Journal of Cryptographic Engineering (2020) 10:135–162
https://doi.org/10.1007/s13389-019-00212-8

REGULAR PAPER

Applications of machine learning techniques in side-channel attacks: a survey

Benjamin Hettwer[1] · Stefan Gehrer[1] · Tim Güneysu[2]

Received: 11 June 2018 / Accepted: 29 March 2019 / Published online: 11 April 2019

Abstract
With increasing expansion of the Internet of Things, embedded devices equipped with cryptographic modules become an important factor to protect sensitive data. Even though the employed algorithms in such devices are mathematically secure in theory, adversaries may still be able to compromise them by means of side-channel attacks. In power-based side-channel attacks, the instantaneous power consumption of the target is analyzed with statistical tools to draw conclusions about the secret keys that are used. There is a recent line of work that additionally makes use of techniques from the machine learning domain to attack cryptographic implementations. Since a complete review of this emerging field has not been done so far, this research aims to survey the current state of the art. We use a target-based classification to differentiate published work and drive general conclusions according to a common machine learning workflow. Furthermore, we outline the relationship between traditional power analysis techniques and machine learning-based attacks. This enables researchers to gain a better understanding of the topic in order to design new attack methods as well as potential countermeasures.

Keywords Side-channel attacks · Power analysis · Machine learning · Deep learning

1 Introduction

Since Paul Kocher published the first publicly-known side-channel attack (SCA) on several public-key cryptosystems back in [60], there has been a lot of attraction in the security community for physical attack vectors such as timing, power consumption [58], electromagnetic (EM) radiations [103], or even sound [31]. Rather than traditional cryptanalysis, which aims to exploit theoretical weaknesses in the cryptographic algorithms themselves, SCAs focus on the actual implementation in software or hardware to recover the secret key. Although the vast majority of the studies on SCAs have been carried out to break cryptographic systems, it has been shown that the underlying principle may pose other kind of threats as well. Examples are acoustic attacks on keyboards to reveal typed text [142], or power analysis of an embedded CPU to retrieve information about the executed instructions [28]. Typical attack setups include visual inspection of the physical traces, statistical methods, and information theory. Power-based SCAs can be roughly divided in non-profiled attacks (e.g. simple/differential power analysis [58]) and profiled attacks (template attacks, stochastic approach [21,117]).

Machine Learning (ML) systems, generally speaking, are able to improve their performance on a specific task with increasing experience [40]. In a typical classification problem, the system is fed with training examples consisting of input data vectors (features) and associated outcome measurements (labels) which is generally referred to as *supervised* learning. During training, the algorithm makes predictions on the basis of the input data and is corrected if those predictions do not match the expected labels. The goal is to build a prediction model that generalizes well on unseen examples, meaning it produces the correct outcome for inputs that have not been part of the training data. *Unsupervised* learning, in contrast, deals with tasks where no outcome labels are available. The learning algorithm tries to deduce useful properties or the underlying structure of the

✉ Benjamin Hettwer
benjamin.hettwer@de.bosch.com

Stefan Gehrer
stefan.gehrer@de.bosch.com

Tim Güneysu
tim.gueneysu@rub.de

1 Corporate Sector Research, Robert Bosch GmbH, Stuttgart, Germany

2 Horst Görtz Institute for IT-Security, Ruhr University Bochum, Bochum, Germany

input data set, e.g. by clustering it into different classes. *Semi-supervised* learning lies somewhere in-between supervised and unsupervised learning and describes settings at which outcome labels are present for a part of the training examples but not for the whole data set.

ML is widely used in many domains such as natural language processing, image recognition, or robotics, and is gaining further importance for future autonomous systems [54]. In addition, there has been also a large amount of papers presented during the last years that incorporate ML techniques for SCAs. Jap et al. summarized a subset of the work which deals with the application of ML to analyze power or EM side-channels of cryptographic implementations [51]. They noted that there is a strong analogy between supervised ML problems and profiled SCAs as well as between unsupervised ML and non-profiled SCAs. However, to the best of our knowledge, there has been no thorough analysis of the topic.

1.1 Contribution and structure of the paper

In this work, we give a broad overview of how ML techniques have been used in SCAs to break different kinds of cryptographic primitives. As illustrated in Fig. 1, the first papers that deal with ML for side-channel analysis published in 2011 targeted unprotected implementations of symmetric ciphers. Later, the community also began to consider asymmetric and stream cipher implementations, as well as SCA protected implementations. There are furthermore a couple of recent publications which are based on advanced Deep Learning (DL) techniques. For each attack vector, we review the approaches presented in the literature and compare them according to several criteria such as the number of physical traces needed or the used feature engineering methods. While this study is not intended to provide an in-depth analysis of each approach, our contribution is rather to point out which ML methods and according parameters are more suitable for attacking a certain cryptographic system. On the one hand, we aim to give helpful insights to build advanced attack methods based on ML in the future, and on the other, we want to raise awareness of developers of security critical applications for such kind of threats.

The remaining part of the paper is structured as follows: Sect. 2 provides background on the different ML techniques utilized in power and EM-based SCAs. In Sect. 3, we introduce standard methods for side-channel analysis apart from ML techniques. These are often referenced as baseline in the analyzed papers in order to asses the quality of the approach. The following subsections cover a review of the individual papers divided in four categories: implementation attacks against block ciphers, stream ciphers, asymmetric ciphers, and other cryptographic constructions

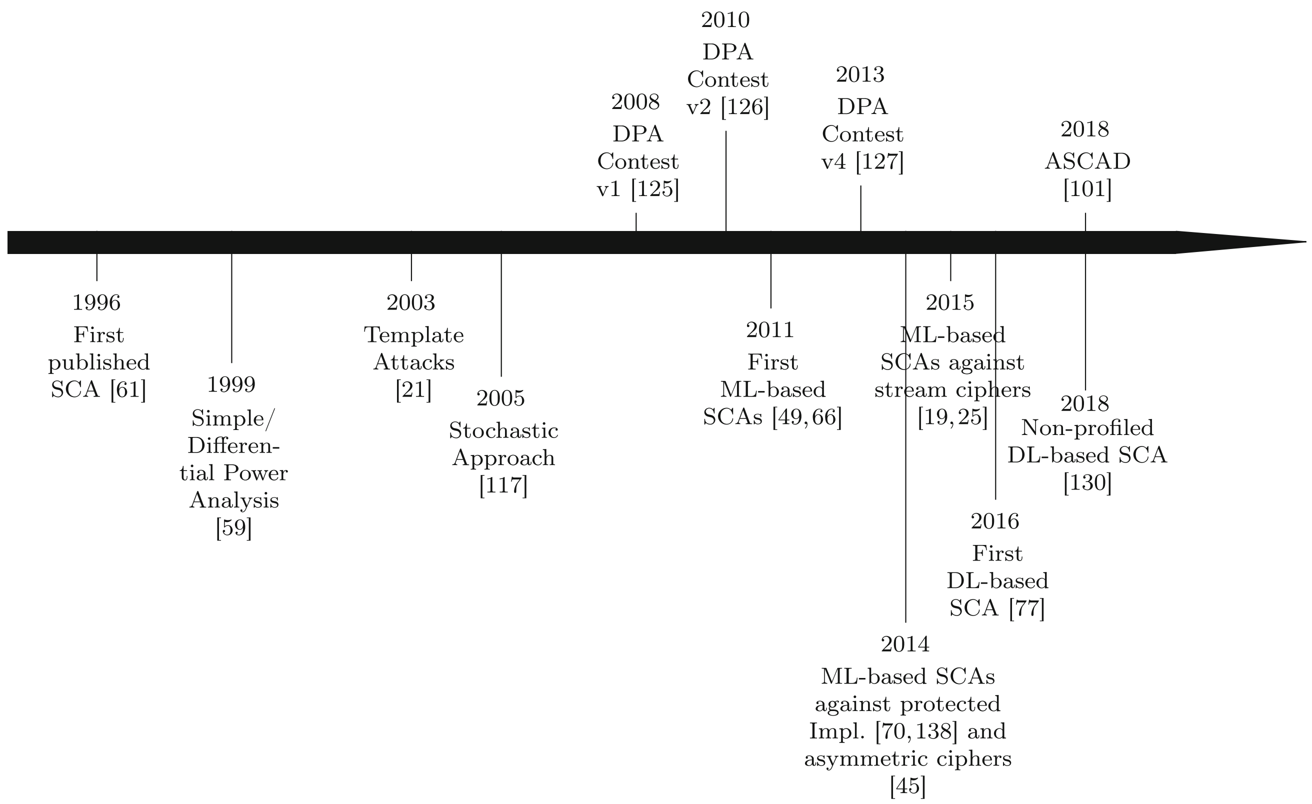
2008 DPA Contest v1 [125]
2010 DPA Contest v2 [126]
2013 DPA Contest v4 [127]
2018 ASCAD [101]
1996 First published SCA [61]
1999 Simple/ Differential Power Analysis [59]
2003 Template Attacks [21]
2005 Stochastic Approach [117]
2011 First ML-based SCAs [49, 66]
2015 ML-based SCAs against stream ciphers [19, 25]
2018 Non-profiled DL-based SCA [130]
2016 First DL-based SCA [77]
2014 ML-based SCAs against protected Impl. [70, 138] and asymmetric ciphers [45]

Fig. 1 Excerpt of the analyzed publications in chronological order. The most used public datasets are shown above the timeline

such as Physical Unclonable Functions (PUFs). Next, we discuss observed commonalities and differences and derive some general recommendations useful to know when considering ML techniques in SCAs. The discussion is augmented by Sect. 6, which gives hints how to properly conduct and report ML experiments for side-channel analysis. We conclude our work with a brief summary and possible future work.

2 Background on machine learning techniques

In this section, we introduce general ML-related terminology before we review several algorithms applied for side-channel analysis.

2.1 Notations and general terminology

Throughout the paper, unless noted otherwise, we use lowercase bold letters to refer to vectors (e.g. a specific training sample is denoted as $\mathbf{x} \in \mathbb{R}^{n\times 1}$), and uppercase bold letter for matrices (e.g. $\mathbf{X} \in \mathbb{R}^{n\times m}$).

In supervised learning, the training set consists of N input–output pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ and the goal is to find a suitable relationship in order to map new inputs to the correct label y. Unsupervised learning algorithms only work on the input data $\{\mathbf{x}_i\}_{i=1}^{N}$, trying to extract meaningful patterns and relations [87]. The goal of almost every ML problem either supervised or unsupervised is to find a model $g(\mathbf{W})$ that explains the distribution of the input data set. It is obtained by fitting its parameters $\mathbf{W}$ to minimize a cost or error function E, which allows to judge how well the model performs on the input data.

In SCA settings, an input training example $\mathbf{x}$ is usually a vector of numbers representing a physical measure of the target during operation, e.g. the power consumption over a certain time interval while performing an encryption. In a usual ML process as shown in Fig. 2, these raw data are preprocessed in a first step. For example, many ML algorithms require that the input data are normalized (meaning it is rescaled to values between 0 and 1) or standardized (having zero mean and unit variance). Then, data points with the highest information content are extracted (or constructed, e.g. by combining/creating additional data), a mechanism that we refer to as feature engineering. Next, a suitable algorithm needs to be selected for the given learning problem and its hyperparameters (the parameters that control the algorithm's behavior which are usually defined a priori) have to be adapted. Finally, the optimized model's performance is verified with 'unseen' data which was not used for training. A typical train/test-split assigns 60% of the data to the training set and 20% to the test set. The remaining 20% is

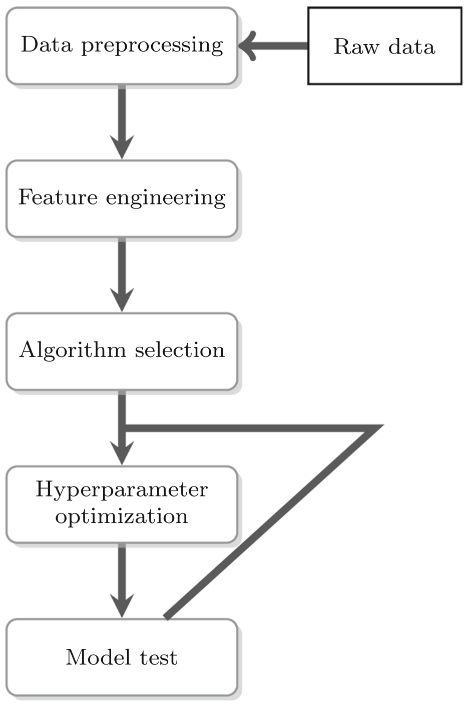

Fig. 2 Standard ML process

used for hyperparameter optimization. However, if the test set becomes too small, statistical uncertainty around the average test error can make comparability between different ML algorithms difficult [36]. That is why in practice often a procedure called *k-fold cross-validation* is employed. The idea is to split the whole data set randomly in k (the number of folds) disjunct subsets and iteratively use one of them as test set, while the rest of the data are used as the training set. The average error across all trails is the estimated generalization error (i.e. the expected test error when applying new data to the trained model). Cross-validation is also often used to estimate suitable hyperparameters.

Two common terms that are related to the performance of a ML model are *underfitting* and *overfitting* [36]. Underfitting occurs when the model is not able to obtain a sufficiently low error on training and test set. Overfitting means that the model performs well on the training set, but not on the test set. One way to control the degree of overfitting or underfitting of a model is by altering its *capacity* (i.e. the ability to fit to a wide range of functions), e.g. by increasing or decreasing the number of parameters.

The following subsections summarize the most important supervised and unsupervised learning algorithms used in SCAs. For better understanding, we have grouped them into three major categories:

– Dimensionality reduction
– Supervised learning

– Clustering

2.2 Dimensionality reduction

Dimensionality reduction techniques aim for decreasing the number of features in the input data set, while preserving as much information as possible. It can be done with both, unsupervised and supervised techniques in order to obtain a lower-dimensional representation of the original input. However, the loss in the input data may also have a negative effect if the removed information is crucial for prediction, although it is not important for the actual representation of input data [9].

2.2.1 Principal component analysis (PCA)

A popular approach for dimensionality reduction is principal component analysis (PCA). It maps each input data vector $\mathbf{x} = \{x_1, \dots, x_n\}$ onto a new vector $\mathbf{x}' = \{z_1, \dots, z_m\}$, where $m < n$. This is done by first calculating the mean vector $\boldsymbol{\mu}$ over all input features of the complete data set. Next, the $n \times n$-dimensional covariance matrix is computed and its eigenvectors and corresponding eigenvalues are found. The m eigenvectors with the largest eigenvalues are called principal components. In order to obtain the input data in its chosen principal components, a $n \times n$ matrix $\mathbf{A}$ is formed from all principal components. The transformed data are then derived by computing:

$$\mathbf{x}' = \mathbf{A}^T(\mathbf{x} - \boldsymbol{\mu}), \tag{1}$$

then skipping the points in output data $\mathbf{x}'$ corresponding to the $n - m$ smallest eigenvalues [26]. Side-channel observations usually deal with a high number of sample points (= features), making calculations computationally expensive. Because of that it is common to first find the sample points where the most leakage information is present, which are usually referred to as Points of Interest (POIs). PCA supports this by capturing the data with the largest variance and thus helping to reduce the amount of noise in the traces. That is why PCA is a heavily used technique in SCAs, even for settings that make otherwise no use of ML techniques (for example in [6]).

