Chapter 8
It Started with Templates: The Future of Profiling in Side-Channel Analysis

Lejla Batina, Milena Djukanovic, Annelie Heuser, and Stjepan Picek

Abstract Side-channel attacks (SCAs) are powerful attacks based on the information obtained from the implementation of cryptographic devices. Profiling side-channel attacks has received a lot of attention in recent years due to the fact that this type of attack defines the worst-case security assumptions. The SCA community realized that the same approach is actually used in other domains in the form of supervised machine learning. Consequently, some researchers started experimenting with different machine learning techniques and evaluating their effectiveness in the SCA context. More recently, we are witnessing an increase in the use of deep learning techniques in the SCA community with strong first results in side-channel analyses, even in the presence of countermeasures. In this chapter, we consider the evolution of profiling attacks, and subsequently we discuss the impacts they have made in the data preprocessing, feature engineering, and classification phases. We also speculate on the future directions and the best-case consequences for the security of small devices.

8.1 Introduction

In 1996, Kocher demonstrated the possibility to recover secret data by introducing a method for exploiting leakages from the device under attack [338]. In other words, implementations of cryptographic algorithms leak relevant information

L. Batina
Radboud University, Nijmegen, The Netherlands

M. Djukanovic
University of Montenegro, Podgorica, Montenegro

A. Heuser
Univ Rennes, Inria, CNRS, IRISA, Rennes, France

S. Picek (✉)
Delft University of Technology, Delft, The Netherlands

G. Avoine, J. Hernandez-Castro (eds.), *Security of Ubiquitous Computing Systems*,
https://doi.org/10.1007/978-3-030-10591-4_8

about the data processed through physical side-channels such as timing [338], power consumption [339], EM emanation [493], and sound [225].

Side-channel attacks (SCAs) exploit weaknesses in the physical implementation of cryptographic algorithms rather than the algorithms themselves [389]. Those weaknesses stem from the physics of the underlying computing elements, i.e., CMOS cells, which makes it hard to eliminate such threats.

Numerous evaluation techniques, which generally involve some form of digital signal processing and statistical computations, have been proposed in the literature. Some of the most important methods include Simple Power Analysis (SPA) [339], Differential Power Analysis (DPA), and Template Attacks (TA) [135].

The SPA technique implies that the attacker aims at reconstructing the secret key using just a single trace of side-channel information, and it often exploits the difference in basic public-key operations such as double-and-add, or add-and-multiply [339]. Still, SPA is not possible if the observed signal-to-noise ratio (SNR) is not high enough. Consequently, most of the time developed countermeasures make SPA futile.

DPA techniques are based on the evaluation of many traces with varying input data for the targeted algorithm. After that step, a brute-force attack, testing sub-key hypotheses, is performed on a part of the algorithm (so-called "divide and conquer"). In the DPA approach, a large number of samples are used in order to reduce noise by averaging, and a single-bit power model is commonly adopted [339]. On the other hand, Correlation Power Analysis (CPA) represents a multi-bit power model in order to reduce the influence of noise on the possibility to execute a successful attack [115]. The main difference between these two techniques is that DPA is based on computing the difference between two trace sets, while CPA uses the correlation coefficient in order to calculate the dependency test. We often also say that the two use different side-channel distinguishers. Side-channel attacks using the above three techniques have been reported on a wide variety of cryptographic implementations, see, e.g., [154, 402, 410, 412, 434, 500] including some real-world applications [196].

In contrast to DPA, TA requires a profiling stage, i.e., a step during which the cryptographic hardware is under full control of the adversary to estimate the probability distribution of the leaked information and make better use of all the information present in each sample [135]. In this way, TA can provide a promising model of the real device, instead of using some a priori model.

