# $k$-Nearest Neighbors Algorithm in Profiling Power Analysis Attacks

*Zdenek MARTINASEK* [1], *Vaclav ZEMAN* [1], *Lukas MALINA* [1], *Josef MARTINASEK* [2]

[1] Dept. of Telecommunications, Faculty of Electrical Engineering and Communication, Brno University of Technology, Technická 3082/12, 616 00 Brno, Czech Republic
[2] Institute of Structural Mechanics, Faculty of Civil Engineering, Brno University of Technology, Veveří 95, 602 00 Brno, Czech Republic

{martinasek, zeman, malina}@feec.vutbr.cz, martinasek.j@fce.vutbr.cz

Manuscript received July 17, 2015

**Abstract.** *Power analysis presents the typical example of successful attacks against trusted cryptographic devices such as RFID (Radio-Frequency IDentifications) and contact smart cards. In recent years, the cryptographic community has explored new approaches in power analysis based on machine learning models such as Support Vector Machine (SVM), RF (Random Forest) and Multi-Layer Perceptron (MLP). In this paper, we made an extensive comparison of machine learning algorithms in the power analysis. For this purpose, we implemented a verification program that always chooses the optimal settings of individual machine learning models in order to obtain the best classification accuracy. In our research, we used three datasets, the first contains the power traces of an unprotected AES (Advanced Encryption Standard) implementation. The second and third datasets are created independently from public available power traces corresponding to a masked AES implementation (DPA Contest v4). The obtained results revealed some interesting facts, namely, an elementary k-NN (k-Nearest Neighbors) algorithm, which has not been commonly used in power analysis yet, shows great application potential in practice.*

## Keywords

Power analysis, machine learning, template attack, comparison, smart cards

## 1. Introduction

Power analysis (PA) measures and analyzes the power consumption of cryptographic devices depending on their activity. The main goal of PA is to determine the sensitive information from the power consumption measured and to apply the information obtained in order to abuse the cryptographic device. There are two basic methods of power analysis: simple and differential. Simple power analysis (SPA) tries to determine the sensitive information (mostly encryption key that is stored in the device) more or less directly from one power traces measured. A typical example of the SPA is the attack aimed on the implementation of the asymmetric cryptographic algorithm RSA (Rivest Shamir Adleman), where the differences in power consumption revels private key [1] (implementation of Square and multiply algorithm). The goal of differential power analysis (DPA) attacks is to reveal the secret key of the cryptographic device based on a large number of the power traces that have been recorded while the device was encrypting various input data. The basic principle of DPA was introduced on DES (Data Encryption Standard) algorithm using the statistical method based on Difference of Means [2]. A general description of PA attacks is presented in [3], [4].

From a different perspective, we can divide the power analysis attacks into two main categories, namely profiling and non-profiling attacks. In profiling attacks, an adversary needs a physical access to a pair of identical (similar) devices that we call a profiling device and a target device. Basically, these attacks consist of two phases (profiling and attack phase). In the first phase, the adversary analyzes the profiling device in order to approximate the leakage behavior and in the second phase, the adversary attacks the target device. Typical examples are the template-based attack (TA) [5], [6] and Stochastic Approach (SA) [7], [8]. The practical aspects of template attacks (TA) have been discussed in [9], [10]. The profiling phase of TA was improved in [6], [11], [12]. Most crucial step during these profiling attacks lies in selecting of the interesting points. Various techniques are used to localize interesting points that provide information about data processed e.g. Normalized Inter-Class Variance (NICV) [13], Sum Of Squared pairwise Differences (SOSD) [14], Sum Of Squared pairwise T-differences (SOST) [14], Principal Components Analysis (PCA) [11], [15] or Pearson Correlation (CPA) [3]. By contrast, the non-profiling attacks are one-phase attacks that perform the attack directly on the target device (it represents a more realistic scenario in practice). The attacker measures a set of power traces for the known plain text and compares these power traces with hypothetical power consumption that was calculated based on every secret key hypotheses and on power consumption model [4], [16], [17]. Only the correct key hypothesis shows dependency of the hypothetical power consumption calculated and the power consumption measured. DPA based on the correlation coefficient and on Hamming weight power consumption mode represents a typical example of non-profiled attacks that are aimed on smart card implementations [3], [18].