2.2.2 Linear discriminant analysis (LDA)

Linear discriminant analysis (LDA) is another dimensionality reduction technique that is applied in SCAs to reduce the number of sample points [15,121]. While PCA seeks for finding the features that maximize the variance of the input data, LDA additionally maintains the discriminatory information needed for separation between multiple classes [104]. LDA is thus considered a supervised method that can be used not only for data compression, but also for classification. The main difference in computation compared to PCA is the calculation of the within-class ($\mathbf{S}_\mathrm{W}$) and between-class ($\mathbf{S}_\mathrm{B}$) matrices which are defined as:

$$\mathbf{S}_\mathrm{W} = \sum_{i=1}^{c}\sum_{j=1}^{N_i}(\boldsymbol{x}_{i,j} - \boldsymbol{\mu}_i)(\boldsymbol{x}_{i,j} - \boldsymbol{\mu}_i)^T \tag{2}$$

$$\mathbf{S}_\mathrm{B} = \sum_{i=1}^{c} N_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T, \tag{3}$$

where c is the number of classes, $\boldsymbol{\mu}_i$ and N_i the sample mean and sizes of the respective class, and $\boldsymbol{\mu}$ the overall mean of samples from all classes. Afterward, the transformation matrix W that maximizes the separation between the classes is created by solving the eigenvalue problem $\mathbf{S}_\mathrm{W}^{-1}\mathbf{S}_\mathrm{B}$, before the updated data set $\mathbf{X}'$ is derived by the multiplication $\mathbf{XW}$. Bruneau et al. carried out a formal analysis of dimensionality reduction techniques in SCAs and showed that LDA is superior than PCA in many contexts [13].

2.3 Supervised learning

Many supervised leaning methods can be used in two modes: classification or regression. In a classification setting, the algorithm is asked to specify the class an input belongs to from a finite set of values. Sometimes not the class is given as output but rather a probability distribution over classes. Regression learning tasks, in contrast, deal with the prediction of quantitative outputs. We describe the different algorithms in the upcoming with a focus on classification-oriented usage, since the majority of work that is analyzed later aims for predicting discrete target labels (e.g. subkey bytes).

2.3.1 Support vector machine (SVM)

SVM is one of the most popular supervised learning algorithms [24]. It is also known as max-margin classifier, as it creates an optimal binary hyperplane between data points belonging to two linear separable classes y_1 and y_2 as shown in Fig. 3. The models prediction function is given by $\mathbf{w}^T\mathbf{x}+b$, where $\mathbf{w}$ is denoted as the weight vector and b the bias. For instances belonging to class y_1, it outputs a positive value. A negative prediction is outputted accordingly for instances belonging to class y_2. The hyperplane with the largest margin between the points of the two classes can be calculated by solving the optimization problem:

$$\begin{aligned}&\min_{\mathbf{w},b}\frac{1}{2}\|\mathbf{w}\|^2\\&\text{subject to: } y_i\ \ (\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1\dots N\end{aligned} \tag{4}$$

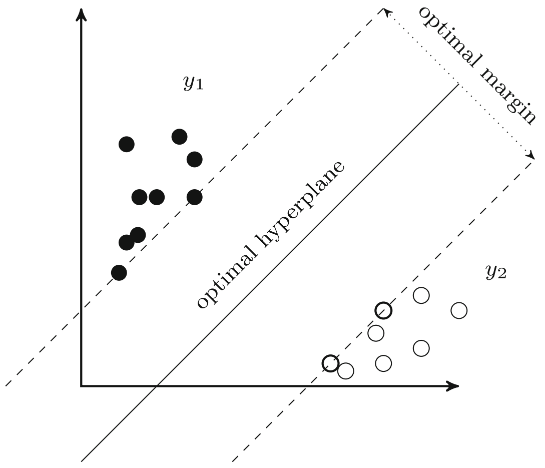

Fig. 3 Classification using a binary hyperplane SVM

The points that lie closest to the separating hyperplane are called support vectors.

Because almost all real-world problems deal with nonlinear separable data (i.e. data that cannot be separated by a linear decision boundary), the kernel trick was introduced. It enables the mapping of data onto a higher-dimensional space where the separation becomes possible again. The choice of the appropriate kernel function is crucial for the performance of a SVM, as it defines the transformed feature space where classification will be done [62]. The linear kernel function and the radial basis function (RBF) kernel were mostly used in SCA settings. A comprehensive description of kernel functions can be found in [118].

2.3.2 Decision trees (DTs)

DTs are a class of supervised learning algorithms that separate the training data by constructing a tree with zero or more internal nodes and at least one leaf node [88]. During training, the feature that best splits the training examples at each internal node is chosen. Branches coming out of these internal nodes are labeled with distinct outcomes of the corresponding feature, while leaf nodes are associated with output labels. New test instances are classified by passing it through the tree starting at the root node which is equivalent to a tree search. Figure 4 shows a simple DT performing a 3-bit AND operation on an input example $\mathbf{x} = (x_1, x_2, x_3)$.

A well-known algorithm for building a decision tree that was also used in SCAs is the *C4.5* algorithm [102]. It has been designed to overcome several issues that can occur when learning a tree, such as overfitting, or how to handle data with missing attribute values [86]. C4.5 chooses decision attributes to split the tree in further branches by using the gain ratio metric (i.e. the information gain that is achieved by a certain split whereby the size and number of branches are considered as well).

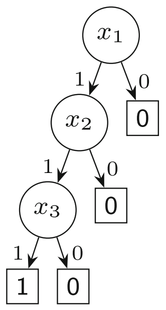

Fig. 4 Decision tree representing an AND operation with 3 input variables

2.3.3 Random forest (RF)

RF is an ensemble method that consists of a collection of tree-based classifiers which are individually trained with identical distributed random vectors of the training data set [10]. Each tree is furthermore constructed only with a random subset of available features (typically the square root of the total number of features or even only one feature [40]). Classification of test set examples after training is done by a majority voting among the grown trees. Thus, the procedure is relatively robust against outliers and noise and can easily be parallelized.

2.3.4 *k*-Nearest neighbors (kNN)

k-Nearest neighbors belong to the class of instance-based learning algorithms. Instead of creating a classifier for a certain target function during training, the provided input data set is just stored in a structured manner. Only when it comes to evaluation of new instances, a set of related examples is retrieved from the training database for classification of the new query instance. This procedure is therefore also referred to as lazy-learning method, since most of the computation is postponed to the classification phase [86]. kNN assumes that the examples in the training data set may be considered as points in the n-dimensional feature space. When an output label y should be found for a new input example $\mathbf{x}$, the k-nearest neighbor instances in the training set are discovered according to a certain distance metric (for example Manhattan or Euclidean distance). The resulting output class is then just the most frequent label among the k-nearest neighbors. The appropriate choice of the parameter k heavily affects the performance of the kNN algorithm. An automated, though computationally expensive, approach for determining the value for k is proposed by Guo et al. [39].

2.3.5 Neural networks (NNs) and deep learning (DL)

Neural networks were partly inspired by biological learning systems (e.g. the human brain) and date back at least into the

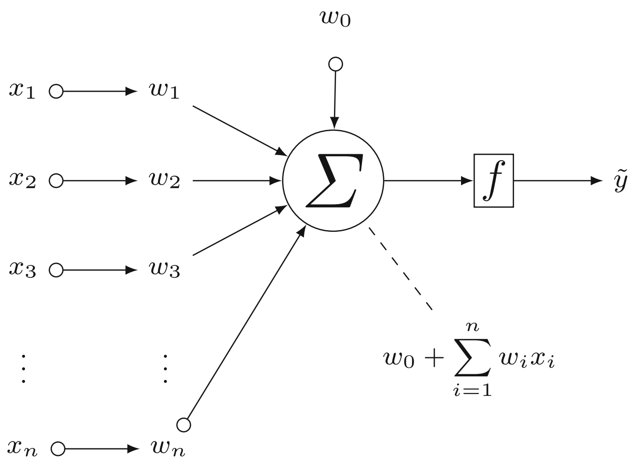

Fig. 5 Perceptron

1960s. They are composed of densely interconnected units called neurons, which take a number of real-valued inputs and produce a single real-valued output [86]. One type of a NN is the *perceptron*. As illustrated in Fig. 5, it receives a vector of input features $\mathbf{x} = (x_1, \ldots, x_n)$ and performs a linear combination with the weight values $w_1, \ldots, w_n$ of its input connections and a bias value w_0. The result is passed through a threshold activation function f (for example the hyperbolic tangent $f(x) = \tanh(x)$) in order to calculate the output value $\tilde{y}$. For learning the perceptron, the weights are adjusted according to the training data set.

Single-layer perceptrons are only able to represent functions whose underlying data set is linear separable such as the boolean AND function. To overcome this limitation and represent more complex mappings, many perceptrons can be stacked together to form a whole network which are generally referred to as multilayer perceptrons (MLPs). An MLP consists of three types of units, typically arranged in layers as shown in Fig. 6. The input layer is just a representation of the raw input features. All neurons of the input layer are connected to each neuron of the following hidden layer. The number of hidden layers in an MLP and the number of units per hidden layer varies, depending on the required model capacity to fit the training data. In general, too many units in the hidden layer may lead to overfitting, while underestimating the number of neurons has a negative effect on the classification performance of the MLP [36]. The units in the output layer directly correspond to the prediction classes of the problem to solve.

Training the MLP is a multi-step process involving the following steps:

1. Initialize the weights $w_{i,j}^{(k)}$ of the network to small random values, where (i, j) denotes the connection between the i-th node of layer k and the j-th node of layer $k+1$.
2. Take a training example $(\mathbf{x}, y)$ and compute an error function E (sometimes also called loss function), depicting the difference between the expected output y and the prediction result $\tilde{y}$ obtained from the MLP.
3. Compute by means of the *backpropagation* algorithm how the individual weights of network have contributed to the calculated error E, i.e. the gradient of the error function with respect to the weights:
$$\nabla E = \frac{\partial E}{\partial w_{i,j}}$$
4. Update the weights in order to minimize E using the gradient information and the learning rate parameter γ. The learning rate is a positive scalar that determines how fast the weights are driven toward the optimal solution:
$$\Delta w_{i,j} = -\gamma \frac{\partial E}{\partial w_{i,j}}$$
5. Repeat the steps 2–5 until some termination criteria are met. An iteration over the complete training data set is called an *epoch*.

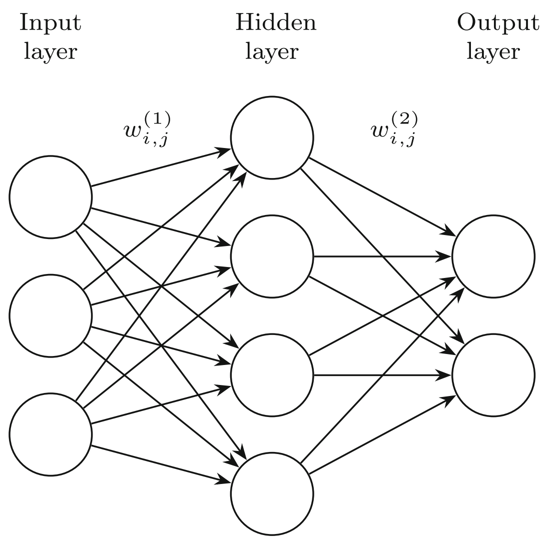

Fig. 6 Example of a simple MLP with 3 input units, 4 hidden units, 2 output units (bias units omitted)

The above sketched algorithm is called *stochastic gradient descent*. However, in practice one would rather use larger batch sizes (i.e. the number of training examples that are propagated through the network) and adaptive learning rates for updating the MLP's weight parameters (for instance the *Adam* algorithm [57]).

In recent years, there has been a growing interest in NN models with multiple hidden layers stacked upon each other, which is commonly referred as to Deep Learning (DL). It is a particular powerful type of ML techniques that are able to represent the learning task as a nested hierarchy of concepts, where more abstract concept representations are built

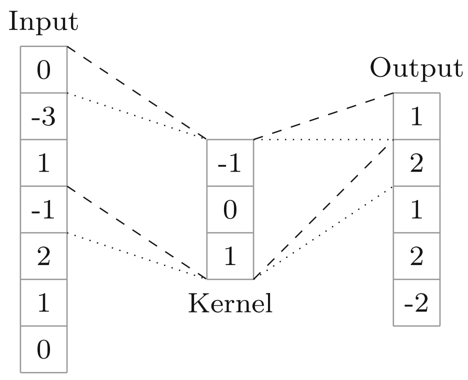

Fig. 7 Example of a 1-D convolution operation. The output is formed by applying the kernel to each part of the input (as with a sliding window)

from simpler ones. The usage of deep neural nets is motivated by the fact that they have succeeded in solving central problems in artificial intelligence such as speech recognition and image classification. These tasks usually deal with high-dimensional data which makes it exponentially more difficult to learn a classifier that generalizes well on unseen examples, a challenge that is also known as the curse of dimensionality [36].

CNNs tackle the challenge of large input data by including task-specific mechanisms into their architecture that allow to reduce the number of parameters of the model while keeping or even increasing the accuracy of the neural network [91]. CNNs are primarily used in the field of pattern recognition within images; however, they can also be used to process 1-D time-series data (as it is the case for side-channel traces). Additional to fully connected layers used in classical MLPs, CNNs include two other types of layers: convolution (CONV) layers and pooling layers. CONV layers determine the output of neurons which are connected to small spatial regions of the input by calculating the scalar product with a set of kernels or filters as illustrated in Fig. 7. The parameter weights of the kernels are learned to activate when they detect a specific feature or pattern at a certain position of the input during interference. Pooling layers perform downsampling of their given input in order to reduce the number of parameters and the computational complexity of the network, mostly by considering the maximum value (= max-pooling) of a certain spatial extent as the output. Common CNN architectures, e.g. the so-called VGG nets proposed for image classification [119] are composed of several Conv and pooling layers before one or more fully connected layers are connected on top.

2.4 Clustering

Cluster analysis techniques mostly belong to the group of unsupervised learning algorithms, meaning they work with unlabeled training examples only. The aim of clustering is to group related objects of the training data set into smaller subsets (clusters). Objects assigned to the same cluster are more similar to each other than objects belonging to different clusters. Similarity (or dissimilarity) between the objects is measured by calculating the pairwise distance of variable values (features) of the objects. This can be done on quantitative, ordinal, or categorical scale depending on the type of variable [40].

2.4.1 *k*-Means

A popular clustering algorithm is k-means. It divides the training set into k different clusters starting with an initial guess. Then, it iteratively identifies the closest cluster center (centroid) for each example in the data set and updates the centroids based on the mean μ_j of all training examples assigned to it until there is no change anymore. More formally, the goal is to find for a data set $\{\mathbf{x}_i\}_{i=1}^N$ the partitioning $\mathbf{c} = (c_1, \ldots, c_k)$ that minimizes the total cluster variance:

$$\min_{\mathbf{c}, \{\mu_j\}_1^k} \sum_{j=1}^{k} \sum_{\mathbf{x}_i \epsilon c_j} \|\mathbf{x}_i - \mu_j\|^2 \quad (5)$$

The squared Euclidean distance serves as similarity measure for determining the closeness of two training examples.

2.4.2 Hierarchical clustering

K-means requires the a priori knowledge of the number of clusters for separating the data set. However, sometimes one is rather interested in the relationship between different subsets of the input data. This can be accomplished by using *hierarchical clustering* techniques. These arrange the data set into a tree-like structure where the clusters at each level are built by merging clusters from the layer below. Leave nodes of the hierarchy represent a single data example. There are in general two types of strategies to create a hierarchical structure from the data set. *Agglomerative* methods work bottom-up and merge related clusters until a single root cluster was found. Closeness of clusters is calculated by group average dissimilarity or max/min distance of single pairs, that is, a threshold value has to be defined upfront. *Divisive* clustering algorithms, in contrast, start with the complete data set and recursively divide it into smaller clusters at each level of hierarchy. The splitting procedure remains active until for all clusters, zero dissimilarity exists among the members or each cluster consists of only one training example.