TA is the best (optimal) technique from an information-theoretic point of view if the attacker has an unbounded number of traces and the noise follows the Gaussian distribution [277, 367]. After the template attack, the stochastic attack emerged using linear regression in the profiling phase [515]. In the years that followed, researchers recognized certain shortcomings of template attacks and tried to modify them in order to deal better with the complexity and portability issues. An example of such an approach is the pooled template attack where only one pooled covariance matrix is used in order to cope with statistical difficulties [142]. Alongside such techniques, the SCA community realized that a similar approach to profiling is used in other domains in the form of supervised machine learning. Consequently, some researchers started experimenting with different machine learning (ML) techniques and evaluating their effectiveness in the SCA context. Although mainly considering distinct scenarios and various ML techniques, all those papers tend to establish different use cases where ML techniques can outperform the template attack and establish themselves as the best choice for profiled SCA. More recently, we are witnessing the relevance of deep learning (DL) techniques in the SCA community with strong results in side-channel analyses, even in the presence of countermeasures.

8.2 Profiled Side-Channel Attacks

Profiled side-channel attacks estimate the worst-case security risk by considering the most powerful side-channel attacker. In particular, one assumes that an attacker can possess an additional device of which he or she has nearly full control. From this device, he obtains leakage measurements and is able to control the used secret key or at least knows which one is used. Knowing the secret key enables him to calculate intermediate processed values that involve the secret key for which he is estimating models. These models can then be used in the attacking phase to predict which intermediate values are processed and therefore carry information about the secret key. Commonly used models are the identity value or Hamming weight/distance.

Uniformly Distributed Classes Targeting intermediate variables, e.g., when loaded or manipulated on the device and resulting mostly in 2^n uniformly distributed classes where n is the number of bits of the intermediate variable.

Binomial Distributed Classes The Hamming Weight (HW) or the Hamming Distance (HD) of a uniformly distributed intermediate variable results in $n + 1$ binomially distributed classes.

8.2.1 Definition of Profiling Attacks

In this section, we consider side-channel attacks on block ciphers for which a divide and conquer approach can be utilized. Note that, as there exist operations within the block cipher which manipulate each block/chunk (e.g., bytes in Advanced Encryption Standard (AES)) independently and most importantly involving only one block/chunk of the secret key, an attacker only needs to make hypotheses about the secret key block/chunk instead of the complete secret key at once.

More formally, let k^* denote (a chunk of) the fixed secret cryptographic key that is stored on the device and let t denote (a chunk of) the plaintext or ciphertext of the cryptographic algorithm. The mapping y maps the plaintext or the ciphertext

$t \in \mathscr{T}$ and the key $k^* \in \mathscr{K}$ to an intermediate value that is assumed to relate to the deterministic part of the measured leakage x. For example,

$$y(t, k) = \texttt{Sbox}[T \oplus k], \tag{8.1}$$

where $\texttt{Sbox}[\cdot]$ is a substitution operation. The measured leakage x can then be written as

$$x = \varphi(y(t, k^*)) + r, \tag{8.2}$$

where r denotes independent additive noise and φ is a device-specific (unknown) deterministic function mapping the intermediate variable to the leakage space. In the rest of this chapter, we are particularly interested in multivariate leakage $\mathbf{x} = x_1, \ldots, x_D$, where D is the number of time samples, i.e., features (also called attributes or points of interest).

Now, it is considered that the attacker has the following information at his disposal to conduct the attack:

- *profiling phase:* N traces (measurements) $\mathbf{x}_{p_1}, \ldots, \mathbf{x}_{p_N}$, the secret key k_p^*, and plaintexts/ciphertexts $t_{p_1}, \ldots, t_{p_N}$, such that he can calculate $y(t_{p_1}, k_p^*), \ldots, y(t_{p_N}, k_p^*)$.
- *attacking phase:* Q traces $\mathbf{x}_{a_1}, \ldots, \mathbf{x}_{a_Q}$ (independent from the profiling traces), plaintexts/ciphertexts $t_{a_1}, \ldots, t_{a_Q}$.

In the attacking phase the goal is to make predictions about $y(t_{a_1}, k_a^*), \ldots, y(t_{a_N}, k_a^*)$, where k_a^* is the secret key on the attacking device. Note, even before running the attack, there are several steps one could do in order to make the attack more powerful. These phases are depicted in Fig. 8.1.