In order to prevent power analysis attacks, one can implement some of the countermeasure techniques. The goal of every countermeasure is to create the power consumption of a cryptographic device independent of intermediate values that are currently processed. Generally the countermeasure techniques are divided into two basic groups, *hiding* and *masking*. In the masking approach, each intermediate value is concealed by a random value that is called mask. Various masking methods of the AES algorithm have been already proposed [19–22]. By contrast, hiding tries to break the link between the power consumption and the data values processed [3]. It is clear, that the implementation of countermeasures brings overhead in terms of memory and time therefore researchers have started to look for the lightweight possibilities. One of these lightweight countermeasures is Rotating Sbox Masking that is a type of Low-Entropy Masking Scheme [23–25]. The main idea is based on the usage of the precomputed table look-ups [26] and at the same time the overhead is reducing by carefully choosing the limited mask set [27]. This essentially allows to reuse S-boxes and reduce the computation of mask compensation because only 16 possible masks are applied. The set of chosen mask can be a public parameter however this set should be shifted by a random *offset* before each encryption. We refer interested readers to works [23], [25], [28] where more details of RSM and its security analysis are provided. RSM has been studied by researchers worldwide under the framework of DPA Contest V4 [29]. DPA Contest is an international framework that allows researchers to compare their power analysis attacks under the same conditions and is organized by Telecom ParisTech French University.

To complete the introduction about power analysis, we note that an adversary can bypass masking techniques using several intermediate values to calculate hypothesis. These types of attacks are called higher-order DPA attacks [30], [31]. Higher-order DPA attacks exploit the joint leakage of several intermediate values that occur inside the cryptographic device. In the paper, we do not take this possibility into account.

Machine learning (ML) as a scientific discipline explores the construction of algorithms that can improve their performance based on previous experiences or trainings [32]. Most of the machine learning problems deal with the classification of various input data. In general, machine learning approaches can be classified as supervised [33] and unsupervised learning [34]. Intuitively, in supervised learning, the machine is presented with a set of training data with the label and the goal is to determine the general function that associates the data with the label. In unsupervised learning, the machine is presented with a set of unlabeled data, and the machine tries to determine the hidden structure of the data.

From the description above, one can clearly see an analogy between machine learning approaches and power analysis attacks. More specifically, profiling attacks are a supervised learning problem, where ML techniques are used for a model creation of the target device. Generally, the model created is based on the multivariate normal distribution in the power analysis. This fact is based on the following simple analysis of a power trace: one can analyze power consumption measured by looking at a single point of a power trace (we look at the power consumption of a cryptographic device at a fixed moment of time). For this point, we can determine the probability distribution that is dependent on the processed data. Generally, it is difficult to make a statement about the data dependency of the power consumption of cryptographic devices. However, for most cryptographic devices, it is valid to approximate the distribution of the data dependency of the power consumption by a normal distribution. Moreover, power consumption of cryptographic devices is mostly proportional to the Hamming weight[1] or the Hamming distance[2] of data processed. In these cases, the distribution is composed of nine normal distributions with different mean and the standard deviation is approximately the same. In order to consider the correlation between more points in the power trace, it is necessary to model a power trace measured as a multivariate normal distribution which constitutes a generalization of the normal distribution to higher dimensions. We refer the book [3] where authors realized the complete analysis of statistical characteristics of power traces.

On the other hand, non-profiling attacks can be seen as an unsupervised learning. Instead of statistical methods, one can apply ML in order to find the desired structures in the data. In this paper, we do not take into account this application of ML.

## 1.1 Related Work

In the field of power analysis, the possibility of using neural networks was first published in [35]. Naturally, this work was followed by the other authors, e.g. [36], who dealt with the classification of individual power prints. These works are mostly oriented towards reverse engineering based on power print classification. Yang et al. [37] proposed MLP in order to create a power consumption model of a cryptographic device in CPA. Lerman et al. [38], [39] compared a template attack with a binary machine learning approach, based on non-parametric methods.

Hospodar et al. [40], [41] analyzed the SVM on a software implementation of a block cipher. Heuser et al. [42]

[1]Typical for smart cards.

[2]Typical for micro controllers.

created the general description of the SVM attack and compared this approach with the template attack. In 2013, Bartkewitz [43] applied a multi-class machine learning model which improves the attack success rate with respect to the binary approach. Moreover, they used (linear) SVM as a preprocessing tool for feature selection, similar as Brank [44]. Recently, Lerman et al. [45] proposed a machine learning approach that takes into account the temporal dependencies between power values. This method improves the success rate of an attack in a low signal-to-noise ratio with respect to classification methods. Another SVM-based attack was presented in [46] where the authors used SVM to recover the secret key (bit by bit) by exploiting the leakage in the key permutation round.

Lerman et al. [47] presented a machine learning attack against a masking countermeasure, using the dataset of the DPA Contest v4. The method of power analysis based on a multi-layer perceptron was first presented in [48]. In this work, the authors used a neural network directly for the classification of the AES secret key. In [49], this MLP approach was optimized by using the preprocessing of the power traces measured. Lerman et al. [50] introduced semi-supervised a Template Attack, that combines supervised and unsupervised learning. The method was confirmed by the experiments on an 8-bit microcontroller and by a comparison to a template attack. The authors proposed an unsupervised learning approach for PA in [51] aimed on DES algorithm. Heyszlet et al. [52] introduced unsupervised cluster classification algorithm $k$-means to attack cryptographic exponentiation of public key cryptographic system and recover secret exponents without any prior profiling. Note that the algorithm $k$-means should not be confused with $k$-nearest neighbor algorithm. Zhanget et al. [53] proposed DPA attack based on the correlation coefficient using Genetic Algorithm. Perin et al. [54] presented the attack based on clustering algorithm that attacks the randomized exponentiation of RSA algorithm. In work [55], the $k$-NN algorithm was briefly mentioned as a possible mutual information estimator.

