". Springer Berlin Heidelberg.
13. J. Danial, D. Das, A. Golder, S. Ghosh, A. Raychowdhury, and S. Sen. EM-X-DL: Efficient Cross-device Deep Learning Side-channel Attack with Noisy EM Signatures. *ACM Journal on Emerging Technologies in Computing Systems*, 18(1):1–17, 2022.
14. D. Das, A. Golder, J. Danial, S. Ghosh, A. Raychowdhury, and S. Sen. X-DeepSCA: Cross-Device Deep Learning Side Channel Attack. In *Proc. of 56th ACM/IEEE Design Automation Conference (DAC'19)*, 2019.
15. M. Elaabid and S. Guilley. Portability of templates. *Journal of Cryptographic Engineering*, 2:63–74, 2012.

16. C. Genevey-Metat, A. Heuser, and B. Gerard. Train or Adapt a Deeply Learned Profile? In *Proc. of International Conference on Cryptology and Information Security in Latin America (Latin Crypt'21)*, 2021.
17. A. Golder, D. Das, J. Danial, S. Ghosh, S. Sen, and A. Raychowdhury. Practical Approaches Towards Deep-Learning Based Cross-Device Power Side Channel Attack. *IEEE. Trans. on Very Large-Scale Integration (VLSI) Systems*, 27(12), 2019.
18. S. Hajra, S. Chowdhury, and D. Mukhopadhyay. EstraNet: An Efficient Shift-Invariant Transformer Network for Side-Channel Analysis. *TCHES*, 2024.
19. B. Hettwer, S. Gehrer, and T. Guneysu. Deep neural network attribution methods for leakage analysis and symmetric key recovery. Cryptology ePrint Archive, Paper 2019/143, 2019. `https://eprint.iacr.org/2019/143`.
20. P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proc. of CRYPTO'99*, 1999.
21. P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1):5–27, Apr 2011.
22. P. Kulkarni, V. Verneuil, S. Picek, and L. Batina. Order vs. Chaos: A Language Model Approach for Side-channel Attacks. `https://eprint.iacr.org/2023/1615.pdf`.
23. D. Kwon, H. Kim, and S. Hong. Non-Profiled Deep Learning-based Side-Channel Preprocessing with Autoencoders. *IEEE Access*, 2021.
24. H. Li, M. Ninan, B. Wang, and J. Emmert. Tinypower github repository, 2023. `https://github.com/UCdasec/TinyPower`.
25. H. Li, M. Ninan, B. Wang, and J. M. Emmert. TinyPower: Side-Channel Attacks with Tiny Neural Networks. In *Proc. of IEEE HOST'24*, 2024.
26. L. Lin, D. Zhu, J. Wen, H. Chen, Y. Lu, N. Cheng, C. Chow, H. Shrivastav, C. W. Chen, K. Monta, and M. Nagata. Multiphysics Simulation of EM Side-Channels from Silicon Backside with ML-based Auto-POI Identification. In *IEEE HOST'21*, 2021.
27. H. Maghrebi, T. Portigliatti, and E. Proff. Breaking cryptographic implementations using deep learning techniques. In *Proc. of International Conference on Security, Privacy and Applied Cryptography Engineering (SPACE'16)*, 2016.
28. L. Masure and R. Strullu. Side channel analysis against the ANSSI's protected AES implementation on ARM. Cryptology ePrint Archive, Paper 2021/592, 2021. `https://eprint.iacr.org/2021/592`.
29. NewAE. Chipwhisperer. `https://www.newae.com/chipwhisperer`.
30. M. Ninan, E. Nimmo, S. Reilly, C. Smith, W. Sun, B. Wang, and J. Emmert. A second look at the portability of deep learning side-channel attacks over em traces. In *2024 International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2024.
31. M. Panoff, H. Yu, H. Shan, and Y. Jin. A Review and Comparison of AI-enhanced Side Channel Analysis. *J. Emerg. Technol. Comput. Syst.*, 2022.
32. K. Papagiannopoulos, O. Glamocanin, M. Azouaoui, D. Ros, F. Regazzoni, and M. Stojilovic. The Side-channel Metrics Cheat Sheet. *ACM Computing Surveys*, 55(10), 2023.
33. G. Perin, L. Wu, and S. Picek. Gambling for success: The lottery ticket hypothesis in deep learning-based SCA. Cryptology ePrint Archive, Paper 2021/197, 2021. `https://eprint.iacr.org/2021/197`.
34. G. Perin, L. Wu, and S. Picek. I know what your layers did: Layer-wise explainability of deep learning side-channel analysis. Cryptology ePrint Archive, Paper 2022/1087, 2022. `https://eprint.iacr.org/2022/1087`.
35. S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina. Sok: Deep learning-based physical side-channel analysis. *ACM Comput. Surv.*, 55(11), feb 2023.