8.2.2 Data Preprocessing

In the data preprocessing phase, the aim is to prepare the data in a way to increase the performance of side-channel analysis. There are several papers considering various data augmentation techniques in order to artificially generate measurements so as to increase the size of the profiling dataset. Cagli et al. propose two data

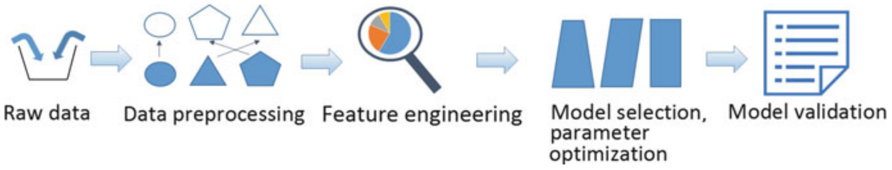

Fig. 8.1 Depiction of an end-to-end profiling attack

augmentation techniques they call Shifting and Add-Remove [122]. They use convolutional neural networks (CNN) and find data augmentation to significantly improve the performance of CNN. Pu et al. use a data augmentation technique where they randomly shift each measurement in order to increase the number of measurements available in the profiling phase [489]. They report that even such simple augmentation can effectively improve the performance of profiling SCA. Picek et al. experiment with several data augmentation and class balancing techniques in order to decrease the influence of highly unbalanced datasets that occur when considering HW/HD models [478]. They show that by using a well-known machine learning technique called SMOTE, it is possible to reduce the number of measurements needed for a successful attack by up to 10 times. Kim et al. investigate how the addition of artificial noise to the input signal can be beneficial to the performance of the neural network [329].

8.2.3 Feature Engineering

When discussing the feature engineering tasks, we can recognize a few directions that researchers follow in the context of SCA:

- feature selection. Here, the most important subsets of features are selected. We can distinguish between filter, wrapper, and hybrid techniques.
- dimensionality reduction. The original features are transformed into new features. A common example of such a technique is Principal Component Analysis (PCA) [25].

When discussing feature engineering, it is important to mention the curse of dimensionality. This describes the effects of an exponential increase in volume associated with the increase in the dimensions [71]. As a consequence, as the dimensionality of the problem increases, the classifier's performance increases until the optimal feature subset is reached. Further increasing the dimensionality without increasing the number of training samples results in a decrease in the classifier performance.

In the SCA community, there are several standard techniques to conduct feature selection:

- Pearson Correlation Coefficient. The Pearson correlation coefficient measures the linear dependence between two variables, x and y, in the range $[-1, 1]$, where 1 is a total positive linear correlation, 0 is no linear correlation, and -1 means a total negative linear correlation. The Pearson correlation for a sample of the entire population is defined by [301]:

$$Pearson(x, y) = \frac{\sum_{i=1}^{N}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}}, \tag{8.3}$$

where $\bar{x}$ and $\bar{y}$ are the empirical means of x and y, respectively.

- SOSD. In [230], the authors proposed as a selection method the sum of squared differences, simply as:

$$SOSD(x, y) = \sum_{i,j>i} (\bar{x}_{y_i} - \bar{x}_{y_j})^2, \tag{8.4}$$

where $\bar{x}_{y_i}$ is the mean of the traces where the model equals y_i. Because of the square term, SOSD is always positive. Another advantage of using the square is that it enlarges big differences.

- SOST. SOST is the normalized version of SOSD [230] and is thus equivalent by the pairwise student T-test:

$$SOST(x, y) = \sum_{i,j>i} \left((\bar{x}_{y_i} - \bar{x}_{y_j}) / \sqrt{\frac{\sigma^2_{y_i}}{n_{y_i}} + \frac{\sigma^2_{y_j}}{n_{y_j}}} \right)^2 \tag{8.5}$$

with n_{y_i} and n_{y_j} being the number of traces where the model equals to y_i and y_j, respectively.