At the end of 2014, Dirmanto realized a small but concise overview of machine learning approaches in power analysis [56]. The survey paper summarizes the main theoretical aspects [56]. During the CHES 2015 conference, Whitnall presented unsupervised clustering algorithm (K-means clustering) in order to recover nominal power model [57]. The model was used to key recovery attack, with minimal requirements in the profiling phase and moreover the approach was effective and robust across an extensive set of distortions.

## 1.2 Contribution

In previous works, individual machine learning (ML) approaches are compared mostly with the template attack or the stochastic attack (SA) [40], [41]. ML approaches have not been compared yet. The work [47] can be mentioned as an exception, where SVM and RF are compared with the TA and the SA. In this article, we try to make an extensive comparison of machine learning algorithms in PA. We focus only on the usage of the individual ML algorithms in profiling attacks where ML techniques are used for a model creation of the target device. We do not consider other possible application such as structure searching, preprocessing or feature selection.

For our research, we implemented a verification program that always chooses the optimal settings of the individual ML models in order to obtain the best classification accuracy. Our research was based on three datasets, the first dataset containing the power traces of an unprotected AES implementation where we classify one byte of the secret key. The second and third datasets were independently prepared from public datasets of power traces corresponding to the masked AES implementation (DPA Contest v4 [29]) where we classify the secret *offset*. We decided to use the first order success rate as a metric of the comparison because our two datasets were focused on mask classification and Guessing entropy is not suitable in this case[3]. Furthermore, we compared every ML approaches with template based attack. In this research, we wanted to answer particularly these questions:

- Which ML algorithm is the most suitable for profiling PA attacks?
- Are there any generally appropriate settings of the ML algorithms that can be used by the potential attacker for PA attacks?
- How big is the influence of the number of power traces and interesting points on the classification results of individual ML algorithms?

Nowadays, the method using the SVM is considered to be the most effective from machine learning algorithms in the power analysis. In many concrete attacks, in which an adversary has only a limited number of power traces available, the SVM is better in comparison with the classical template attack or the stochastic attack. Based on the results obtained, we propose a power analysis method based on the $k$-NN algorithm as the most effective method. Even there is no "intelligence", the algorithm shows great application potential in PA, because the usage of the algorithm provides some advantages for the attacker in comparison with the other machine learning approaches and classic power analysis attack. Moreover, we describe the general scheme of this method in profiling power analysis attacks.

[3]It is usually more suitable to use *Guessing Entropy* as a metric to compare different key recovery side-channel attack implementations [58], but we focused on *offset* revelation using two datasets and on secret key recovery using one dataset. Therefore we used a confusion matrix and success rate, which is also often used during profiling PA attacks [59].

## 2. *k*-Nearest Neighbors

In the previous section, we have already provided relevant references that deal with the well-known ML approaches in power analysis attack (such as SVM, MLP and RF). Therefore, the following text focuses only on the description of the approach proposed based on $k$-NN algorithm. We provide the general scheme of this method in profiling power analysis attacks.

**Preliminaries:** a learning set $\mathbf{Y}$ (sometimes denoted as a training set) and a test set $\mathbf{X}$ with $n$ and $m$ instance represents power traces measured in the context of the power analysis. Each instance $\mathbf{y}_i$ where $i = 1, \ldots, n$ and $\mathbf{x}_j$ where $j = 1, \ldots, m$ in the learning and training set contains one assignment (a class label that determines which class the concrete instant belongs to) and several attributes $\mathbf{y}_i = y_1, \ldots, y_N$, $\mathbf{x}_j = x_1, \ldots, x_N$ (features or observed variables). These attributes represent the interesting points of power traces in time (samples). The learning set is used in the profiling phase of the profiled attack and the test set is used during the attack phase. In profiling power analysis attack, the label represents the desired byte value of secret key i.e. together 256 possible variants (0 to 255). From the perspective of ML, we can see this problem as multiclass classification where ML classifies the instance into 256 possible classes. The second method that is often used is to transfer this problem into the multi-label classification. The multi-label classification represents the problem of finding a model that maps inputs $\mathbf{x}_j$ to binary output vectors $\mathbf{y}_i$. There are two main methods for tackling the multi-label classification problem: problem transformation methods and algorithm adaptation methods [46], [60], [61]. Problem transformation methods transform the multi-label problem into a set of binary classification (two classes 0 or 1) that can be realized by single-class classifiers (such as binary relevance or label powerset). Algorithm adaptation methods adapt the algorithms to directly perform multi-label classification (Multilabel Neural Networks).