3 Power analysis attacks

This section introduces common techniques for a certain class of SCAs, the so-called power-based SCAs or power analysis attacks. SCAs belonging to this group exploit the fact that the instantaneous amount of power used by a device depends on the processed data and performed operations [79]. Due to this correlation, cryptographic secrets such as a symmetric key used in the advanced encryption standard (AES) block cipher are exposed when a device is performing a cryptographic operation. To measure a device's power consumption, one can insert a small resistor in series with the power or ground input (also called shunt). The voltage drop along the resistor divided by its resistance yields the power consumption [61]. A sequence of such power measurements sampled with an oscilloscope over a certain period is called a *trace*. Several techniques have been proposed for analyzing power traces in order to extract cryptographic secrets. Simple power analysis (SPA) is considered as the most straightforward approach as it attempts to directly interpret the power consumption variation in a trace to deduce information about a device's operation and used key material [58].

3.1 Differential/correlation power analysis (DPA/CPA)

In practice, SPA leaks are often hidden in noise or hard to interpret manually (as for example when targeting FPGA design which performs many operations in parallel). In this case, it is advantageous to apply more advanced analysis techniques based on statistical methods, namely differential power analysis (DPA) or correlation power analysis (CPA). Using DPA, the basic idea is to divide a set of traces into subsets according to a selection function which reflects the expected value of an internal intermediate operation O of the attacked cipher (typically a function that depends partially on the secret key and known plaintext or ciphertext). Next, the average power consumption of each subset is calculated and the pairwise difference is determined. Given that the selection of which trace belongs to each subset is uncorrelated with the measurements in the traces, the calculated differences will become small. However, for correct guesses the difference trace will show clearly visible spikes having enough power measurements [59]. In a CPA, the attacker first estimates the hypothetical power consumption for the targeted intermediate operation. Common power models are the Hamming Weight (HW), Hamming Distance (HD), and Zero Value (ZV) model [79]. Then, real power measurements of the cryptographic device are collected using the same known plaintext (or ciphertext) as for the hypothetical model. In a final step, the estimated power consumptions $\mathbf{M}$ are compared with the collected traces $\mathbf{T}$ using the correlation coefficient as distinguisher [11,61]:

$$\mathrm{Corr}(\mathbf{M}, \mathbf{T}) = \frac{\mu(\mathbf{M} \times \mathbf{T}) - \mu(\mathbf{M}) \times \mu(\mathbf{T})}{\sqrt{\sigma^2(\mathbf{M}) \times \sigma^2(\mathbf{T})}} \tag{6}$$

where μ denotes the mean and σ^2 the variance. Other side-channel distinguishers that have been proposed in the literature to compare key-dependent predictions of the physical leakages with actual measurements are mutual information analysis (MIA) [33] and Kolmogorov–Smirnov analysis (KSA) [134].

3.2 Profiling power analysis

CPA is most effective for settings where the power leakage model of the attacked device is known upfront. If no correlation can be found with any of the aforementioned models, one can try to build a customized leakage model using a second so-called *profiling* device. It is a copy of the attacked device which the adversary can manipulate to characterize the leakages very precisely with statistical techniques.

Template attacks [21] assume that the traces' leakage follows a multivariate Gaussian distribution which can be described by an according mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. They are built for each possible value o_i of the attacked operation O. The probability density function (PDF) for an n-multivariate leakage $\mathbf{l}$ is therefore estimated by the equation:

$$\mathrm{PDF}(\mathbf{l}|O = o_i) = \frac{\exp\left(-\frac{1}{2}(\mathbf{l} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{l} - \boldsymbol{\mu}_i)\right)}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}_i|}} \tag{7}$$

Once the leakage model has been characterized, the adversary acquires a new set of traces from the attacked device and computes those probabilities of originating from hypothesized operations $\tilde{O}$ using the templates. Finally, maximum likelihood principle is applied to combine the individual guesses and recover the secret key.

The Stochastic Approach (SA) of Schindler et al. provides another kind of profiled side-channel analysis tool [117]. Contrastingly to template attacks, SA describes the leakage function as sum of a data-dependent (and thus also key-dependent) part and a noise term which are approximated separately during profiling. The stochastic model assumes a linear relationship in the data-dependent part and neglects nonlinear dependencies. It was shown that in contexts where the leakage can be described by a monomial of small degree and the number of profiling traces is limited, the SA is more efficient than template attacks due to its simplicity. Furthermore, template attacks can be seen as a special case of the SA with maximal degree [34,66].

3.3 Electromagnetic side-channel attacks

Side-channel attacks due to electromagnetic (EM) emanations are also an active branch of research. In general, EM measurements can be analyzed using the same techniques

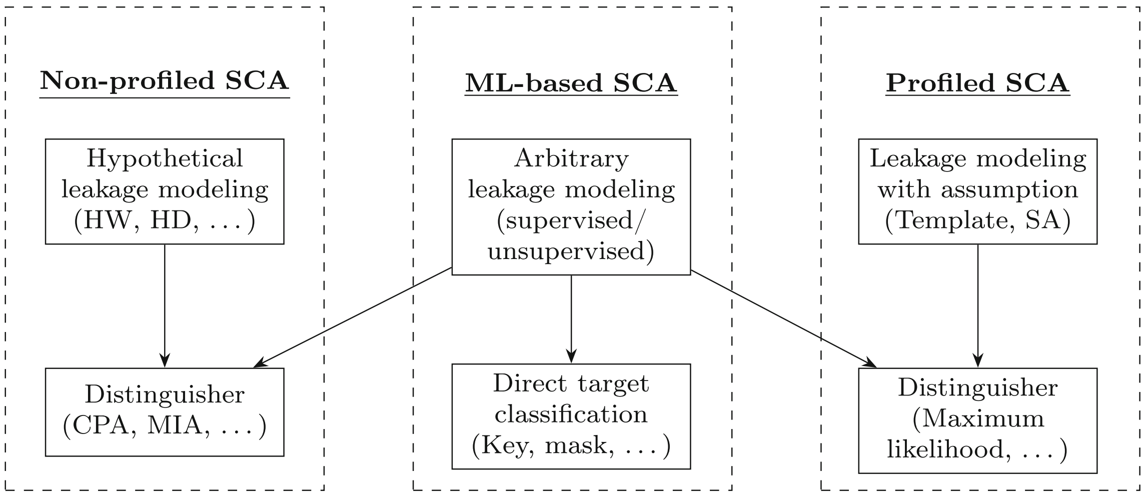
Non-profiled SCA
Hypothetical leakage modeling (HW, HD, . . .)
Distinguisher (CPA, MIA, . . .)
ML-based SCA
Arbitrary leakage modeling (supervised/ unsupervised)
Direct target classification (Key, mask, . . .)
Profiled SCA
Leakage modeling with assumption (Template, SA)
Distinguisher (Maximum likelihood, . . .)

Fig. 8 Relationship between traditional profiled/non-profiled SCAs and SCAs based on ML techniques

that apply to power measurements [61]. Simple electromagnetic attacks (SEMA) and differential electromagnetic attacks (DEMA) [30,103] are the common terms to refer to such kind of attacks. However, EM measurements are much more flexible and allow to capture (often more useful) information in very close proximity to the attacked chip even if power traces are available. EM allows, for example, to observe only the parts of a device's chip that have the highest side-channel leakage. In this paper, we consider work that is based on both, power and EM measurements, because of the strong relationship between them.

4 Attack vectors

Now that we have introduced ML and power-based SCAs in general, this chapter deals with the combination of both worlds. Figure 8 shows a high-level overview of how SCAs based on ML techniques can be fit into the common profiled/non-profiled SCA classification setting.

Whereas in non-profiled and profiled SCAs, the side-channel leakage is estimated according to a certain power model (respectively under specific assumptions about the leakage's probability distribution), most ML models are able to approximate any function of the leakage (under certain conditions which are explained later). During the attack phase, the ML models can either be embedded in a non-profiled (see Sect. 4.1.3) or profiled attack flow, or can be used directly to make hypotheses about the target values (e.g. individual key bits).

In the following subsections, we give an exhaustive overview on such approaches where ML methods were applied to extract confidential information of implemented cryptographic primitives. For each attack vector, we summarize the contributions from academia and compare them later with respect to different criteria such as preprocessing and feature selection methods.

4.1 Attacks on block cipher implementations

By far the most analyzed work considered symmetric block ciphers as a target. In order to highlight different possibilities for ML methods in this field and ease readability, we organized them into several subcategories. However, we stress that a clear distinction is not always possible and some approaches may fit into more than one class.

4.1.1 Recovery of intermediate cipher states

One of the first papers that deals with the application of ML techniques in SCAs of cryptographic implementations was presented by Hospodar et al. [48]. They used a variant of SVM called least square support vector machine (LS-SVM) to distinguish power traces of an unprotected software AES implementation regarding three properties of the S-Box output: HW smaller/larger than 4, even or odd HW, and the value of the fourth least significant bit. The most relevant features were discovered by a Sum Of Squared pairwise T-differences (SOST) analysis [34], Pearson correlation and PCA. They examined that the choice of the LS-SVM parameters significantly affects the performance of the classification, whereas the size of the training set is less important.

Heuser and Zohner were the first who used multi-class SVM classification to make assumptions about the HW of an intermediate value byte of an AES implementation running on an ATMega-256-1 microcontroller [43]. They show that their SVM attack is more suitable than the template attack for power traces with high noise level since it relaxes the assumption that the data underlie a multivariate Gaussian distribution. This provided the basis for the work of Bartkewitz and Lemke-Rust one year later, who designed probabilistic multi-class SVMs the same way as being done in template attacks [5]. For finding the POIs, they applied a technique called *normal-based feature selection*. Here, the absolute values of the weight vector **w** determine if a corresponding feature has significant influence on the classification performance or not. Weight values with small absolute value are therefore just set to zero to disregard unimportant features. The efficiency of the approach was measured according to the so-called *Key Guessing Entropy* (KGE), a technique which quantifies the difficulty to retrieve the correct value of a key regarding the required number of traces [123]. They observed that the linear kernel does not perform well in SVM-based template attacks since it implies a linear classification problem, whereas the RBF kernel is appropriate for non-linear problems.

Banciu et al. investigated several classifiers in the context of single trace attacks [3]. These type of attacks assume an adversary which only has access to a single attack trace. When targeting symmetric ciphers, the attacks should be error tolerant in a sense that side-channel leakage information for an intermediate value can be a set of possible values. Examples from literature are pragmatic SPA [78], which tolerates a set of five HW guesses, whereas algebraic SCAs [105] are restricted to three possible HW values. In the study, templates, SVM, kNN, DT and RF were considered to output a ranked list of HWs given power consumption traces obtained from an AES implementation running on two experimental platforms: An 8-bit microcontroller and an ARM7 microprocessor. Only Gaussian template, RF and SVM performed well across the two data sets and different numbers of traces/features used for training.

Picek et al. studied in detail the effectiveness of Bayes classifiers compared to template-based attacks [95]. In addition to the Naïve Bayes approach which considers the features as conditionally independent of each other regarding the classification, the averaged one-dependence estimators (A1DE) strategy was investigated as well. It relaxes this strong assumption by making all attributes independent given the class except one privileged attribute called the super-parent [132]. between 5000 and 100,000 measurements from two public data sets (DPA contest v2 and v4 [126,127]) were randomly selected in order to predict the outputs of the AES S-Box (respectively, the HW of the S-Box outputs) using a train/test ratio of 2:1. Two additional evaluation metrics besides classification accuracy were used to report the result: area under ROC curve (AUC) and F-measure. ROC analysis plots the true-positive rate against the false-positive rate, and F-measure effectively references the true positives to the arithmetic mean of predicted positives and real positives [100]. In most scenarios, A1DE achieved a higher accuracy than template attacks and an improved version with pooled covariance matrix [23]. The authors therefore suggested A1DE as an alternative when the profiling base is small and other ML techniques such as SVM and RF are too costly to tune and perform.

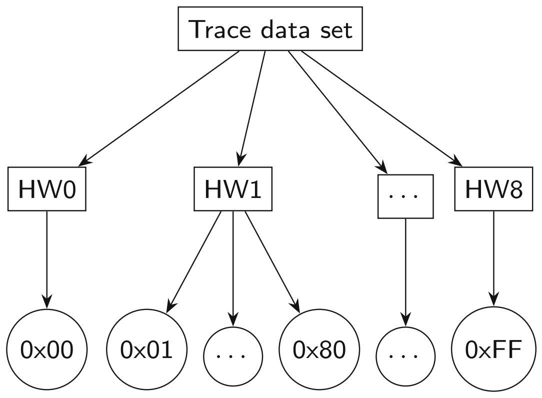

Fig. 9 Hierarchical trace classification using HW as intermediate nodes [97]

In a further contribution, Picek et al. presented an approach called hierarchical classification [97]. The idea is to explore the natural clustering of the leakage in order to arrange the class variables (i.e. sensitive target values) in a tree structure. Figure 9 shows their attack methodology: First, the attack traces are divided in the corresponding HW of the sensitive variable (e.g. output of AES S-Box) and then each subset is classified into the actual value of the sensitive variable itself. Naïve Bayes, C4.5, rotation forest [106] and SVM were considered as classification algorithms and evaluated along with standard template attacks. In most cases of the conducted experiments, hierarchical SVM performed best. As an extension, the authors proposed to combine the hierarchical approach with a standard flat classification to increase the accuracy.

Finally, Picek et al. showed in an additional study the importance of proper parameter tuning when using (parameterizable) ML techniques for side-channel analysis [99]. From the set of examined supervised classifiers (SVM, rotation forest, RF and MultiBoost [29]), the best results (in terms of classification accuracy using tenfold cross-validation) through parameter tuning were obtained for SVM. However, rotation forest and MultiBoost performed only slightly worse with their optimal settings, but were far more robust to parameter value changes. It is furthermore shown that a carefully tuned algorithm is able to reach a relatively high accuracy (more than 70% when having low noise) even if only a small number of relevant features is used (here 20%). Selection of the features was done with the *information gain* method, which is related to the C4.5 used to grow DTs (see Sect. 2). The authors also presented a novel side-channel metric called *Data Confusion Factor* that quantifies the difficulty of a given ML problem regarding a certain data set.

Maghrebi et al. were the first who applied DL in the context of side-channel analysis [76]. Apart from CNNs which were introduced in Sect. 2, the authors investigated stacked auto-encoders [85] as well as Long and Short Term Memory (LSTM) [46] from the field of DL techniques. A common factor that motivates the usage DL in general is that they intrinsically incorporate feature extraction mechanism. That is, unlike most standard ML classifiers, deep NNs can learn from the raw input data set as they are able to identify the most informative points themselves without human engineering. In order to check if this holds also for SCAs on cryptographic algorithms, they performed a series of experiments using the aforementioned DL techniques, classic ML classifiers (SVM, RF, MLP) and templates to attack unprotected and protected hardware and software implementations of AES (using data sets from DPA contest). Summarizing the results, one can say that the DL methods mostly outperformed the other attack techniques. Especially for the protected software, implementation was no prior mask profiling necessary to break the key compared to the attacks described in several other publications (see Sect. 4.1.4). Interestingly, the impact of PCA as preprocessing step in combination with MLP did not enhance the efficiency, a finding that stays in contrast to most related work (e.g. [35]).

4.1.2 Direct (sub-)key recovery

Lerman et al. [65] showed the applicability of supervised ML to attack individual key bits of an FPGA-based 3DES implementation. They considered Self Organizing Maps (SOMs), SVMs and RFs as prediction models and combined them with different dimension reduction techniques (PCA, minimum Redundancy Maximum Relevance (mRMR), Ranking), whereas the combination of RF with PCA performed best. A brute-force strategy for key bits with uncertain predictions is suggested in order to enhance the attack. Additionally, a comparison with a standard template-based power attack was performed and evaluated using the KGE metric.