There are several more relevant works in the domain of feature selection and SCA. The work of Lerman et al. [367] compared template attacks and machine learning on dimensionality reduction. They concluded that template attacks are the method of choice as long as a limited number of features can be identified in leakage traces containing most of the relevant information. Zheng et al. looked into feature selection techniques but they did not consider machine learning options [600]. Picek et al. conducted a detailed analysis of various feature selection techniques where some are also based on machine learning (so-called wrapper and hybrid methods) [477]. They concluded that commonly used feature selection techniques in SCA are rarely the best ones and they mentioned L1 regularization as a powerful feature selector in many scenarios.

8.3 Template Attacks

In this section, we start by explaining the details of template attacks, and after that we give details about two techniques that emerged from template attacks—pooled template attacks and stochastic attacks.

8.3.1 Context of Template Attack

In the pioneering template attacks article of Chari, Rao, and Rohatgi, it is shown that template attacks apply advanced statistical methods and can break implementations secure against other forms of side-channel attacks [135].

In some works template attacks are built to classify the state of a byte, e.g., a key byte in RC4 [135, 498]. The weakness of these papers is the need to create 256 templates for each byte. Additionally, the template building process can only be guided by partial attack results. In [498], the authors reduce the number of points of a trace by using an efficient algorithm instead of the standard principal component analysis method, which increases the speed of selecting points of interest. Also, by introducing a preprocessing phase with the use of discrete Fourier transformation on traces, the authors improve the template attack results in practice.

Agrawal et al. develop two new attack techniques that extend the work of the previously mentioned research results [11]. The first is a single-bit template attack technique that creates templates from peaks observed in a DPA attack resulting with a high probability value of a single DPA-targeted bit. Their second, template-enhanced DPA attack technique can be used to attack DPA protected cards and consists of two steps: a profiling phase and a hypothesis testing phase. In the first, profiling phase, the attacker, who is in possession of a smart card with a biased RNG, builds templates, and in the hypothesis testing phase the attacker uses previously built templates to mount a DPA-like attack on a target card which is identical to the test smart card, but has a perfect RNG. The authors illustrate these two attack techniques considering unprotected implementations of DES and AES on smart cards.

Archambeau et al. take template attacks techniques a step further by transforming leakage traces in order to identify important features (i.e., transformed time instants) and their number automatically. Actually, they use the optimal linear combination of the relevant time samples and execute template attacks in the principal subspace of the mean traces creating a new approach, the principal subspace-based template attack (PSTA) [25]. The authors validate this approach by attacking the RC4 stream cipher implementation and an FPGA implementation of AES.

In the literature, the main focus is on template attacks aiming at recovering the secret key of a cryptographic core from measurements of its dynamic power consumption. But with scaling of technology, static power consumption grows faster and creates new issues in the security of smart card hardware. Therefore, Bellizia et al. proposed Template Attack Exploiting Static Power (TAESP) in order to extract information from a hardware implementation of a cryptographic algorithm using temperature-dependence of static currents as a source of information leakage [70].

8.3.2 *Standard Template Attack*

The template attack is based on the Bayesian rule and works under the simplifying assumption that the measurements are mutually independent among the D features given the target class. More precisely, given the vector of N observed attribute values x, the posterior probability for each class value y is computed as:

$$p(Y = y|\mathbf{X} = \mathbf{x}) = \frac{p(Y = y)p(\mathbf{X} = \mathbf{x}|Y = y)}{p(\mathbf{X} = \mathbf{x})}, \tag{8.6}$$

where $\mathbf{X} = \mathbf{x}$ represents the event that $\mathbf{X}_1 = \mathbf{x}_1 \wedge \mathbf{X}_2 = \mathbf{x}_2 \wedge \ldots \wedge \mathbf{X}_N = \mathbf{x}_N$.