In machine learning, the k-Nearest Neighbors algorithm is a non-parametric method used for classification and belongs to the simplest machine learning algorithms [62]. The training phase of the algorithm consists only of storing the learning set into the memory. In the classification phase, $k$ is a user-defined constant (typically small), and a point of the test set is classified by assigning the label which is most frequent among the $k$ training samples nearest to that classified point. If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor.

A typical example of $k$-NN classification is shown in Fig. 1. The test sample denoted as a gray square should be classified either to the first class of the white stars or to the second class of the black circles. If the classification process takes into account the three nearest points $k = 3$, the test sample is assigned to the second class, because there are 2 stars and only 1 circle inside the selected area (solid line inner circle). If $k = 5$, test sample is naturally assigned to the first

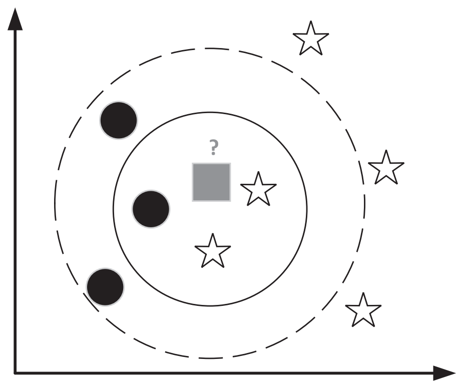

**Fig. 1.** Example of $k$-NN classification.

class because 3 black circles and 2 stars are in the selected area (dashed line outer circle).

A commonly used distance metric for continuous variables are defined as:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}, \qquad \text{Euclidean}, \tag{1}$$

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} |x_i - y_i|, \qquad \text{Manhattan}, \tag{2}$$

where $i$ represents a number of attributes in the learning set. The overlap metric or Hamming distance are other possible metrics for discrete variables. The best choice of $k$, that strongly depends on the learning set, is very important. Generally, large $k$ reduces the effect of the noise on the classification but makes the boundaries between the classes less distinct. Suitable $k$ can be selected by the various heuristic techniques, for example the hyperparameter optimization [63]. The following text describes the power analysis method based on $k$-NN.

**Profiling Phase**

In the attack based on $k$-NN, we assume that we can characterize the profiling device by labeling of measured data. One can implement a certain part of the cryptographic algorithm and execute the sequence of instructions on a profiling device with different data $d_i$ and different key values $k_j$, and record the power consumption. After measuring $n$ power traces, we create the matrix $\mathbf{Y}_n$ that contains power traces corresponding to the pair $(d_i, k_j)$. According to the key value, we add label in to the matrix $\mathbf{Y}_n$. In case of byte classification, the label can be expressed by four columns where every row represents a class using the binary expression 00000000 to 11111111 (every possible byte value from 0 to 255). The matrix $\mathbf{Y}_n$ represents a learning set which is stored into the memory.

**Attack Phase**

During the attack phase, the adversary uses the stored learning set together with the measured power trace from the target device (denoted as $\mathbf{t} = [x_1, \ldots, x_N]$) to determine the secret key value. Let's assume that for our $k$-NN algorithm we chose the following parameters: $k = 5$ and Euclidean distance. The classification takes three steps:

- at the beginning, the algorithm calculates Euclidean distances of all stored training vectors $\mathbf{y}_n$ to vector $\mathbf{t}$:

$$d(\mathbf{x}, \mathbf{y}_n) = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}. \tag{3}$$

- In the second step, 5 closest training points are found according to the distances calculated.
- In the last step, the class is selected based on the majority vote.

The result of this classification is the most probable class based on training set $\mathbf{Y}$. Since each training instance $\mathbf{y}_n$ is associated with a secret key value, the adversary obtains the information about the secret key stored in the target device.

**Discussion of Simple Application of $k$-NN in Power Analysis**

In the following text, we provide a simple example of the attack based on $k$-NN using the real data of power consumption (we used only two dimension ie two interesting points were selected) in order to demonstrate the suitability and simplicity of approach proposed.

Let's assume that the adversary wants to determine a Hamming Weight (HW) of processing data or secret key value. During the profiling phase, the adversary measures 2 560 power traces, 10 for every byte value (it means that 10 power traces corresponding to the HW = 0, 80 power traces corresponding to the HW = 1 and so on). In power traces, the adversary chooses two interesting points that leak information about HW. The profiling phase is finished by storing the points into the memory of a computer. Figure 2 shows scatter plot of two chosen interesting points. The division of two dimensional space into nine groups according to the HW is clearly visible. We can approximate this data dependency of the power consumption by a two variable normal distribution (we refer interested readers to consult statistical analysis with book [3]). In these cases, the distribution is composed of nine normal distributions.