Liu et al. also focused on multi-class SVM as a distinguisher in their work published in [74], but exploited EM as side-channel instead of power consumption. For reducing the high amount of sample points in the traces, they first determined the 4000 most relevant components using SOST analysis and then applied a PCA. This setting was used to determine the state of a 6-bit subkey of an unprotected DES implementation.

Another SVM-based attack on DES was presented by He et al. resulting from a semester project of an ML course at Stanford University in [41]. They used standard SPA and DPA to determine the exact location of the first and second key permutation and the first XOR operation round function in the power traces. For each region, a separate classifier was trained, whereas more sophisticated feature engineering also including the ciphertext was needed for the XOR classifier in order to get an accuracy greater than 80%.

Martinasek and Zeman proposed a three-layer MLP to determine the first AES key byte from power traces obtained from a smart card microcontroller [138]. Their method was able to identify the correct value in around 85% of the cases. By calculating the average power trace and subtraction of measured traces from this average trace, they were able to further improve their method [81]. In [82], a comparison of their MLP techniques with templates is presented using the same data set and an adapted version of SOST [12] for finding the POIs. It turned out that the optimized MLP version was almost equally efficient as a template-based power attack. Second and third experiments based on the public data set of DPA contest v4 were presented in [83]. Here, the main outcome was that if the adversary has only a limited number of power traces and POIs available, MLP-based attacks are more effective. However, when using template attacks based on a pooled covariance matrix and a larger learning data set, the results were practically the same compared to the developed MLP architecture.

4.1.3 Leakage modeling

Side-channel leakage modeling based upon NNs was studied for the first time by Yang et al. [136]. They motivate its usage by the fact that NNs are able to capture nonlinear power leakages without specific restrictions, as compared to other nonlinear models [47,75]. These generally assume that the leakage behavior (respectively the power consumption) of individual bits of a sensitive operation is independent of each other. The NN is trained to predict the hypothetical leakage (real value) given an intermediate value of the target cryptographic algorithm, meaning it is not used to directly recover the secret key as in a classification setting. Instead, it is combined with CPA and MIA where it replaces standard leakage models such as the HW. Effectiveness of their attack constructions was reported with a series of experiments including different levels of noise and different evaluation metrics (KGE, success rate and distinctive level [50]).

Lerman et al. [64] incorporated methods from time-series modeling into multi-class profiling attacks based on SVM and RF. The rationale is to explore potential information available in temporal dependencies between individual values of a power trace. This can also be seen as sort of dimensionality transformation of the input data to a space with lower variance, making the approach more robust against noise. They furthermore evaluated other feature selection techniques (MAX function, mRMR, SOST) with respect to different levels of noise showing the advantage of their method. The experiments were carried out on power leakages from the DPA contest v1 (unprotected ASIC DES) [125].

Jap et al. [52] presented a method for leakage modeling called Support Vector Regression (SVR). As the name implies, SVR is based on support vectors like SVMs, but uses regression instead of classification. The procedure is quite similar to the SA of Schindler. The attacker first acquires two sets of traces from the reference device: one set for model building and the second one for noise covariance estimation (profiling). Then, the attacker measures new traces from the target device and determines the secret key by using maximum likelihood and the built profile. However, the advantage of SVR is the kernel method, allowing it to capture non-linear dependencies, whereas the method of Schindler et al. [117] is based on linear regression. The quality of SVR was compared with SA and HW when used as leakage model in a CPA against an AVR microcontroller running an AES implementation. It could be shown that SVR reaches almost the same preciseness of a complex SA model with 256 dimensions.

Whitnall and Oswald presented a general strategy to integrate unsupervised clustering methods into a profiled DPA-style attack at CHES 2015 [133]. The intent was to extract a nominal power model (i.e. a power model that does not need to be perfect) during a profiling phase and use it in the subsequent attack phase to hypothetically map new traces to classes for each potential key. The guess that is associated with the most meaningful pattern is probably the secret key. They evaluated k-means and hierarchical clustering in combination with several partition-based DPA distinguishers to attack software and hardware implementations of AES [122]. The strength of their method is the robustness to natural distortions, meaning it does not require identical measurement setups or even identical preprocessing steps for the profiling and attack traces. This is especially an advantage over gaussian templates which are not effective in non-ideal attack scenarios.

4.1.4 Mask recovery

Zeng et al. [139] presented a successful attack against an AES software implementation which is protected with a lightweight countermeasure called Rotating S-Box Masking (RSM). Since the mask values are fixed in this countermeasure, the critical part is to break the 4-bit random offset value that denotes the starting index [89]. Therefore, they first trained a separate SVM classifiers to discover the secret mask offset and then attacked every S-Box using the HW and bit power models [79] to recover all subkey bytes. In order to determine the optimal number of profiling traces and POIs, they conducted a series of 60 experiments.

A similar attack strategy against RSM was proposed by Lerman et al. [67,69]. Their approach applies a profiled attack to extract the mask values considering RF, SVM, templates, SA and multivariate regression analysis [124] followed by a non-profiled (using CPA) or profiled step that retrieves the

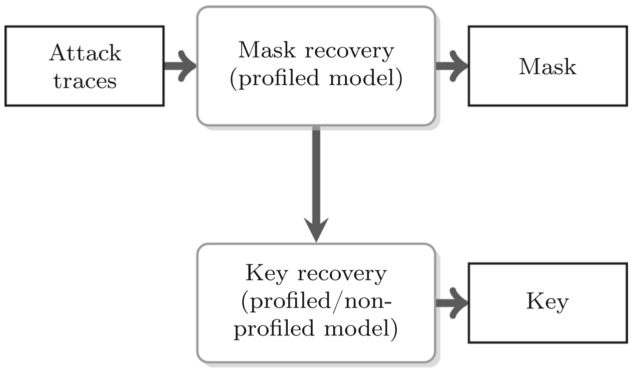
Attack traces
Mask recovery (profiled model)
Mask
Key recovery (profiled/non-profiled model)
Key

Fig. 10 Attack strategy against masked implementation by Lerman et al. [67]

secret key as illustrated in Fig. 10. Since in their setting a large number of traces followed an unknown PDF that was not gaussian (determined by Shapiro–Wilk test [2]), the ML models were able to extract more information when analyzing the same data set (with respect to template attacks). Regarding SA, the main advantage of the ML approach is reduced execution time (four times less).

Based on the previously mentioned work, Gilmore et al. [35] performed an attack on the same target. However, instead of training a SVM to recover the mask value and a non-profiled attack to recover the secret key, they proposed a strategy solely based on MLPs. Initially, a first MLP is built to determine the mask value and afterward a second MLP is trained with the same traces and the knowledge of the mask value to attack bytes of the key. By using cross-validation, they found out that a single hidden layer with the number of neurons equal the number of input features (= number of sample points in the traces) is sufficient for the aforementioned tasks. In contrast to other approaches, they employ PCA not for finding a small amount of features that carry the most information, but as preprocessing technique to discard sample points that contribute less than 2% to the total signal variance.

Martinasek et al. [80] used gaussian templates and MLPs to attack an improved version of RSM that combines masking and shuffling techniques (denoted as DPA Contest v4.2 [7]). The templates served for revealing the secret offset values in a first step, while the MLP from an earlier paper was used in order to retrieve the output of the S-Box. By doing so, they were able to discover the whole 16-byte AES key needing only 13 power traces in the attack phase.

In another contribution, Martinasek et al. [84] provided an extensive comparison between a range of ML algorithms and different template attacks. They implemented a verification program that automatically searches for the optimal hyperparameter setting of each classifier that achieves the best classification accuracy for a given dataset. For that purpose, tenfold cross-validation was performed in a loop over a range of possible parameter combinations. The selection of parameters was done based on experience of the authors gained from previous studies. The overall process of the testing program and the considered algorithms are described in Algorithm 1. Summarizing the obtained results for three different data sets, the authors concluded that every ML algorithm can be optimized to get almost the same classification results (all tested algorithms achieved a success rate between 84 and 95%). However, the time required for finding the best hyperparameter setting for a certain algorithm varied heavily. For example, it took approximately eight days until the best parameters of the SVM were returned by the verification program, whereas optimizing kNN terminated after only 6 minutes and 35 seconds. This is plausible since kNN only has a single parameter to choose and does not include a learning phase (see Sect. 2). Therefore, kNN is advertised as an alternative tool for profiled SCAs.

Algorithm 1 Optimize Hyperparameters

```
Input: $\mathbf{X}$, the input data set
Input: $M$, a ML classifier of set {SVM, DT, MLP, kNN, RF, LDA}
Input: $\mathbf{p}$, the parameters to test
Output: $\mathbf{p}_{opt}$, the optimal parameter combination
Output: $AC_{opt}$, the classification accuracy obtained by using $\mathbf{p}_{opt}$
  for all parameter combinations in p do
    $\mathbf{p}_{sel} \leftarrow$ select_parameters($\mathbf{p}$)
    $AC_{sel} \leftarrow$ tenfold_cross_validation($\mathbf{X}$, $M$, $\mathbf{p}_{sel}$)
    if $AC_{sel} > AC_{opt}$ then
      $AC_{opt} \leftarrow AC_{sel}$
      $\mathbf{p}_{opt} \leftarrow \mathbf{p}_{sel}$
    end if
  end for
```

Hou et al. [49] explored the effect of wavelet analysis when used as a kernel function in an SVM-based SCA. According to the authors, the wavelet kernel is able to approximate almost any function in continuous space and therefore improves generalization of the SVM compared to traditional kernel functions such as Gaussian. They conducted an exhaustive analysis using different kernel and wavelet functions for attacking protected and unprotected AES implementations that substantiate their assumption. Also the required time effort for training could be reduced by 40% through optimal choice of SVM parameters.

4.1.5 Simulated settings

An approach that includes probabilistic NNs and discrete wavelet transform as preprocessing tool was presented by Saravanan et al. [114]. Wavelet transform is a special form of Fourier transform that correlates the original signal with a set of template functions obtained from scaling and shifting a wavelet function in order to decompose the original signal into several components with different frequency bands. The power traces that were processed by the wavelet transform were collected by varying the atmospheric temperature from 27 to 70 °C within a Cadence Spectre simulation of the AES S-Box. They were able to successfully determine the correct used key out of four random keys with a single attack trace. A more comprehensive evaluation of the approach including other wavelet families was shown in [113].

A more theoretical contribution of Lerman et al. systematically investigated the influence of the number of dimensions (i.e. the number of sample points considered) on two types of profiled SCAs: ML-based and template attacks [71]. Using a formal proof, they showed that ML attacks are less beneficial compared to template attacks in case of a perfect profiling (i.e. when the probability density function of the leakage is perfectly modeled). However, a perfect profiling never occurs in practice due to noise and interdependencies between individual sample points. Therefore, they created a simulated environment where they could vary the number of informative and useless dimensions in order to examine the effect of imperfect profiling. They observed that ML-based attacks (here SVM and RF) become more interesting when the number of useless samples in the traces is high and the training data set is limited. The main conclusion of the authors was thus that template-based attacks are suitable for well understood devices, whereas ML-based attacks are more promising in black-box settings.

Heuser et al. [42] investigated lightweight block ciphers regarding side-channel resistance in profiled and non-profiled scenarios. In the case of non-profiled attacks, they evaluated several ciphers with 4-bit and 8-bit S-Boxes using the confusion coefficient metric [129] and software simulations. The authors could not examine that 4-bit S-Boxes are generally weaker than the considered 8-bit S-Boxes. For the profiled experiments, Naïve Bayes, C4.5, and MLP were used from the family of supervised ML algorithms to attack PRESENT and AES simulations. Their study showed that a single feature (with sufficient information) may be enough for mounting a successful attack. However, it was also shown that the difference between AES and PRESENT in terms of side-channel resilience is rather low.

4.1.6 Attacking unlabeled traces

An unsupervised, regression-based attack methodology was proposed by Chou et al. [22]. As laid out by the authors, unsupervised attack settings generally require no reference device for generating training data (i.e. by feeding different plaintexts and keys into the reference device and recording associated power traces). Their framework was furthermore able to consider information from multiple DES decryption rounds in order to deal with situations of only limited number of traces.

Between the previous method and the other aforementioned work lies the semi-supervised template approach introduced in [70]. Here, an attacker is able to collect two sets of traces from the same device: The first set using several known keys (for example from different users having different keys) and the second set with fixed unknown key (for example when the attacker manipulates the device). By using the clustering technique partitioning around medoids (PAM) [128], they were able to discover the HW of a byte of the attacked symmetric cryptographic key.

Another semi-supervised SCA based on collaborative learning was proposed by Liu et al. [73]. In this attack setting that required only a small set of labeled traces, two different SVM classifiers (RBF and linear) were used to predict the class of unlabeled traces with corresponding class probability. If the result of the two SVM classifiers was consistent and the class probabilities were within a certain range, the selected training example was added to the labeled training set and a re-training was performed. This cycled until a certain termination criteria was met (five iterations). Compared to the single SVM approach [67] which used the same data set from DPA contest v4, the presented collaborative model was superior when dealing with the same number of labeled traces. This is due to the fact that it is able to make efficient use of the information of unlabeled samples as well. However, an additional CPA was needed in order to reveal the mask values in a first step.

4.1.7 Attacking misaligned traces

The contribution of Lerman et al. [68] aimed to clarify which profiled SCAs have the lowest sensitivity to signal modifications in real-world settings. They considered several ML classifiers (SVM, RF, MLP) as well as standard and extended template attacks based on a pooled covariance matrix [23]. The conducted experiments were grouped in four scenarios: invalid traces (traces which are associated with a wrong label), misaligned traces, increased noise, and a different DC offset in the profiling and attack traces. For the first scenario, it was shown that the ML models mostly outperform template attacks in case of a high number of wrong traces. This is consistent with the outcome of numerous other studies (e.g. [71,83]) showing that ML techniques are more advantageous when the training set is small. In case of misaligned traces in the attacking set, the five models performed similarly (success rate decreases linearly with growing misalignment). However, ML models performed better when the profiling set also contains misaligned traces. By contrast, pooled templates outperformed all other models or had similar results for noisy traces and traces with varying DC offsets between the profiling and attack set (at least when considering traces from the DPA contest v4.2). This led the authors to conclude that there is no best model for each scenario, an observation that is generally referred to as 'no free lunch' theorem [135].

Cagli et al. investigated the applicability of CNNs to break cryptographic implementations which are protected with jitter-based countermeasures [16]. This type of protection mechanisms creates misalignments in the side-channel traces for example by insertion of random delays through dummy operation or by generating an unstable clock signal. To defeat such countermeasures, the authors proposed the use of a CNN in combination with data augmentation techniques. These are commonly used as a regularization operation when training CNNs for image recognition tasks in order to increase the training set size by adding extra copies of examples which are modified by label invariant modifications. In the context of side-channel analysis, this means the traces are distorted artificially in order to simulate a clock jitter effect that does not influence the target label information. To this end, *shifting deformations* (simulates random delay) and *add-remove deformations* (simulates clock jitter) were applied to the traces of different training-sets. They created a CNN architecture encompassing more than ten layers and trained it to classify the HW of an S-Box lookup of an AES software implementation. Two case-studies were presented, one regarding software-based and the other one regarding hardware-based countermeasures. The results showed that the data augmentation technique heavily helps to defeat overfitting of the CNN and reaching a validation accuracy of around 80% although only a limited amount of training traces were used in each setting. However, the performance on a secured hardware AES smartcard implementation was rather marginal compared to a gaussian template attack with manual realignment preprocessing.