Note that the class variable Y and the measurement X are not of the same type: Y is discrete while X is continuous. So, the discrete probability $p(Y = y)$ is equal to its sample frequency where $p(\mathbf{X} = \mathbf{x}|Y = y)$ displays a density function. In most state-of-the-art models $p(\mathbf{X} = \mathbf{x}|Y = y)$ is assumed to be based on a (multivariate) normal distribution and is thus parameterized by its mean and its covariance matrix:

$$p(\mathbf{X} = \mathbf{x}|Y = y) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_y|}}e^{-\frac{1}{2}(\mathbf{x}-\bar{x}_y)^T\Sigma_y^{-1}(\mathbf{x}-\bar{x}_y)}. \tag{8.7}$$

8.3.3 *Pooled Template Attack*

In practice, the estimation of the covariance matrices for each class value y can be ill-posed mainly due to an insufficient number of traces for each class. The authors of [142] propose to use only one pooled covariance matrix to cope with statistical difficulties and thus a lower efficiency. Accordingly, Eq. (8.7) changes to

$$p(\mathbf{X} = \mathbf{x}|Y = y) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}}e^{-\frac{1}{2}(\mathbf{x}-\bar{x}_y)^T\Sigma^{-1}(\mathbf{x}-\bar{x}_y)}. \tag{8.8}$$

The works in, e.g., [142, 476, 477, 481] showed that indeed the pooled version is more efficient, in particular for a smaller number of traces in the profiling phase.

8.3.4 *Stochastic Attack*

Compared to TA, the stochastic attack (SA) utilizes linear regression instead of probability density estimation [515]. One critical aspect of SA is the choice of regressors (aka base functions), as for example shown in [275]. A natural choice in the context of side-channel analysis is the bitwise selection of the intermediate variable, i.e., let $[\cdot]_b$ define the function selecting the bth bit and using the same intermediate variable as in Sect. 8.2.1 then

$$([\mathtt{Sbox}[T \oplus k]]_1 \quad [\mathtt{Sbox}[T \oplus k]]_2 \quad \ldots \quad [\mathtt{Sbox}[T \oplus k]]_n) \tag{8.9}$$

is an n-dimensional vector used as regressors. One benefit of SA is the constructive feedback of side-channel leakage detection it might bring to the evaluator (see, e.g., [278]).

8.4 Machine Learning-Based Attacks

Machine learning encompasses a number of methods used for classification, clustering, regression, feature selection, and other knowledge discovering methods [423]. A typical division of machine learning algorithms is into supervised, semi-supervised, and unsupervised approaches. Each of those paradigms can also be used in SCAs—supervised (profiling) attacks, semi-supervised attacks (profiling), unsupervised (non-profiling) attacks.

In Fig. 8.2, we depict differences in the supervised and semi-supervised cases.

Supervised Techniques

The supervised approach assumes that the attacker first possesses a device similar to the one under attack. Having this additional device, he is then able to build a precise profiling model using a set of measurements while knowing the plaintext/ciphertext and the secret key of this device. In the second step, the attacker uses the earlier profiling model to reveal the secret key of the device under attack. For this, he additionally measures a new set of traces, but as the key is secret he has no further information about the intermediate processed data and thus builds hypotheses. The only information that the attacker transfers between the profiling phase and the attacking phase is the profiling model he builds.

When considering supervised machine learning and SCA, in recent years there have been numerous papers considering various targets, machine learning algorithms, and scenarios. Actually, the most common denominator for most of the work

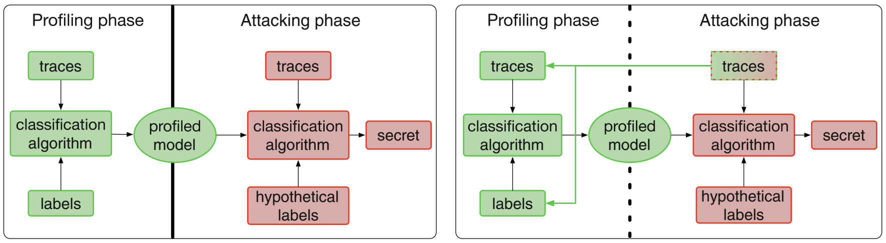

Fig. 8.2 Profiling side-channel scenario: supervised (left), semi-supervised (right)

is the fact that they attack AES [235, 274, 279, 285, 363–365, 367, 475, 476, 479, 481]. More recently, deep learning (DL) techniques started to capture the attention of the SCA community. Accordingly, the first results confirmed expectations, with most of the early attention being paid to convolutional convolutional neural networks [122, 329, 386, 482].