In the attack phase, the adversary measures power trace from the target device and puts same interesting points to the $k$-NN algorithm. Figure 3 shows the process of classification for unknown point that is marked with a black star and for parameter $k = 5$. All five nearest neighbors points belong to the distribution marked with blue color that corresponding to the HW= 0. The adversary obtains the desired information that the HW of processed data is 0. Similarly, the adversary can continue with additional power traces in order to reveal desire sensitive information.

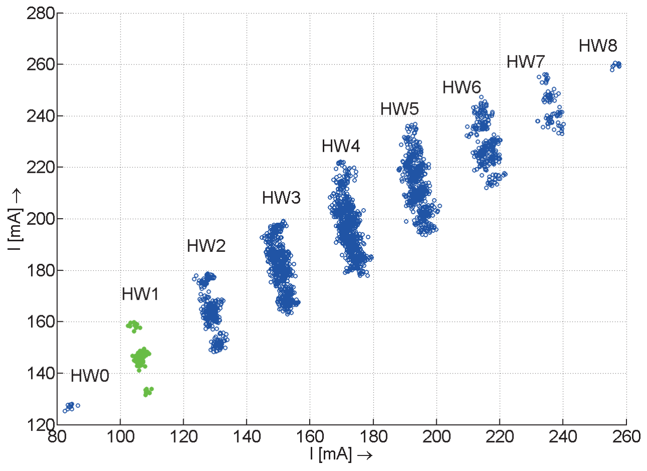

**Fig. 2.** Scatter plot of two interesting points that leak HW.

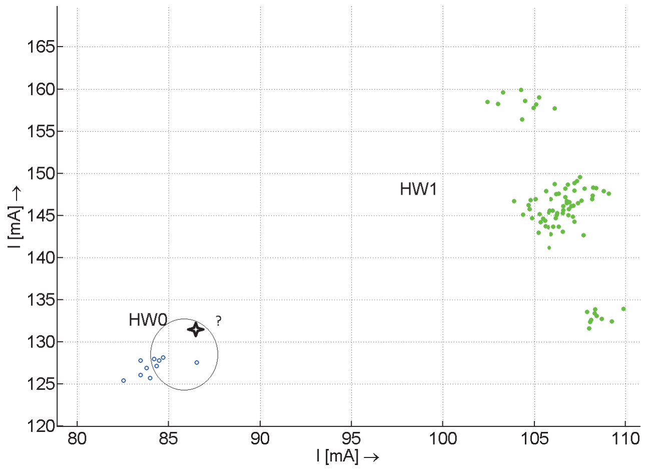

**Fig. 3.** Detail of scatter plot.

It is obvious, if we focus on two points of the power consumption at fixed time in our example, and realize measurement of power consumption repetitively for constant data than points measured will appear more or less on the same are (group). Similar situation occurs for different data proceed by cryptographic device therefore the points are in clusters. Widely known fact is that $k$-NN algorithm is one of many algorithms that is robust, simple and suitable for classification problem. Using this simple example, we wanted to demonstrate the classification problem during the power analysis attacks and simple application of ML algorithms.

## 3. Settings of Experiments

The following text summarizes the most important facts about the experimental setup and the verification program (implemented attacks). We created three datasets in order to test chosen machine learning approaches. Based on the state of the art, we chose SVM, MLP, $k$-NN, DT (Decision trees), RF and LDA (Linear Discriminant Analysis).