Prouff et al. [101] conducted a comprehensive study of MLPs, CNNs and template attacks based on a public database named ASCAD provided by the same authors. ASCAD aims for establishing a common framework to evaluate and compare different ML models in the context of side-channel analysis. It consists of three sets of SCA traces and associated metadata of a first-order secured software AES implementation which exhibit different levels of jitter, along with a number of python scripts to set up and parameterize the database. For finding the optimal network configuration for the MLP and CNN attacks, the authors followed a two-stage selection process: first, the training hyperparameters (e.g. learning rate) were tuned, followed by an evaluation of the impact of different architecture-related design choices (number of layers etc.). A comparison of the best models elaborated by this method showed almost similar on perfectly synchronized traces, while the CNN outperforms the other models in the presence of de-synchronization (confirming the results of Cagli et al. [16]).

Shortly after the publication of the ASCAD data set at the beginning of 2018, Timon exploited the database for the first non-profiled SCA based on deep learning [130]. The idea is to create hypothetical intermediate values over all key hypothesis as in a standard CPA in a first step. Then, for each key guess, a deep learning training is performed using the calculated intermediates as labels. Given the circumstance that the key is guessed correctly, the training metrics (accuracy, loss) should be significant better than for trials where the key is wrong. It could be shown that the proposed method is more efficient than a CPA in case of de-synchronized traces and is also able to reveal the correct key from first- and second-order protected implementations. Furthermore, it is demonstrated that data augmentation for CNNs as suggested in [16] can also be applied in combination with MLPs to enhance the attack performance.

4.2 Attacks on stream cipher implementations

Chakraborty et al. suggested a power-based SCA on the stream cipher Grain v1 using LS-SVM [19]. They built a power consumption model for the initialization phase based on previous work about power analysis of Fibonacci Linear/Nonlinear Feedback Shift Registers [14,137] in order to attack the 80-bit key. The LS-SVM served for identifying the HD between two successive clock cycles which is needed for the proposed strategy. The expected number of Initialization Vectors (IVs) to recover the whole key was less than 100. The same authors combined their SCA with a fault analysis scheme to overcome fault attack countermeasures developed for the Grain stream cipher family [18]. An external function generator was used to introduce clock glitches that may cause single bit faults. The idea of the attack is to first perform the SCA during the initialization of the cipher and thereafter combine it with the single bit error model to retrieve the secret key.

An EM SCA based on multi-class directed acyclic graph SVM (DAG-SVM) was proposed by Duan et al. [25]. A DAG-SVM for a k-class classification problem was created of $k(k-1)/2$ binary SVMs that were arranged in a rooted binary tree as shown in Fig. 11. At inference time, the tree is traversed top down whereby each binary SVM is tested against each other. In order to find the optimal hyper- and training parameter for the SVMs, the authors conducted an automatic tuning technique using Particle Swarm Optimization (PSO). This is a parallel evolutionary algorithm inspired by social behavior of birds looking for food [27]. Cross-validation result of the SVMs was used as fitness function. They accomplished a 100% key HW classification rate when attacking a RC4 software implementation with the best parameter solution, beating several other ML algorithms such as k-means and probabilistic NNs.

Results of a power analysis attack against the stream cipher MICKEY-128 2.0 were reported by Chakraborty et al. [20]. The authors used PSO to generate specially crafted IVs

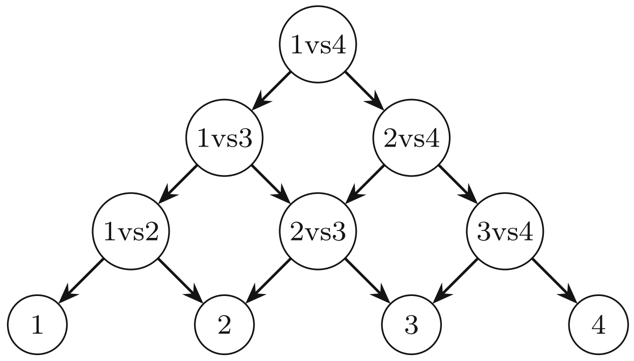

Fig. 11 Example of a DAG with four binary SVMs [25]

which allowed them to built power templates based on the HD model for individual key bits. These were the training base for an LS-SVM classifier. Additionally, they showed that PSO generated IVs can reduce the amount of power traces needed for a standard CPA attack from 3000 to 500.

Zhang et al. [140] also used wavelet transform for their attack on a software implementation of RC4. They analyzed the impact of four kind of signals on the success rate: The original signal, reconstructed signal, and high- and low-frequency signals. PCA was used to lower the dimension of the individual signals and a SVM served as classifier. In the range of 100–800 dimensions (features), the best classification result of the four signals was that of the reconstructed signal followed by the low-frequency signal.

4.3 Attacks on asymmetric cipher implementations

Published work in this area can mainly be divided in single trace attacks (where the adversary is able to use only the leakage of a single execution) and attacks which includes a profiling step.

4.3.1 Single trace attacks

Heyszl et al. utilized k-means clustering to attack an FPGA-based Elliptic Curve Cryptography (ECC) point multiplication to recover the secret scalar [44]. Since the secret scalar changes in every execution, they proposed to perform simultaneous EM measurements at different positions on the die and combine them in order to gather more leakage information and lower the remaining brute-force complexity. Later, Specht et al. [120] improved the attack by using PCA as dimensionality reduction and feature selection technique, discarding the highest-ranked as well as many low-ranked components. However, their setting based on expectation–maximization clustering did not gain significant benefit from the combination of different channel measurements.

A single trace attack against an FPGA implementation of the Rivest–Shamir–Adleman (RSA) algorithm protected with leakage resistant arithmetic and exponent blinding was presented by Perin et al. [94]. Here, k-means was used in the search for the POIs in the traces and fuzzy k-means was employed in the cluster classification. Furthermore, they combined the cluster classifications of several POIs in the traces to recover the bits of the secret exponent. Additional hardware countermeasures that reduce the signal-to-noise ratio (SNR) such as time disarrangement or dummy cycles were proposed to avoid memory access related leakage of information due to conditional tests in the algorithm (for example in practical implementations of the Montgomery ladder).

Järvinen and Balasch [53] investigated single trace attacks in the context of ECC algorithms with precomputations. Precomputations are generally used to speed up the scalar multiplication operation and are computed from the base point P. The authors focused on a certain SPA-resistant width-w non-adjacent form (w-NAF) scalar multiplication algorithm [90], but the attack also applies to other scalar multiplication algorithms which use precomputation methods. Two methodologies were presented, one based on correlation and the other one on k-means clustering, whereas the latter is, in general, weaker but requires less knowledge about the attacked implementation. Feasibility of the approaches was demonstrated by Matlab simulations and real experiments using an 8-bit AVR microcontroller. The results showed that software implementations of that particular type of scalar ECC multiplication which use small word sizes (8/16-bit) are potentially at risk.

4.3.2 Profiled attacks

Saeedi and Kong investigated the influence of dimensionality reduction with PCA regarding classification accuracy and time complexity when targeting an ECC point multiplication [111]. They found that roughly 600 components out of 2500 sample points are enough to achieve a maximum accuracy of approximately 95% when using SVM with RBF or polynomial kernel. For less components, the RBF kernel performed better. Later, they verified several other kernel functions and the influence of corresponding function parameters on multi-class SVMs [109].

Furthermore, Saeedi et al. presented a SCA against an ECC FPGA implementation based on a cascade-forward backpropagation (CFBP) neural network [110]. In contrast to the static architecture of MLPs, CFBP is able to dynamically adjust the number of neurons in the hidden layer(s). Starting with a network that only consists of the input layer and one output neuron, new neurons are trained and added to the network one by one during training based on the input data [116]. Each new neuron is connected to all input units as well as to all unit of previous hidden layers, forming an architecture as illustrated in Fig. 12. The authors performed an exhaustive

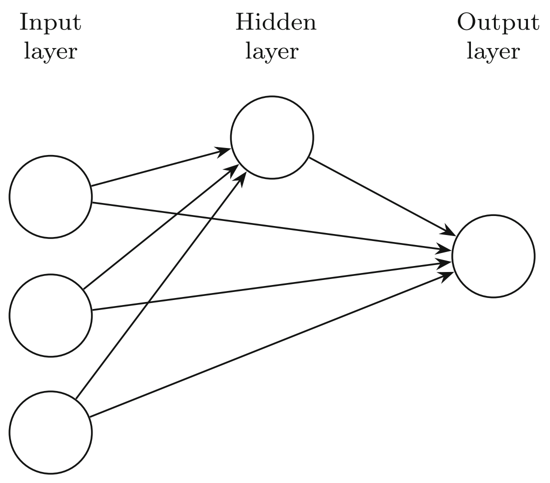

Fig. 12 Example CFBP architecture [110]. The network is built dynamically during training by adding new neurons and connecting them with the neurons of previous layers

investigation where the CFBP was trained with 12 different training algorithms and a varying number of hidden layers. It turned out that the *Levenberg–Marquardt* training algorithm and a CFBP network architecture with a number of hidden layers between 20 and 30 were the most efficient choice for the considered data set. They achieved a classification accuracy of around 75% regarding four different bits of the secret scalar.

Later, the authors applied a learning vector quantization (LVQ) neural network on the same target [112]. LVQ is a special learning algorithm that combines competitive learning with supervision. The neurons of a competitive layer create a prototype vector during training, by calculating the distance between the input vectors and the prototype vector. These directly define (nonlinear) class boundaries. Since the NN architecture (number of hidden layers, number of neurons per layer) plays an important role when using LVQ as well, an experimental study with different numbers of hidden layers between 10 and 110 was performed. Architectures with 90 to 100 hidden layers showed the lowest training error. Classification results were reported using a confusion matrix regarding four different key bits, where an overall accuracy of 86% was shown.

The approach of Özgen et al. [93] took advantage of supervised classification methods as well in order to reveal private key bits of asymmetric cryptographic implementations. Here, the target was the *binary left-to-right double-and-always-add* algorithm [55], a regular and thus (in theory) leakage resilient version of an elliptic curve scalar multiplication implemented in the open source library mbedTLS. Naïve Bayes classification, kNN, and SVM were trained to built so-called online templates for individual key bits using the multiplication pattern contained in EM traces (time range where the critical operation is performed). They are called online since the templates are created after the acquisition of a trace from the target device. The proposed attack achieved 100% success rate for each classification method if more than 20 template traces (obtained from the training device) were used for a scalar bit.

Chakraborty et al. demonstrated a successful attack on a software-based RSA implementation [17]. This was protected with Joey's ladder [56], an algorithm that secures the modular exponentiation operation in RSA from unintentional information leakage through side-channels. Although the operations in the algorithm have no explicit dependency on the secret exponent, a registers' update sequence is different for individual exponent bits which can be monitored in associated power traces. Therefore, they built two classes of power templates by suitable selection of plaintext messages and known key values in the profiling phase. The plaintexts were generated by PSO in a way such that the HD between the targeted register update differs substantially for the key bit guesses in the same time window. As in the aforementioned attacks by Chakraborty, LS-SVM was used as distinguisher. The results showed a classification rate of around 82% when the three highest SNR points were selected as features. The authors also discussed threats due to possible cache timing side-channels in the Joye ladder and proposed a suitable countermeasure.

4.4 Attacks on hash functions and physical unclonable functions (PUFs)

There are some other attack scenarios, besides breaking cipher implementations, where power side-channels have been combined with ML techniques in the context of cryptography. We shortly recap them here but do not consider them in the following discussion.

Zohner et al. [143] used SVMs to attack hash functions. They built profiles for all HWs of each intermediate value on the Grøstl reference implementation, a candidate for SHA-3, in order to solve a system of equations that allows to recover the processed input data.

Mahmoud et al. [77] presented an hybrid power side-channel and modeling attack on strong Physical Unclonable Functions (PUFs). PUFs are a class of security primitives that leverage material imperfection and micro- or nanoscale structural disorders to implement a unique and individual function which is assumed to be non-reproducible, not even for the device manufacturer. They are used, e.g. in challenge-response protocols for authentication or key exchange. The aim of an adversary here is generally to create a correct PUF response for unseen challenge vectors. In the reported attack, the information contained in simulated power traces was used to determine the number of zero and ones before entering the final XOR operation within so-called XOR arbiter or lightweight PUFs. This information helped to improve modeling the PUF behavior using ML methods (the authors especially mention a variant of logistic regression [107]). A similar attack methodology applied to FPGA implementations was shown by Rührmair et al. [108].

5 Discussion

In order to better evaluate the different approaches and give a comprehensive overview, we summarized them in "Appendix" according to the attacked target. It becomes evident that the majority of attacks were performed against block cipher implementations. We assume this is primary due to the publicly available traces from the DPA challenges which are based on hardware or software realizations of DES and AES. (We denote the used data sets in brackets in the targets column where applicable.) In typical ML-related tasks such as object detection, it is common to have several reference data sets to compare the effectiveness of different solutions. However, apart from the DPA contests and the recently released ASCAD database, there are no other public benchmark data sets available in the area of SCAs. This is especially true for attacks against implementations of stream or asymmetric ciphers. Most of the analyzed work therefore includes a detailed description of the used measurement setup. Although this is a crucial point, we do not go into detail here and refer the interested reader to the individual approaches or for a general discussion to [79]. We instead discuss the parts that relate to the general ML pipeline introduced in Sect. 2.

5.1 Preprocessing

Regarding data preprocessing, the presented work can be divided into three main categories. It is either used:

– To transform the data set into another representation (e.g. by applying normalization or wavelet transform),
– To reduce the amount of noise in the traces (e.g. by computing average traces or by lowering the dimension), or
– To ensure that the measurement points are properly aligned

However, in a large number of publications no preprocessing steps are mentioned at all. We assume that for those approaches preprocessing was completely skipped or not considered as useful. Specht et al. [120] furthermore refer to the work of Heyszl et al. [45] which states that compression methods are generally not advantageous for high-resolution EM measurements. This is, however, contradictory to their own clustering-based EM attack [44]. A general recommendation whether to use preprocessing or not is thus hard, but there are some points we want to highlight. First, some ML classifiers are very sensitive to changes in the input space. Ensuring the same scale for all features by using data normalization/standardization implicitly weights them all equally and avoids numerical difficulties in the training process. Second, while realignment techniques normally can be used to defeat shuffling-based countermeasures [79], it has been shown that exactly the opposite approach (deforming the input with data augmentations) leads to better results when combined with CNNs [16]. We therefore argue that the choice of a suitable preprocessing mechanism is dependent on the used classifier.

5.2 Feature engineering

Feature engineering from the ML domain is tightly coupled with POI analysis in SCAs and refers to the question which sample points in the leakage traces correspond to manipulations of the targeted sensitive intermediate variable [141]. There are manifold techniques available in the classical side-channel community for finding those points, and some of them were also applied to ML-based attacks (e.g. SOST). Above all, Pearson correlation as given in Eq. (6) is the most prominent candidate. Actually proposed as side-channel distinguisher to indicate the most promising key candidate in a CPA, it can also be used as a feature selection method. PCA is another widely applied technique. In the reviewed approaches, it served both purposes, trace preprocessing or feature engineering. However, the effect of a dimensionality reduction by PCA on the actual classification accuracy is not clear yet. In some of the previous contributions (e.g. [65,75,120]), a positive effect was noticed, whereas in [76] a negative impact is reported. Also choosing the correct number of retained principal components of the analyzed data set is not an easy task. Often an (arbitrary) value is set as a threshold value. A more analytical, though computational complex, approach is to model the attack efficiency as a function of principal components (as done in [111,140]). Although proper feature selection was in general identified as crucial for attack success by almost all the investigated contributions, there is only a single paper that systematically compares the effectiveness of proposed techniques from the side-channel domain (Pearson correlation, SOST, etc.) in context of profiled ML attacks [98]. Two additional feature selection classes from the ML domain (so-called Wrapper and Hybrid methods) were presented as well, which gave slightly better results. However, these are computationally intractable (search complexity is partly exponential) when dealing with raw, unfiltered traces.