36. S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina. SoK: Deep Learning-based Physical Side-channel Analysis. *ACM Computing Surveys*, 55(11), 2023.
37. J. Rijsdijk, L. Wu, G. Perin, and S. Picek. Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021.
38. U. Rioja, L. Batina, and I. Armendariz. When Similarities Among Devices are Taken for Granted: Another Look at Portability. In *Proc. of AFRICACRYPT 2020*, pages 337 – 357, 2020.
39. I. Savu, M. Krcek, G. Perin, L. Wu, and S. Picek. The Need for MORE: Unsupervised Side-channel Analysis with Single Network Training and Multi-output Regression. `https://eprint.iacr.org/2023/1681.pdf`.
40. N. Sehatbakhsh, B. Yilmaz, A. Zajic, and M. Prvulovic. Emsim: A microarchitecture-level simulation tool for modeling electromagnetic side-channel signals. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 71–85, 2020.
41. D. Shanmugam and P. Schaumont. Improving Side-Channel Leakage Assessment Using Pre-Silicon Leakage. In *International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE)*, 2023.
42. B. Timon. Non-Profiled Deep Learning-based Side-Channel Attacks with Sensitivity Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(2):107–131, 2019.
43. C. Wang, M. Ninan, S. Reilly, J. Ward, W. Hawkins, B. Wang, and J. M. Emmert. Portability of Deep-Learning Side-Channel Attacks against Software Discrepancies. In *Proc. ACM WiSec'23*, 2023.
44. C. Wang, M. Ninan, S. Reilly, J. Ward, W. Hawkins, B. Wang, and J. M. Emmert. Softpower github repository, 2023. `https://github.com/UCdasec/SoftPower`.
45. H. Wang, M. Brisfors, S. Forsmark, and E. Dubrova. 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)*, 2019.
46. P. Wang, P. Chen, Z. Luo, G. Dong, M. Zheng, N. Yu, and H. Hu. Enhancing the Performance of Practical Profiling Side-Chanel Attacks Using Conditional Generative Adversarial Networks. https://arxiv.org/abs/2007.05285.
47. L. Wu, A. Ali-Pour, A. Rezaeezade, G. Perin, and S. Picek. Breaking Free: Leakage Model-free Deep Learning-based Side-channel Analysis. https://eprint.iacr.org/2023/1110.pdf.
48. L. Wu, G. Perin, and S. Picek. Hiding in Plain Sight: Non-profiling Deep Learning-based Side-channel Analysis with Plantext/Ciphertext. https://eprint.iacr.org/2023/209.pdf.
49. L. Wu, G. Perin, and S. Picek. I Choose You: Automated Hyperparameter Tuning for Deep Learning-based Side-Channel Analysis. *IEEE Transactions on Emerging Topics in Computing*, 2022.
50. L. Wu, S. Tiran, G. Perin, and S. Picek. An End-to-end Plaintext-based Side-channel Collision Attack without Trace Segmentation. `https://eprint.iacr.org/2023/1109.pdf`.
51. H. Yu, H. Shan, M. Panoff, and Y. Jin. Cross-Device Profiled Side-Channel Attacks using Meta-Transfer Learning. In *Proc. of the 58th ACM/IEEE Design Automation Conference (DAC'21)*, 2021.
52. H. Yu, S. Wang, H. Shan, M. Panoff, M. Lee, K. Yang, and Y. Jin. Dual-Leak: Deep Unsupervised Active Learning for Cross-Device Profiled Side-Channel Leakage Analysis.

In *Proc. of IEEE HOST'23*, 2023.
53. F. Zhang, B. Shao, G. Xu, B. Yang, Z. Yang, Z. Qin, and K. Ren. From Homogeneous to Heterogeneous: Leveraging Deep Learning based Power Anlysis across Devices. In *Proc. of 57th ACM/IEEE Design Automation Conference (DAC'20)*, 2020.