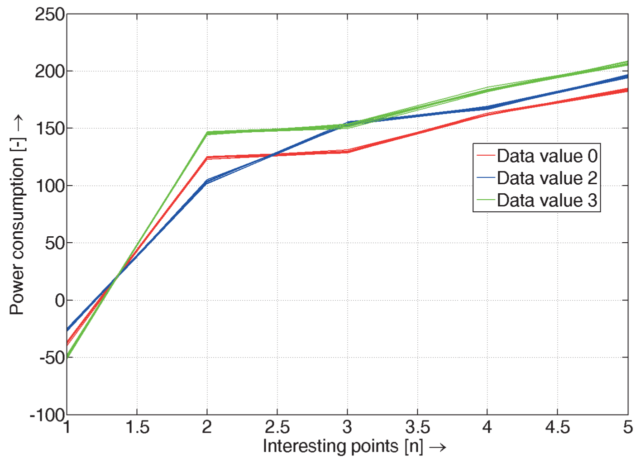

**Fig. 4.** Example of power traces selected for DS1.

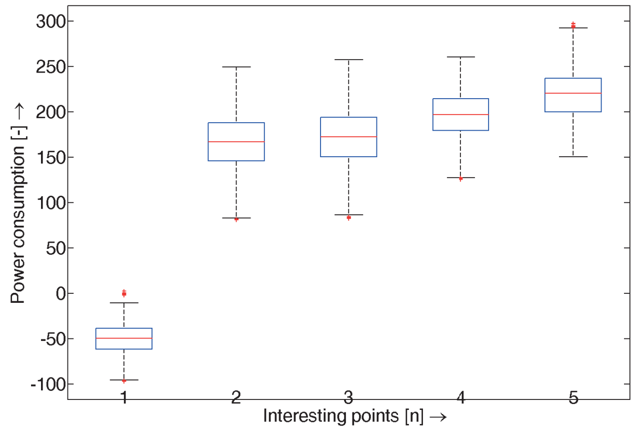

**Fig. 5.** Characteristics of DS1.

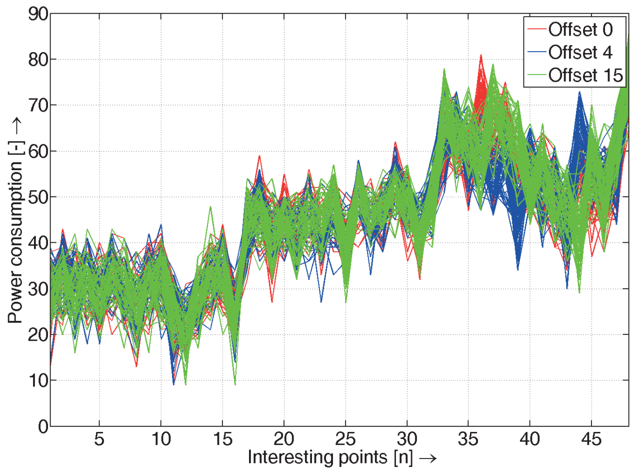

**Fig. 6.** Example of power traces selected for DS2.

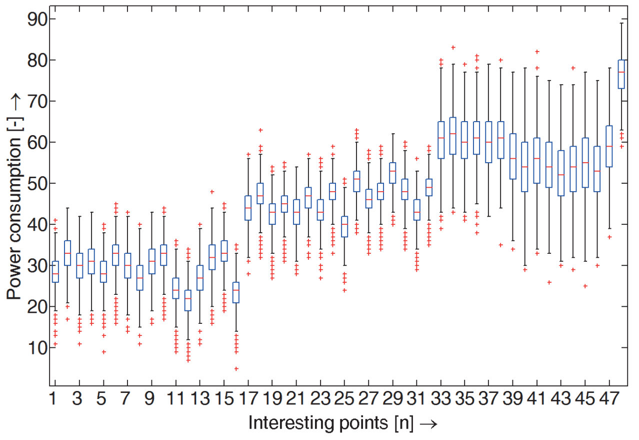

**Fig. 7.** Characteristics of DS2.

**The first dataset (DS1)** is focused only on the first byte classification of the secret key. Dataset is prepared from power traces of unprotected AES-128 implementation in our testbed. The cryptographic module was represented by the PIC 8-bit micro controller, and for the power consumption measurement we used the CT-6 current probe and the Tektronix DPO-4032 digital oscilloscope. We used standard operating conditions with 5 V power supply. Stored power traces had 100 000 of samples and covered the `AddRoundKey` and `SubBytes` operations in the initialization phase of the algorithm. Our implementation was realized in the assembly language and the executed instruction of examined operation were exactly the same for every key byte (identical power prints). Therefore it was possible to use the place, where the first byte is processed, in order to create a model with which it was possible to determine the whole secret key byte by byte. We verified this assumption experimentally and it is naturally conditioned by the excellent synchronization of measured power traces. Finally, we chose 5 interesting points based on standard CPA method (note, we used well know CPA method to localize interesting points in our whole research). Every of our chosen interesting points leaked information about Hamming weight of the processed data. We chose points that had a distance at least one clock cycle from each other. This restriction for having IP not too close avoids the numerical problems when inverting the covariance matrix during the template based attack.

An example of power traces selected are depicted in Fig. 4. Overall characteristics of IP selected are depicted in Fig. 5 using the box plot. On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. It can be observed that the first dataset does not include almost any noise. Consequently, our first dataset represents a matrix $2\,560 \times 13$ where the last 8 values are labels. Each label is expressed by four columns where every row represents a class using the binary expression from 00000000 to 11111111.