As far as we know, when considering machine learning-based attacks on other ciphers, there are only a few papers. Heuser et al. consider Internet of Things scenarios and lightweight ciphers where they compare 11 lightweight ciphers and AES in terms of their SCA resilience and conclude that lightweight ciphers cannot be considered to be significantly less resilient than AES [274, 276].

Semi-supervised Techniques
Semi-supervised learning is positioned in the middle between supervised and unsupervised learning. There, the basic idea is to take advantage of a large quantity of unlabeled data during a supervised learning procedure [517]. This approach assumes that the attacker is able to possess a device to conduct a profiling phase but has limited capacities. This may reflect a more realistic scenario in some practical applications, as the attacker may be limited by time or resources, or also face implemented countermeasures, which prevent him from taking an arbitrarily large amount of side-channel measurements while knowing the secret key of the device.

The first application of semi-supervised SCA was done by Lerman et al., where the authors conclude that the semi-supervised setting cannot compete with a supervised setting [366]. Note, the authors compared the supervised attack with $n + m$ labeled traces for all classes with a semi-supervised attack with n labeled traces for one class and m unlabeled traces for other unknown classes (i.e., in total $n + m$ traces). Picek et al. conduct an analysis of two semi-supervised paradigms (self-training and graph-based learning) where they show that it is possible to improve the accuracy of classifiers if semi-supervised learning is used [480]. What is especially interesting is that they show how semi-supervised learning is able to significantly improve the behavior of the template attack when the profiling set is (very) small.

8.4.1 Conducting Sound Machine Learning Analysis

Since it is not possible (in general) to expect machine learning techniques to give us theoretical observations or proofs of results, we need to rely on a set of procedures to run experiments such that the results are convincing and easy to reproduce. In the next section, we briefly discuss several steps to be considered in order to make the analysis more reproducible.

Datasets
When preparing the data for machine learning analysis, it is necessary to discuss the number of measurements, the number of features, and the number of classes (if known). Additionally, if the data come from different distributions, one needs to

discuss those. If not all data from datasets are used, it is necessary to state how the samples are chosen and how many are used in the experiments. One needs to define the level of noise appearing in the data in a clearly reproducible way, e.g., using the signal-to-noise ratio (SNR). Finally, if some feature engineering procedure is used, it needs to be clearly stated in order to know what features are used in the end.

Algorithms
When discussing the choice of algorithms, first it is necessary either to specify which framework and algorithms are used or provide pseudo-code (for example, when custom algorithms are used). As a rule of thumb, more than one algorithm should always be used: the algorithms should ideally belong to different machine learning approaches (e.g., a decision tree method like Random Forest and a kernel method like Support Vector Machine (SVM)). Next, all parameters that uniquely define the algorithm need to be enumerated.

Experiments
Regarding the experiments, it is first necessary to discuss how the data are divided into training and testing sets. Then, for the training phase, one needs to define the test options (e.g., whether to use the whole dataset or cross-validation, etc.) After that, for each algorithm, one needs to define a set of parameter values to conduct the tuning phase. There are different options for tuning, but we consider starting with the default parameters as a reasonable approach and continue varying them until there is no more improvement. Naturally, this should be done in a reasonable way, since the tuning phase is the most expensive from the computational perspective and it is usually not practical to test all combinations of parameters.