Ideally, an attacker wants to get rid of all the manual feature engineering needed for SCAs. This is where ML algorithms that are capable of automatically determining the most important sample points become interesting. Normal-based feature selection mechanism tailored for SVMs suggested was the first work in this regard [5]. More recently introduced attacks based on deep neural network architectures [16,76] extend this path even further. As already mentioned in Sect. 2, DL techniques such as CNNs are intrinsically able to learn abstract representations that are composed of lower level features. Apart from the ability to automatically identify those features, building a hierarchy of features is especially valuable when performing so-called higher-order DPA (HO-DPA) attacks. These kinds of attacks exploit joint statistical properties of multiple aspects of the analyzed signal and are typically used to defeat masked implementations [32]. In practice, HO-DPA attacks often require to combine the leakage of multiple sample points of a trace (at least when targeting serial block cipher implementations). Template attacks and variants of it naturally perform such a multivariate analysis as well; however, they run into numerical problems when the number of sample points to consider is large.

5.3 Algorithm selection

Choosing a suitable ML technique for a given problem and data set is generally not an easy task since it depends on many factors. However, when examining the list(s) of algorithms we can make some observations. Most of the attacks against block ciphers were based on supervised ML methods. There are only two clustering attacks targeting AES [70,133] and one unsupervised, regression-based approach [22]. The situation for implementation attacks on asymmetric ciphers is not so clear. Here, unsupervised clustering techniques make up approximately half of the total number of published work. Stream ciphers, interestingly, have been exclusively targeted with SVMs (respectively, variants of it). Although certain trends are more or less visible, the choice between supervised and unsupervised techniques is not only affected by the class of attacked cipher implementation, but also on the (assumed) attacker model. If the adversary is in possession of labeled trace data from the device under attack (or able to create those using an exact copy), supervised classifiers are preferable. Unsupervised methods can still be applied in case a profiling is not possible, which is equal to a far weaker attacker.

Looking at the overall, the most commonly of used ML techniques are SVMs followed by MLPs. The initial choice for SVMs was motivated by its good results in other domains [131], its solid mathematical foundation, and its easiness to use. The advantage of MLPs is that they are universal function approximators given the circumstance that the network is given enough hidden units [36] and are therefore (theoretically) able to capture any side-channel leakage [136]. However, they are more expensive to train due to the large number of parameters. Both arguments apply to deep learning techniques as well, but these are even more powerful due to the reasons mentioned above. Simpler models such as kNN and Bayes classification may lead to good results as well [84,93,95]. An adversary or evaluator searching for the correct classifier faces a dilemma called *bias-variance tradeoff* [36]: Is it better to take a model with high representational capacity (meaning the models ability to fit a wide variety of functions and therefore high variance) having the risk of overfitting the training data, or a simpler model that varies not so much but tends to underfit the data (high bias)? The most common approach to solve this problem still used in practice today (and also partly in the analyzed work) is to train a set of possible algorithms and select the one which gives the lowest generalization error on the test set (respectively applying cross-validation). A more systematic way for selecting a suitable algorithm was presented by Lerman et al. [72]. They suggest a framework to decompose the error rate of a classifier into a bias and variance term. By knowing the impact of the contributing factors on the error rate, one can decide if a more complex model (in case the framework detects a high bias) or a simpler model (in case of high variance) is needed in order to increase the success of an attack.

5.4 Hyperparameter optimization

Hyperparameter optimization is related to the question of model selection as well. It refers to the process of finding the optimal parameter settings for a considered algorithm which maximizes its accuracy. In terms of a NN, that means for instance the number of hidden layers or the type of used activation function for the neurons of a certain layer. Variables such as the aforementioned usually have a strong impact on the representational capacity of a ML technique. However, the importance of this step has not been recognized by all authors of the reviewed papers (or was considered as not important for reporting it). In particular, there are only two contributions that explicitly investigated the influence of proper parameter tuning on the effectiveness of SCAs [84,99]. Based on experience gained from our own experiments with NNs, we can confirm that selection of hyperparameters is in fact essential for attack success. Due to space restrictions and improved readability, we have not reported all the various optimization mechanisms in “Appendix.” The range of used techniques, however, goes from just applying standard values taken from literature [5,67] over grid search [41,48] up to advanced methods like PSO [25] and genetic algorithms [76]. When choosing an appropriate algorithm, one should also consider that simpler ML algorithms or standard side-channel analysis tools require less optimization overhead since they usually only have a few parameters that need to be tuned (in case of Naïve Bayes even none at all).

5.5 Model test

Comparing the outlined approaches regarding attack quality is hard due to the different targets and used evaluation methods. While some earlier work on the topic focused on the classification of individual bits of an intermediate value or the secret key [48,65], other authors tried to extract the HW of a certain substate or key bytes [3,16,25,43,70], or even revealed the complete secret [76]. Similarly, attack efficiency was measured differently, whereas accuracy and key guessing entropy are the predominant metrics. However, the latter is an established method to assess efficiency of SCAs and is more suitable than classification accuracy if the occurrence of target classes is not balanced. For example, when considering HW, certain class values are more likely than others.

Another point that complicates comparability are the diverse data sets used. As mentioned above, there are no other side-channel benchmark application than the DPA contests and ASCAD and that is why most approaches were evaluated with self-generated traces or simulations. Additionally, some contributions considered the influence of different noise levels on the analysis [43,64,136] where others did not. We stress that this is a common problem in the SCA community and although there are efforts to standardize the evaluation process (for example the Side-channel Attack Standard Evaluation BOard (SASEBO) project [38]); it is nowadays still hard to reproduce the results of a third party. Nevertheless, in order to give a hint on the attack complexity, we have listed the number of used traces for most approaches.[1]

6 Guidelines for conducting machine learning in side-channel analysis

We already stressed out in the previous section that comparing the analyzed work is difficult, since distinct methodologies were applied, experiments were almost never conducted over the same datasets, and distinct evaluation metrics were applied. In order to be able to properly compare and reproduce results of upcoming work, this section gives recommendations for publishing results of ML techniques in SCAs:

- It should be clarified which kind of preprocessing technique is used. If the analysis is carried out on raw data, this should also be explicitly mentioned.
- The hyperparameters used for each ML algorithm should be described to an extent that the approach can be easily

[1] Note that a lower number not necessarily indicates a more powerful attack due to major differences in the attack methodology among the approaches.

reimplemented by an ML novice. This is especially true for models that exhibit a large number of parameters, such as NNs. Not only the parameters needed to instantiate the ML model (sometimes called architectural parameters), but also the parameters that affect the training procedure should be stated (e.g. optimizer, learning rate, batch size, number of epochs).

– Authors should furthermore explain how they found out the optimal parameters for their ML models. If a hyperparameter search was conducted, the corresponding ranges for each value should be given. The information may be valuable to other researchers looking for options to increase the performance of their models.
– For the experiments, authors should not stick to a single metric to measure the performance of their models. Having a combination of several metrics from ML as well as from the SCA domain eases comparability with related work.
– Cross-validation as discussed in Sect. 2.1 should be applied for estimating the final performance of the model.
– Instead of using a fixed number of traces for training, learning curves should be applied to gain an insight into how the methods behave given more or less data. Alternatively, the restricted attacker framework proposed in [96] can be applied. An extensive study will furthermore successively remove sensitive components from the algorithm in order to identify where the power of a certain approach comes from (e.g. preprocessing) [63].
– The required (resource) overhead for an attacker should be quantified (e.g. minimum number of traces needed, time for training the model, computing resources).
– We finally advise authors, if possible, to publish their software experiments and associated data on an open source repository like GitHub. This is already best practice in the ML area and would certainly stimulate research in the SCA domain.

7 Conclusion and future work

In this work, we presented a comprehensive overview on the various ML methods used in SCAs against cryptographic implementations. After a general comparison of different ML techniques and an introduction to power-based SCAs, we extensively reviewed scientific work and classified them according to the targeted cipher. For each approach, main findings and the steps related to a common ML workflow were reported. Finally, we discussed our observations and gave recommendations on conducting ML experiments for side-channel analysis. Summarizing the result of this study, it becomes clear that ML techniques are a powerful alternative to standard side-channel evaluation methods. For black-box settings without details about the implementation and leakage behavior, they might even be the better tool due to their ability to adapt a wide range of functions. From a practical perspective, the effort for applying ML for SCAs is typically lower since there are several open source implementations available for all types of algorithms (e.g. the python framework scikit-learn [1]).

Future work may investigate the effect of other preprocessing techniques known from the side-channel and signal-processing domain such as linear transforms presented in [92]. The problem of proper hyperparameter selection for the design of ML models (especially of deep NNs) is also heavily discussed in the research community. An interesting research question is if currently proposed methods (for instance based on reinforcement learning [144]) may be beneficial for ML-based SCAs as well. Another important topic is the development of customized learning algorithms for side-channel analysis. The work of Bartkewitz [4] can be seen as a first promising step in this regard. Furthermore, there is up to now no line of work that deals with the application of ML techniques to defeat devices with stronger side-channel countermeasures or devices that combine multiple countermeasures. So-called higher-order threshold implementations [8] that are based on multiparty computation and secret sharing would be an interesting target. Approaches that apply dynamic hardware reconfiguration to effectively hide side-channel leakages [37,115] might also be attractive to attack, since subjects that underlie constant random changes are usually much harder to model with ML methods. As ML is a very volatile domain nowadays, we expect a lot of new ideas and approaches upcoming in close future.

Funding This work is supported in parts by the German Federal Ministry of Education and Research (BMBF) under grant agreement number 16KIS0606K(SecRec).

Appendix

See Tables 1, 2 and 3.

Table 1 Summary attacks on block ciphers

Ref.	Targets (data sets)	Preprocessing	Feature engineering	Machine learning technique	Traces	Evaluation method
[48]	Unprotected SW AES	Outliers removal	Correlation, SOST, PCA	LS-SVM	500–5000 (Train/Test: 70/30)	Accuracy
[43]	Unprotected SW AES		Correlation	SVM	Train/Test: 2700 (300 per HW)/1000	KGE
[5]	Unprotected SW AES	Compression, peak extraction	Normal-based feature selection	SVM	Train: 10–100 traces per HW, Test: 1	KGE
[3]	Unprotected SW AES	Filtering (for ARM platform)	PCA, correlation	SVM, RF, DT, kNN	Train: 25–500 per HW, Test: 450	Accuracy
[95]	Unprotected FPGA AES, Protected SW AES (DPAv2, DPAv4)		Correlation	Naïve Bayes, A1DE	5000–100,000 traces (2:1 split)	Accuracy, AUC, F-measure
[97]	Unprotected FPGA AES, Protected SW AES (DPAv2, DPAv4)		Correlation	Naïve Bayes, rotation forest, SVM, C4.5	20.000 for each data set, train/test: 2/1	Accuracy, AUC, F-measure
[99]	Protected SW AES (DPA Contest v4)		Information gain	SVM, rotation forest, RF, MultiBoost	20.000, train/test: 2/1	Accuracy, AUC, F-measure, data confusion factor
[76]	Unprotected FPGA AES, Unprotected & protected SW AES (DPAv2, DPAv4)	PCA (only for MLP)		CNN, auto-encoder, LSTM, MLP, SVM, RF	Train: 1000 per target intermediate value, Test: 20,000	KGE
[65]	Unprotected FPGA 3DES		PCA, Ranking, mRMR, SOM	SOM, SVM, RF	400 for each key byte	KGE
[74]	Unprotected SW DES		SOST, PCA	SVM	Train: 1000 traces per S-Box subkey, Test: 100	Accuracy
[41]	Unprotected SW DES	Trace averaging, alignment	Handcrafted feature matrix	SVM	Train/Test: 10,000/1000	Accuracy
[138]	Unprotected SW AES	Trace averaging		MLP	Train/Test: 2560/2560	Accuracy
[81]	Unprotected SW AES	Calculation of difference power traces		MLP	Train/Test: 2560/2560	tenfold cross-validation
[82]	Unprotected SW AES	Trace averaging	Variant of SOST	MLP	2560 (tenfold cross-validation)	KGE
[83]	Protected SW AES (DPA v4)		Correlation	MLP	Train/Test: 1000/1500	Success rate
[136]	Unprotected SW AES	Trace averaging		MLP	Train: 256 (one for each intermediate value), Test: 1000	Success rate, KGE, distinctive level
[64]	Unprotected ASIC DES (DPA v1)	Trace averaging	Correlation, MAX, mRMR, SOST	RF, SVM	Train/Test: 6482/1613	Success rate (as function of noise)
[52]	Unprotected SW AES		Correlation, SNR	SVR	Train/Test: 25,000/25,000	KGE

Table 1 continued

Ref.	Targets (data sets)	Preprocessing	Feature engineering	Machine learning technique	Traces	Evaluation method
[133]	Unprotected SW/HW AES		PCA	*k*-means, hierarchical clustering	SW: 10,000, HW: 5000	KGE
[139]	Protected SW AES		Correlation	SVM	Train: 2000–10,000, Test: 90,000	Accuracy
[67,69]	Protected SW AES (DPA v4)		Correlation	SVM, RF	Train/Test: 1500/1500	Success rate
[35]	Protected SW AES (DPA v4)	Normalization, PCA	Correlation	MLP	Train/Val./Test: 7000/1500/1500	Error rate
[80]	Protected SW AES (DPA Contest v4.2)			MLP	Test: 100	Accuracy
[84]	Unprotected & unprotected SW AES (DPA v4)		Correlation	kNN, SVM, DT, MLP, RF	DS1: 2560, DS2: 1000, DS3: 1000	Success rate
[49]	Unprotected FPGA AES & protected SW AES (DPA v4)		Correlation	SVM	Train/Test: 500/250	Accuracy, KGE
[113,114]	Unprotected AES simulation	Wavelet transform, normalization	PCA	Probabilistic NN	Train/Test: 72/48	Success rate
[71]	Unprotected AES simulation		Random (intrinsic in RF)	RF, SVM		Success rate
[42]	Unprotected simulations PRESENT & AES		Single feature used	Naïve Bayes, C4.5, MLP	10,000, 30,000, 50,000 (Train/Test: $\frac{2}{3}/\frac{1}{3}$)	Accuracy
[22]	Unprotected ASIC DES (DPA v1)	Normalization, trace averaging	20 samples per trace	Regression (unsupervised)	1000	Success rate
[70]	Unprotected SW AES	Trace averaging (10x128)	Mutual information	PAM (clustering)	Train/Test: 8/2 (per byte)	Success rate
[73]	Protected SW AES (DPA v4)		Correlation	SVM	Train: 5000 (labeled: 10–100), Test: 1000	Accuracy
[68]	Protected SW AES (DPA v4.2)		Correlation	SVM, RF, MLP	Train: 500–4000, Test: 1000	Success rate
[16]	Protected SW/HW AES	Data augmentation		CNN	SW-Count: Train/Val. 1000/700, HW-Count: Train/Val./Test 9000/1000/100,000, HW AES: Train/Val: 98,000/1000	Accuracy, KGE
[101]	Protected SW AES (ASCAD)			MLP, CNN	Train/Test: 50,000/10,000	KGE, accuracy
[130]	Unprotected & protected SW AES (ASCAD)	Data augmentation		MLP, CNN	20,000	Success rate, number of epochs

Table 2 Summary attacks on stream ciphers

Ref.	Targets	Preprocessing	Feature engineering	Machine learning technique	Traces	Evaluation method
[19]	Unprotected FPGA Grain v1	Trace averaging	SNR	LS-SVM	Train/Test: 2500/1000	Accuracy
[25]	Unprotected SW RC4	Signal alignment (pattern matching)	Correlation	DAG-SVM	Train/Test: 1000/300	Accuracy
[20]	Unprotected FPGA MICKEY-128 2.0	Trace averaging	Correlation	LS-SVM	Train: 1000, Test: 8–641	Accuracy
[140]	Unprotected SW RC4	Wavelet transform	PCA	SVM		Success rate