**The second dataset (DS2)** is focused on the mask classification and consists of 1 000 power traces. DS2 is prepared from electromagnetic traces that are freely available on the website of DPA Contest v4 [29]. The masked block-cipher AES-256 in encryption mode without any mode of operation is implemented on target cryptographic device Atmel

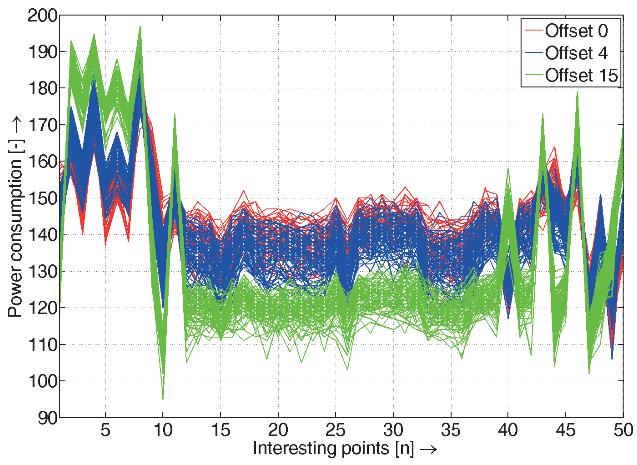

**Fig. 8.** Example of power traces selected for DS3.

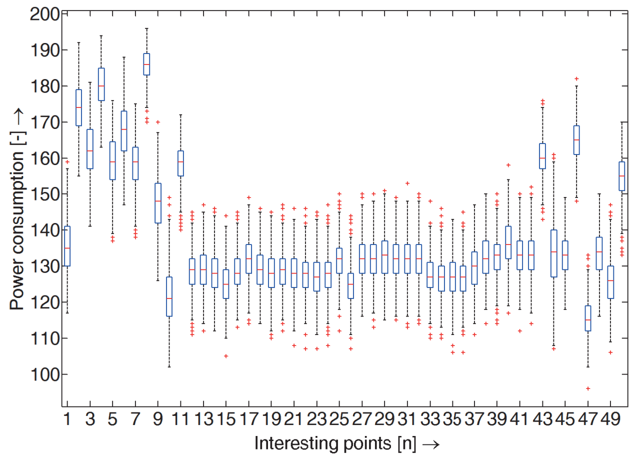

**Fig. 9.** Characteristics of DS3.

ATMega-163-based smart card. The implemented masking scheme is a variant of the Rotating Sbox Masking [23], [64]. According to the authors, this masking scheme keeps performance and complexity close to the unprotected scheme and is resistant to several side-channel attacks. Sixteen masks are public information that are incorporated in the computation of the algorithm. *offset* value, which is drawn randomly at the beginning of computation, is a secret value. Mask values are rotating according to the *offset* value [23], [64]. Each stored trace has 435 002 samples associated to the same secret key and corresponds to the first and to the beginning of the second round of AES algorithm. For DS2, we chose only the points that are the most correlated with the secret *offset* value.

We realized classical CPA for operation `Plaintext blinding` dependent on *offset* value in order to locate the interesting points. We chose the 3 highest correlated points for every mask value, together 48 interesting points were selected. In other words, DS2 represents a matrix 1 000 × 52 where the last four values are labels. In our case, the label value corresponds with the *offset* value 0 to 15 (sixteen possible variants). An example of power traces selected are depicted in Fig. 6. The overall characteristics of the interesting points selected are depicted in Fig. 7 using the box plot.

**The third dataset (DS3)** was created by Liran Lerman during preparation of the attack in DPA Contest v4 [47]. This DS is focused on the mask classification and we used first 1 000 traces of 1 500 available. The author chose 50 interesting points according to the computed Pearson correlation between each instance of 1 500 traces and the *offset* value. In other words, our DS3 represents a matrix 1 000×54 where the last four values are labels. Again, the label value corresponds with the *offset* value 0 to 15 (sixteen possible variants). We refer the work [47] for more information about the original dataset. An example of power traces selected is depicted in Fig. 8. The overall characteristics of interesting points selected are depicted in Fig. 9.

A well known fact is that noise always poses the problem during the power consumption measurement. Every stored power trace from DS1 was calculated as an average power trace from ten power traces measured using the digital oscilloscope to reduce electronic noise. We refer the website [29] to consult level of noise in DS2 and DS3.

## 3.1 Implemented Program

Figure 10 shows the main principle of the verification algorithm implemented[4]. The main part of the implemented program is the block denoted as **Optimize Parameters** This block finds the optimal values of the selected parameters for the tested machine learning algorithms. In other words, it executes each model for all combinations of user selected values of the parameters and then delivers the optimal parameter values as a result. Selected specific parameters are described in more details in the next section. The second important block of the program is the **Cross-validation**. Cross-validation (CV) is a standard statistical method to estimate the generalization error of a predictive model. In $l$-fold cross-validation, a training set is divided into $l$ equal-sized subsets. Then a model is trained using the other $(l-1)$ subsets and its performance is evaluated on the current subset. This procedure is repeated for each subset. In other words, each subset is used for testing exactly once. The result of the cross-validation is the average of the performances obtained from $l$ rounds.