Results
For the tuning phase, it is usually sufficient to report the accuracy. For the testing results, one should report the accuracy but also some other metric like the area under the ROC curve (AUC) or the F-measure. The area under the ROC curve is used to measure the accuracy and is calculated via Mann-Whitney statistics [580]; the ROC curve is the ratio between the true positive rate and the false positive rate. An AUC close to 1 represents a good test, while a value close to 0.5 represents a random guess. The F-measure is the harmonic mean of the precision and recall, where precision is the ratio between true positive (TP, the number of examples predicted positive that are actually positive) and predicted positive. The recall is the ratio between true positives and actual positives [488]. Both the F-Measure and the AUC can help in situations where accuracy can be misleading, i.e., where we are also interested in the number of false positive and false negative values.

8.5 Performance Metrics

When considering profiling SCA, there are three performance metrics we mention: accuracy, guessing entropy, and success rate. The accuracy is the proportion between the correctly classified measurements and all measurements:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{8.10}$$

TP refers to true positive (correctly classified positive), TN to true negative (correctly classified negative), FP to false positive (falsely classified positive), and FN to false negative (falsely classified negative) instances. TP, TN, FP, and FN are well-defined for hypothesis testing and binary classification problems. In the multi-class classification, they are defined in a one class vs. all other classes manner, and are calculated from the confusion matrix.

A side-channel adversary $A_{E_K,L}$ conducts an experiment $\mathsf{Exp}_{A_{E_K,L}}$, with time-complexity τ, memory complexity m, and making Q queries to the target implementation of the cryptographic algorithm. The attack outputs a guessing vector g of length o, and is considered a success if g contains correct key k^*. o is also known as the order of the success rate. The oth order success rate of the side channel attack $\mathsf{A}_{\mathsf{E}_{\mathsf{K}},\mathsf{L}}$ is defined as:

$$\mathrm{SR}^{o}_{A_{E_K,L}}(\tau, m, k^*) = \mathsf{Pr}[\mathrm{Exp}_{A_{E_K,L}} = 1] \tag{8.11}$$

The Guessing entropy measures the average number of key candidates to test after the attack. The $\mathsf{Guessing\ entropy}$ of the adversary $A_{E_k,L}$ against a key class variable S is defined as:

$$GE_{A_{E_K,L}}(\tau, m, k^*) = \mathsf{E}[\mathrm{Exp}_{A_{E_K,L}}] \tag{8.12}$$

8.6 Countermeasures Against SCA

There are various countermeasures against SCAs that have been proposed over the years. A general approach focuses on decreasing the information gathered from the measurements:

- Noise Addition. Introducing external noise in the side-channel, shuffling the operations or inserting dummy operations in cryptographic implementations is often used as a countermeasure against SCAs. The basic objective is to reduce the signal-to-noise ratio (SNR) and thereby decrease the information gathered from measurements. Still, as shown already by Durvaux et al. [194], these countermeasures become insecure with increasing attack time.

- Dynamic and Differential CMOS Logic. Tiri et al. [557] proposed Sense Amplifier Based Logic (SABL)—a logic style that uses a fixed amount of charge for every transition, including the degenerated events in which a gate does not change state.
- Leakage Resilience. Another countermeasure, typically applied at the system level, focuses on restricting the number of usages of the same key for an algorithm. Still, generation and synchronization of new keys have practical issues. Dziembowski et al. introduced a technique called leakage resilience, which relocates this problem to the protocol level by introducing an algorithm to generate these keys [195].
- Masking. One of the most efficient and powerful approaches against SCAs is masking [134, 243], which aims to break the correlation between the power traces and the intermediate values of the computations. This method achieves security by randomizing the intermediate values using secret sharing and carrying out all the computations on the shared values.

8.7 Conclusions

In this chapter, we discussed profiling side-channel attacks where we started with data preprocessing and feature engineering. Then we presented several template-like techniques and afterward machine learning techniques. Next, we discussed how to conduct a sound machine learning analysis that should result in reproducible experiments. We finished the chapter with a short discussion on how to test the performance of SCA and what are some of the possible countermeasures to make such attacks more difficult.