Table 3 Summary attacks on asymmetric ciphers

Ref.	Target	Preprocessing	Feature engineering	Machine learning technique	Traces	Evaluation method
[44]	Protected FPGA ECC	Compression		K-means	1 (combined from 1 to 9 different positions)	Remaining brute-force complexity
[120]	Protected FPGA ECC		PCA	Expectation–maximization clustering	1 (combined from 3 positions)	Remaining brute-force complexity
[94]	Protected FPGA RSA	Trace alignment, compression	k-means, difference of means	Fuzzy k-means	1	Accuracy
[53]	Protected SW ECC	Compression	Manual	K-means	1	Success rate
[111]	FPGA ECC		PCA	SVM	1000	Time complexity, accuracy
[109]	FPGA ECC		PCA	SVM	5120	tenfold cross-validation
[110]	FPGA ECC		PCA	CFBP	Train/Test: 5120/605	Confusion-matrix, ROC, error histogram
[112]	FPGA ECC			LVQ NN		Confusion-matrix, ROC, error histogram
[93]	Protected SW BR256r1		Correlation	Naïve Bayes, kNN, SVM	Train: 10–160 traces per scalar bit, Test: 1	Success rate
[17]	Protected SW RSA	Trace averaging	SNR	LS-SVM	Train/Test: 200/150	Accuracy

References

1. Scikit-learn: Machine learning in Python. http://scikit-learn.org/stable/. Accessed 19 Mar 2019
2. Alva, J.A.V., Estrada, E.G.: A generalization of Shapiro–Wilk's test for multivariate normality. Commun. Stat. Theory Methods **38**(11), 1870–1883 (2009)
3. Banciu, V., Oswald, E., Whitnall, C.: Reliable information extraction for single trace attacks. In: Proceedings of the 2015 Design, Automation and Test in Europe Conference, DATE '15, pp. 133–138. EDA Consortium, San Jose (2015)
4. Bartkewitz, T.: Leakage prototype learning for profiled differential side-channel cryptanalysis. IEEE Trans. Comput. **65**(6), 1761–1774 (2016)
5. Bartkewitz, T., Lemke-Rust, K.: Efficient template attacks based on probabilistic multi-class support vector machines. In: Mangard, S. (ed.) Smart Card Research and Advanced Applications: 11th International Conference, CARDIS 2012, Graz, Austria, November 28–30, 2012, Revised Selected Papers, pp. 263–276. Springer, Berlin (2013)
6. Batina, L., Hogenboom, J., van Woudenberg, J.G.J.: Getting more from PCA: first results of using principal component analysis for extensive power analysis. In: Dunkelman, O. (ed.) Topics in Cryptology—CT-RSA 2012: The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27–March 2, 2012. Proceedings, pp. 383–397. Springer, Berlin (2012)
7. Bhasin, S., Bruneau, N., Danger, J.L., Guilley, S., Najm, Z.: Analysis and improvements of the DPA contest v4 implementation. In: Chakraborty, R.S., Matyas, V., Schaumont, P. (eds.) Security, Privacy, and Applied Cryptography Engineering: 4th International Conference, SPACE 2014, Pune, India, October 18–22, 2014. Proceedings, pp. 201–218. Springer, Cham (2014)
8. Bilgin, B., Gierlichs, B., Nikova, S., Nikov, V., Rijmen, V.: Higher-order threshold implementations. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology—ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014, Proceedings, Part II, pp. 326–343. Springer, Berlin (2014)
9. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press Inc, New York (1995)
10. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
11. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.J. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11–13, 2004. Proceedings, pp. 16–29. Springer, Berlin (2004)
12. Brier, E., Clavier, C., Olivier, F.: Improved template attacks. In: COSADE 2010—First International Workshop on Constructive Side-Channel Analysis and Secure Design (2010)
13. Bruneau, N., Guilley, S., Heuser, A., Marion, D., Rioul, O.: Less is more. In: Güneysu, T., Handschuh, H. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2015, pp. 22–41. Springer, Berlin (2015)
14. Burman, S., Mukhopadhyay, D., Veezhinathan, K.: Lfsr based stream ciphers are vulnerable to power attacks. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) Progress in Cryptology—INDOCRYPT 2007: 8th International Conference on Cryptology in India, Chennai, India, December 9–13, 2007. Proceedings, pp. 384–392. Springer, Berlin (2007)
15. Cagli, E., Dumas, C., Prouff, E.: Enhancing dimensionality reduction methods for side-channel attacks. In: Homma, N., Medwed, M. (eds.) Smart Card Research and Advanced Applications, pp. 15–33. Springer, Cham (2016)
16. Cagli, E., Dumas, C., Prouff, E.: Convolutional neural networks with data augmentation against jitter-based countermeasures. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2017: 19th International Conference, Taipei, Taiwan, September 25–28, 2017, Proceedings, pp. 45–68. Springer, Cham (2017)
17. Chakraborty, A.: Template attack on SPA and FA resistant implementation of montgomery ladder. IET Inf. Secur. **10**(6), 245–251 (2016)
18. Chakraborty, A., Mazumdar, B., Mukhopadhay, D.: Combined side-channel and fault analysis attack on protected grain family of stream ciphers. Cryptology ePrint Archive, Report 2015/602 (2015)
19. Chakraborty, A., Mazumdar, B., Mukhopadhyay, D.: A practical DPA on grain v1 using LS-SVM. In: 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 44–47 (2015)
20. Chakraborty, A., Mukhopadhyay, D.: A practical template attack on mickey-128 2.0 using PSO generated IVS and LS-SVM. In: 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), pp. 529–534 (2016)
21. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. Cryptographic Hardware and Embedded Systems–CHES 2002: 4th International Workshop Redwood Shores. CA, USA, August 13–15, 2002 Revised Papers, pp. 13–28. Springer, Berlin (2003)
22. Chou, J.W., Chu, M.H., Tsai, Y.L., Jin, Y., Cheng, C.M., Lin, S.D.: An unsupervised learning model to perform side channel attack. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) Advances in Knowledge Discovery and Data Mining: 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14–17, 2013, Proceedings, Part I, pp. 414–425. Springer, Berlin (2013)
23. Choudary, O., Kuhn, M.G.: Efficient template attacks. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27–29, 2013. Revised Selected Papers, pp. 253–270. Springer, Cham (2014)
24. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)
25. Duan, L., Hongxin, Z., Qiang, L., Xinjie, Z., Pengfei, H.: Electromagnetic side-channel attack based on PSO directed acyclic graph SVM. J. China Univ. Posts Telecommun. **22**(5), 10–15 (2015)
26. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience, New York (2000)
27. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS '95, pp. 39–43 (1995)
28. Eisenbarth, T., Paar, C., Weghenkel, B.: Building a side channel based disassembler. Transactions on Computational Science X: Special Issue on Security in Computing, Part I, pp. 78–99. Springer, Berlin (2010)
29. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997). https://doi.org/10.1006/jcss.1997.1504
30. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2001: Third International Workshop Paris, France, May 14–16, 2001 Proceedings, pp. 251–261. Springer, Berlin (2001)
31. Genkin, D., Shamir, A., Tromer, E.: Acoustic cryptanalysis. J. Cryptol. **30**(2), 392–443 (2017)
32. Gierlichs, B., Batina, L., Preneel, B., Verbauwhede, I.: Revisiting higher-order DPA attacks. In: Pieprzyk, J. (ed.) Topics in

Cryptology—CT-RSA 2010: The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1–5, 2010. Proceedings, pp. 221–234. Springer, Berlin (2010)
33. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis. In: Oswald, E., Rohatgi, P. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2008: 10th International Workshop, Washington, DC, USA, August 10–13, 2008. Proceedings, pp. 426–442. Springer, Berlin (2008)
34. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. stochastic methods. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2006: 8th International Workshop, Yokohama, Japan, October 10–13, 2006. Proceedings, pp. 15–29. Springer, Berlin (2006)
35. Gilmore, R., Hanley, N., O'Neill, M.: Neural network based attack on a masked implementation of AES. In: 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 106–111 (2015)
36. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
37. Güneysu, T., Moradi, A.: Generic side-channel countermeasures for reconfigurable devices. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011. Proceedings, pp. 33–48. Springer, Berlin (2011)
38. Guntur, H., Ishii, J., Satoh, A.: Side-channel attack user reference architecture board SAKURA-G. In: 2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE), pp. 271–274 (2014)
39. Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K.: KNN model-based approach in classification. In: Meersman, R., Tari, Z., Schmidt, D.C. (eds.) On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3–7, 2003. Proceedings, pp. 986–996. Springer, Berlin (2003)
40. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction, 2nd edn. Springer, Berlin (2009)
41. He, H., Jaffe, J., Zou, L.: Side channel cryptanalysis using machine learning. Standford University, CS229 Fall Project (2012)
42. Heuser, A., Picek, S., Guilley, S., Mentens, N.: Side-channel analysis of lightweight ciphers: does lightweight equal easy? Cryptology ePrint Archive, Report 2017/261. http://eprint.iacr.org/2017/261 (2017). Accessed 19 Mar 2019
43. Heuser, A., Zohner, M.: Intelligent machine homicide. In: Schindler, W., Huss, S.A. (eds.) Constructive Side-Channel Analysis and Secure Design: Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3–4, 2012. Proceedings. Springer, Berlin (2012)
44. Heyszl, J., Ibing, A., Mangard, S., De Santis, F., Sigl, G.: Clustering algorithms for non-profiled single-execution attacks on exponentiations. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27–29, 2013. Revised Selected Papers, pp. 79–93. Springer, Cham (2014)
45. Heyszl, J., Merli, D., Heinz, B., De Santis, F., Sigl, G.: Strengths and limitations of high-resolution electromagnetic field measurements for side-channel analysis. In: Mangard, S. (ed.) Smart Card Research and Advanced Applications: 11th International Conference, CARDIS 2012, Graz, Austria, November 28–30, 2012. Revised Selected Papers, pp. 248–262. Springer, Berlin (2013)
46. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
47. Hoogvorst, P.: The variance power analysis. In: COSADE 2010—First International Workshop on Constructive Side-Channel Analysis and Secure Design (2010)
48. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. J. Cryptogr. Eng. **1**(4), 293 (2011)
49. Hou, S., Zhou, Y., Liu, H., Zhu, N.: Wavelet support vector machine algorithm in power analysis attacks. Radioengineering **26**(3), 890–902 (2017)
50. Huang, J., Zhou, Y., Liu, J.: Measuring the effectiveness of DPA attacks-from the perspective of distinguishers' statistical characteristics. In: 2010 3rd International Conference on Computer Science and Information Technology, vol. 4, pp. 161–168 (2010)
51. Jap, D., Breier, J.: Overview of machine learning based side-channel analysis methods. In: 2014 International Symposium on Integrated Circuits (ISIC), pp. 38–41 (2014)
52. Jap, D., Stöttinger, M., Bhasin, S.: Support vector regression: exploiting machine learning techniques for leakage modeling. In: Proceedings of the Fourth Workshop on Hardware and Architectural Support for Security and Privacy, HASP '15, pp. 2:1–2:8 (2015)
53. Järvinen, K., Balasch, J.: Single-trace side-channel attacks on scalar multiplications with precomputations. In: Lemke-Rust, K., Tunstall, M. (eds.) Smart Card Research and Advanced Applications: 15th International Conference, CARDIS 2016, Cannes, France, November 7–9, 2016. Revised Selected Papers, pp. 137–155. Springer, Cham (2017)
54. Jordan, M.I., Mitchell, T.M.: Machine learning: trends, perspectives, and prospects. Science **349**(6245), 255–260 (2015)
55. Joye, M.: Elliptic curves and side-channel analysis. ST J. Syst. Res. **4**, 17–21 (2003)
56. Joye, M., Yen, S.M.: The Montgomery powering ladder. In: Kaliski, B.S., Koç, K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002. Revised Papers, pp. 291–302. Springer, Berlin (2003)
57. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR arXiv:1412.6980 (2014)
58. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. Advances in Cryptology–CRYPTO' 99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999. Proceedings, pp. 388–397. Springer, Berlin (1999)
59. Kocher, P., Jaffe, J., Jun, B., Rohatgi, P.: Introduction to differential power analysis. J. Cryptogr. Eng. **1**(1), 5–27 (2011)
60. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. Advances in Cryptology–CRYPTO '96: 16th Annual International Cryptology Conference Santa Barbara. California, USA August 18–22, 1996 Proceedings, pp. 104–113. Springer, Berlin (1996)
61. Koeune, F., Standaert, F.X.: A tutorial on physical security and side-channel attacks. In: Aldini, A., Gorrieri, R., Martinelli, F. (eds.) Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures, pp. 78–108. Springer, Berlin (2005)
62. Kotsiantis, S.B.: Supervised machine learning: a review of classification techniques. In: Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering, pp. 3–24. IOS Press (2007)
63. Langley, P.: Crafting papers on machine learning. In: Proceedings of the Seventeenth International Conference on Machine Learning (ICML), pp. 1207–1212 (2000)
64. Lerman, L., Bontempi, G., Ben Taieb, S., Markowitch, O.: A time series approach for profiling attack. In: Gierlichs, B., Guilley, S., Mukhopadhyay, D. (eds.) Security, Privacy, and Applied Cryptography Engineering: Third International Conference, SPACE 2013, Kharagpur, India, October 19–23, 2013. Proceedings, pp. 75–94. Springer, Berlin (2013)
65. Lerman, L., Bontempi, G., Markowitch, O.: Side channel attack: an approach based on machine learning. In: COSADE 2011—

Second International Workshop on Constructive Side-Channel Analysis and Secure Design (2011)
66. Lerman, L., Bontempi, G., Markowitch, O.: The bias-variance decomposition in profiled attacks. J. Cryptogr. Eng. **5**(4), 255–267 (2015). https://doi.org/10.1007/s13389-015-0106-1
67. Lerman, L., Bontempi, G., Markowitch, O.: A machine learning approach against a masked AES. J. Cryptogr. Eng. **5**(2), 123–139 (2015)
68. Lerman, L., Martinasek, Z., Markowitch, O.: Robust profiled attacks: should the adversary trust the dataset? IET Inf. Secur. **11**(4), 188–194 (2017)
69. Lerman, L., Medeiros, S.F., Bontempi, G., Markowitch, O.: A machine learning approach against a masked AES. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27–29, 2013. Revised Selected Papers, pp. 61–75. Springer, Berlin (2014)
70. Lerman, L., Medeiros, S.F., Veshchikov, N., Meuter, C., Bontempi, G., Markowitch, O.: Semi-supervised template attack. In: Prouff, E. (ed.) Constructive Side-Channel Analysis and Secure Design: 4th International Workshop, COSADE 2013, Paris, France, March 6–8, 2013. Revised Selected Papers. Springer, Berlin (2013)
71. Lerman, L., Poussier, R., Bontempi, G., Markowitch, O., Standaert, F.X.: Template attacks vs. machine learning revisited (and the curse of dimensionality in side-channel analysis). In: Mangard, S., Poschmann, A.Y. (eds.) Constructive Side-Channel Analysis and Secure Design: 6th International Workshop, COSADE 2015, Berlin, Germany, April 13–14, 2015. Revised Selected Papers, pp. 20–33. Springer, Cham (2015)
72. Lerman, L., Veshchikov, N., Markowitch, O., Standaert, F.: Start simple and then refine: bias-variance decomposition as a diagnosis tool for leakage profiling. IEEE Trans. Comput. **67**(2), 268–283 (2018). https://doi.org/10.1109/TC.2017.2731342
73. Liu, B., Ding, Z., Pan, Y., Li, J., Feng, H.: Side-channel attacks based on collaborative learning. Data Science: Third International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2017, Changsha, China, September 22–24, 2017. Proceedings, Part I, pp. 549–557. Springer, Singapore (2017)
74. Liu, B., Feng, H., Yuan, Z., Gao, Y.: Learning to attack from electromagnetic emanation. In: 2012 6th Asia-Pacific Conference on Environmental Electromagnetics (CEEM), pp. 202–205 (2012)
75. Liu, J., Zhou, Y., Han, Y., Li, J., Yang, S., Feng, D.: How to characterize side-channel leakages more accurately? In: Bao, F., Weng, J. (eds.) Information Security Practice and Experience: 7th International Conference, ISPEC 2011, Guangzhou, China, May 30–June 1, 2011. Proceedings, pp. 196–207. Springer, Berlin (2011)
76. Maghrebi, H., Portigliatti, T., Prouff, E.: Breaking cryptographic implementations using deep learning techniques. In: Carlet, C., Hasan, M.A., Saraswat, V. (eds.) Security, Privacy, and Applied Cryptography Engineering: 6th International Conference, SPACE 2016, Hyderabad, India, December 14–18, 2016. Proceedings, pp. 3–26. Springer, Cham (2016)
77. Mahmoud, A., Rührmair, U., Majzoobi, M., Koushanfar, F.: Combined modeling and side channel attacks on strong PUFs. Cryptology ePrint Archive, Report 2013/632. https://eprint.iacr.org/2013/632 (2013). Accessed 19 Mar 2019
78. Mangard, S.: A simple power-analysis (SPA) attack on implementations of the AES key expansion. In: Lee, P.J., Lim, C.H. (eds.) Information Security and Cryptology—ICISC 2002: 5th International Conference Seoul, Korea, November 28–29, 2002. Revised Papers, pp. 343–358. Springer, Berlin (2003)
79. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards, 1st edn. Springer, New York (2010)
80. Martinasek, Z., Dzurenda, P., Malina, L.: Profiling power analysis attack based on MLP in DPA contest v4.2. In: 2016 39th International Conference on Telecommunications and Signal Processing (TSP), pp. 223–226 (2016)
81. Martinasek, Z., Hajny, J., Malina, L.: Optimization of power analysis using neural network. In: Francillon, A., Rohatgi, P. (eds.) Smart Card Research and Advanced Applications: 12th International Conference, CARDIS 2013, Berlin, Germany, November 27–29, 2013. Revised Selected Papers, pp. 94–107. Springer, Cham (2014)
82. Martinasek, Z., Malina, L.: Comparison of profiling power analysis attacks using templates and multi-layer perceptron network. Math. Methods Sci. Eng. (2014)
83. Martinasek, Z., Malina, L., Trasy, K.: Profiling power analysis attack based on multi-layer perceptron network. In: Mastorakis, N., Bulucea, A., Tsekouras, G. (eds.) Computational Problems in Science and Engineering, pp. 317–339. Springer, Cham (2015)
84. Martinasek, Z., Zeman, V., Malina, L., Martinasek, J.: k-Nearest neighbors algorithm in profiling power analysis attacks. Radioengineering **25**(2), 365–382 (2016)
85. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) Artificial Neural Networks and Machine Learning—ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14–17, 2011. Proceedings, Part I, pp. 52–59. Springer, Berlin (2011)
86. Mitchell, T.M.: Machine Learning, 1st edn. McGraw-Hill Inc, New York (1997)
87. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge (2012)
88. Murthy, S.K.: Automatic construction of decision trees from data: a multi-disciplinary survey. Data Min. Knowl. Discov. **2**(4), 345–389 (1998)
89. Nassar, M., Souissi, Y., Guilley, S., Danger, J.L.: RSM: a small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In: 2012 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 1173–1178 (2012)
90. Okeya, K., Takagi, T.: The width-w NAF method provides small memory and fast elliptic scalar multiplications secure against side channel attacks. In: Joye, M. (ed.) Topics in Cryptology—CT-RSA 2003: The Cryptographers' Track at the RSA Conference 2003 San Francisco, CA, USA, April 13–17, 2003. Proceedings, pp. 328–343. Springer, Berlin (2003)
91. O'Shea, K., Nash, R.: An introduction to convolutional neural networks. CoRR arXiv:1511.08458 (2015)
92. Oswald, D., Paar, C.: Improving side-channel analysis with optimal linear transforms. In: Mangard, S. (ed.) Smart Card Research and Advanced Applications: 11th International Conference, CARDIS 2012, Graz, Austria, November 28–30, 2012. Revised Selected Papers, pp. 219–233. Springer, Berlin (2013)
93. Özgen, E., Papachristodoulou, L., Batina, L.: Template attacks using classification algorithms. In: 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 242–247 (2016)
94. Perin, G., Imbert, L., Torres, L., Maurine, P.: Attacking randomized exponentiations using unsupervised learning. In: Prouff, E. (ed.) Constructive Side-Channel Analysis and Secure Design: 5th International Workshop, COSADE 2014, Paris, France, April 13–15, 2014. Revised Selected Papers, pp. 144–160. Springer, Cham (2014)
95. Picek, S., Heuser, A., Guilley, S.: Template attack versus Bayes classifier. J. Cryptogr. Eng. **7**(4), 343–351 (2017)

96. Picek, S., Heuser, A., Guilley, S.: Profiling side-channel analysis in the restricted attacker framework. Cryptology ePrint Archive, Report 2019/168. https://eprint.iacr.org/2019/168 (2019). Accessed 19 Mar 2019
97. Picek, S., Heuser, A., Jovic, A., Legay, A.: Climbing down the hierarchy: hierarchical classification for machine learning side-channel attacks. In: Joye, M., Nitaj, A. (eds.) Progress in Cryptology—AFRICACRYPT 2017: 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24–26, 2017. Proceedings, pp. 61–78. Springer, Cham (2017)
98. Picek, S., Heuser, A., Jovic, A., Legay, A.: On the relevance of feature selection for profiled side-channel attacks. Cryptology ePrint Archive, Report 2017/1110. https://eprint.iacr.org/2017/1110 (2017). Accessed 19 Mar 2019
99. Picek, S., Heuser, A., Jovic, A., Ludwig, S.A., Guilley, S., Jakobovic, D., Mentens, N.: Side-channel analysis and machine learning: A practical perspective. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 4095–4102 (2017)
100. Powers, D.M.W.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. J. Mach. Learn. Technol. **2**, 37–63 (2011)
101. Prouff, E., Strullu, R., Benadjila, R., Cagli, E., Dumas, C.: Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. Cryptology ePrint Archive, Report 2018/053. https://eprint.iacr.org/2018/053 (2018). Accessed 19 Mar 2019
102. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
103. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (EMA): measures and counter-measures for smart cards. Smart Card Programming and Security: International Conference on Research in Smart Cards, E-smart 2001 Cannes, France, September 19–21, 2001. Proceedings, pp. 200–210. Springer, Berlin (2001)
104. Raschka, S.: Linear discriminant analysis: bit by bit. https://sebastianraschka.com/Articles/\2014_python_lda.html. Accessed 27 Oct 2018
105. Renauld, M., Standaert, F.X.: Algebraic side-channel attacks. In: Bao, F., Yung, M., Lin, D., Jing, J. (eds.) Information Security and Cryptology: 5th International Conference, Inscrypt 2009, Beijing, China, December 12–15, 2009. Revised Selected Papers, pp. 393–410. Springer, Berlin (2010)
106. Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: a new classifier ensemble method. IEEE Trans. Pattern Anal. Mach. Intell. **28**(10), 1619–1630 (2006)
107. Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., Schmidhuber, J.: Modeling attacks on physical unclonable functions. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS '10, pp. 237–249. ACM (2010)
108. Rührmair, U., Xu, X., Sölter, J., Mahmoud, A., Majzoobi, M., Koushanfar, F., Burleson, W.: Efficient power and timing side channels for physical unclonable functions. In: Batina, L., Robshaw, M. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2014: 16th International Workshop, Busan, South Korea, September 23–26, 2014. Proceedings, pp. 476–492. Springer, Berlin (2014)
109. Saeedi, E., Hossain, M.S., Kong, Y.: Multi-class SVMs analysis of side-channel information of elliptic curve cryptosystem. In: Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems, Spects '15, pp. 1–6. Society for Computer Simulation International, San Diego (2015)
110. Saeedi, E., Hossain, M.S., Kong, Y.: Side-channel information characterisation based on cascade-forward back-propagation neural network. J. Electron. Test. **32**(3), 345–356 (2016)
111. Saeedi, E., Kong, Y.: Side channel information analysis based on machine learning. In: 2014 8th International Conference on Signal Processing and Communication Systems (ICSPCS), pp. 1–7 (2014)
112. Saeedi, E., Kong, Y., Hossain, M.S.: Side-channel attacks and learning-vector quantization. Front. Inf. Technol. Electron. Eng. **18**(4), 511–518 (2017)
113. Saravanan, P., Kalpana, P.: A novel approach to attack smartcards using machine learning method. J. Sci. Ind. Res. (JSIR) **76**, 95–99 (2017)
114. Saravanan, P., Kalpana, P., Preethisri, V., Sneha, V.: Power analysis attack using neural networks with wavelet transform as pre-processor. In: 18th International Symposium on VLSI Design and Test, pp. 1–6 (2014)
115. Sasdrich, P., Moradi, A., Mischke, O., Güneysu, T.: Achieving side-channel protection with dynamic logic reconfiguration on modern FPGAs. In: 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 130–136 (2015). https://doi.org/10.1109/HST.2015.7140251
116. Schetinin, V.: An evolving cascade neural network technique for cleaning sleep electroencephalograms. CoRR arXiv:cs/0504067 (2005)
117. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2005: 7th International Workshop, Edinburgh, UK, August 29–September 1, 2005. Proceedings, pp. 30–46. Springer, Berlin (2005)
118. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
119. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR arXiv:1409.1556 (2014)
120. Specht, R., Heyszl, J., Kleinsteuber, M., Sigl, G.: Improving non-profiled attacks on exponentiations based on clustering and extracting leakage from multi-channel high-resolution EM measurements. In: Mangard, S., Poschmann, A.Y. (eds.) Constructive Side-Channel Analysis and Secure Design: 6th International Workshop, COSADE 2015, Berlin, Germany, April 13–14, 2015. Revised Selected Papers, pp. 3–19. Springer, Cham (2015)
121. Standaert, F.X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Oswald, E., Rohatgi, P. (eds.) Cryptographic Hardware and Embedded Systems-CHES 2008, pp. 411–425. Springer, Berlin (2008)
122. Standaert, F.X., Gierlichs, B., Verbauwhede, I.: Partition vs. comparison side-channel distinguishers: an empirical evaluation of statistical tests for univariate side-channel attacks against two unprotected CMOs devices. In: Lee, P.J., Cheon, J.H. (eds.) Information Security and Cryptology—ICISC 2008: 11th International Conference, Seoul, Korea, December 3–5, 2008. Revised Selected Papers, pp. 253–267. Springer, Berlin (2009)
123. Standaert, F.X., Malkin, T.G., Yung, M.: A unified framework for the analysis of side-channel key recovery attacks. In: Joux, A. (ed.) Advances in Cryptology—EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26–30, 2009. Proceedings. Springer, Berlin (2009)
124. Sugawara, T., Homma, N., Aoki, T., Satoh, A.: Profiling attack using multivariate regression analysis. IEICE Electron. Express **7**(15), 1139–1144 (2010). https://doi.org/10.1587/elex.7.1139
125. TELECOM ParisTech SEN research group: DPA Contest v1. http://www.dpacontest.org/index.php. Accessed 19 Mar 2019
126. TELECOM ParisTech SEN research group: DPA Contest v2. http://www.dpacontest.org/v2/. Accessed 19 Mar 2019
127. TELECOM ParisTech SEN research group: DPA Contest v4. http://www.dpacontest.org/v4/index.php. Accessed 19 Mar 2019

128. Theodoridis, S., Koutroumbas, K.: Pattern Recognition, 4th edn. Academic Press Inc, Orlando (2008)
129. Thillard, A., Prouff, E., Roche, T.: Success through confidence: evaluating the effectiveness of a side-channel attack. In: Bertoni, G., Coron, J.S. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2013: 15th International Workshop, Santa Barbara, CA, USA, August 20–23, 2013. Proceedings, pp. 21–36. Springer, Berlin (2013)
130. Timon, B.: Non-profiled deep learning-based side-channel attacks. Cryptology ePrint Archive, Report 2018/196. https://eprint.iacr.org/2018/196 (2018)
131. van Gestel, T., Suykens, J.A., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., de Moor, B., Vandewalle, J.: Benchmarking least squares support vector machine classifiers. Mach. Learn. **54**(1), 5–32 (2004)
132. Webb, G.I., Boughton, J.R., Wang, Z.: Not so naive Bayes: aggregating one-dependence estimators. Mach. Learn. **58**(1), 5–24 (2005)
133. Whitnall, C., Oswald, E.: Robust profiling for DPA-style attacks. In: Güneysu, T., Handschuh, H. (eds.) Cryptographic Hardware and Embedded Systems—CHES 2015: 17th International Workshop, Saint-Malo, France, September 13–16, 2015. Proceedings, pp. 3–21. Springer, Berlin (2015)
134. Whitnall, C., Oswald, E., Mather, L.: An exploration of the Kolmogorov–Smirnov test as a competitor to mutual information analysis. In: Prouff, E. (ed.) Smart Card Research and Advanced Applications, pp. 234–251. Springer, Berlin (2011)
135. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evolut. Comput. **1**(1), 67–82 (1997)
136. Yang, S., Zhou, Y., Liu, J., Chen, D.: Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations. In: Kim, H. (ed.) Information Security and Cryptology—ICISC 2011: 14th International Conference, Seoul, Korea, November 30–December 2, 2011. Revised Selected Papers, pp. 169–185. Springer, Berlin (2012)
137. Zadeh, A.A., Heys, H.M.: Simple power analysis applied to nonlinear feedback shift registers. IET Inf. Secur. **8**(3), 188–198 (2014)
138. Zdenek, M., Zeman, V.: Innovative method of the power analysis. Radioengineering **22**(2), 586–594 (2013)
139. Zeng, Z., Gu, D., Liu, J., Guo, Z.: An improved side-channel attack based on support vector machine. In: 2014 Tenth International Conference on Computational Intelligence and Security, pp. 676–680 (2014)
140. Zhang, H., Han, G., Li, J.: Wavelet transform-principal component analysis in electromagnetic attack. In: 2015 7th Asia-Pacific Conference on Environmental Electromagnetics (CEEM), pp. 420–423 (2015)
141. Zheng, Y., Zhou, Y., Yu, Z., Hu, C., Zhang, H.: How to compare selections of points of interest for side-channel distinguishers in practice? In: Hui, L.C.K., Qing, S.H., Shi, E., Yiu, S.M. (eds.) Information and Communications Security: 16th International Conference, ICICS 2014, Hong Kong, China, December 16–17, 2014. Revised Selected Papers, pp. 200–214. Springer, Cham (2015)
142. Zhuang, L., Zhou, F., Tygar, J.D.: Keyboard acoustic emanations revisited. ACM Trans. Inf. Syst. Secur. **13**(1), 3:1–3:26 (2009)
143. Zohner, M., Kasper, M., Stöttinger, M., Huss, S.A.: Side channel analysis of the SHA-3 finalists. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE ’12, pp. 1012–1017. EDA Consortium, San Jose (2012)
144. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. ArXiv e-prints (2016)