In our verification program, we used typical 10-fold cross-validation. We repeated CV four or eight times in the **Loop**, because we created four or eight models for individual bit classification depending on the dataset. In other words, in our program we chose the multi-label classification where a model maps inputs $\mathbf{x}_j$ to binary outputs vectors $\mathbf{y}_i$ using single-class classifiers. The verification program returned two output values: the best parameters for learning models and the obtained accuracy using these parameters. The accuracy was described using the typical confusion matrix.

[4]We use Rapid Miner for implementation [65].

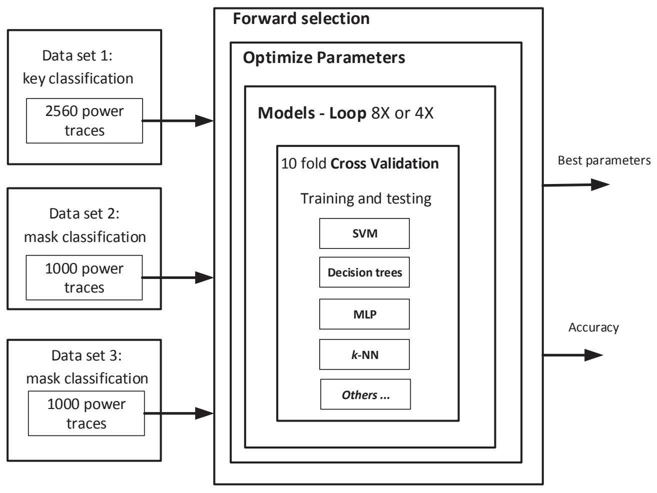

**Fig. 10.** Block scheme of our testing program.

The original implemented program contained the block called **Forward selection** that selected individual attributes of DSs. In each round, this block can add attribute and the performance is estimated using the inner operators, e.g. a cross-validation. This configuration allows us to get the best result of machine learning algorithms depending on individual parameters setting and attributes selected. In this way, we tested the influence of number and the combination of selected attributes on the classification results.

## 3.2 Selected Parameters

It is obvious, that time required to solve the program implemented strongly depends on the number of selected parameters of individual ML approaches. One can test an infinite number of parameters in theory, but it has no sense in practice. For example, it is really unnecessary to test $k$-NN algorithm for $k > 11$, because the results are worse and the advantages of the algorithm are reduced (based on long years practical experience with ML approaches). For these reasons, we chose only limited number of parameters that are relevant and important for testing individual ML algorithms. Selection of parameters was realized based on our experience and knowledge with ML.

In order to test **SVM** approach, we chose the complexity parameter from 0 to 50, epsilon from 0.1 to 1 and type of kernel radial, linear, polynomial, together it was 3 333 of combinations. We selected three parameters: the depth from 1 to 100, the confidence from $1.0\,\mathrm{e}^{-7}$ to 0.5 and criterion set to gain ratio, information gain, gini index and accuracy (484 combination) to test **decision trees (DT)**. The **MLP** approach was tested by the following parameters: one hidden layer, type of the activation function, a number of training cycle from 1 to 1 000, learning rate, neural network momentum both from 0 to 1 and normalization true or false (5 324 combinations). During the testing of the ***k*-NN** algorithm, we selected different types of metrics: Euclidean, Camberra, Manhattan and Chebychev distance, Correlation, Cosine, Dice, MaxProduct, Overlap and Jaccard similarity, parameter $k = 1, 3, \ldots, 11$ and weighted vote (true or false). Together, only 132 of combinations were tested. Testing of **Random forest (RF)** model involved the parameters: the depth from 1 to 100, the confidence $1.0\,\mathrm{e}^{-7}$ to 0.5, criterion set to gain ratio, information gain, gini index and accuracy and last parameter was a number of trees from 1 to 500 (5 324 combinations). As a last tested machine learning algorithm, we involved linear discriminant analysis **LDA** in default setting.

In order to complete our comparison, we implemented the classical **Template attack**. We were interested in comparison of effective template attack based on pooled covariance matrix [6], therefore we calculated the pool covariance matrix as an average value of all covariance matrices and we calculated the probability density function (equation (4)) with this matrix. Implementations of template attacks were done according to the equation (4):

$$p(\mathbf{t}; (\mathbf{m}, \mathbf{C})_{d_i,k_j}) = \frac{\exp(-\frac{1}{2} \cdot (\mathbf{t} - \mathbf{m})' \cdot \mathbf{C}^{-1} \cdot (\mathbf{t} - \mathbf{m}))}{\sqrt{(2 \cdot \pi)^{\mathrm{NI}} \cdot \det(\mathbf{C})}} \quad (4)$$

where $(\mathbf{m}, \mathbf{C})$ represents templates prepared in profiling phase based on multivariate normal distribution that is fully defined by a mean vector and a covariance matrix. Measured power trace from the target device is denoted as **t** and NI is the number of interesting points. In the following text, classical template and template attack based on the pooled covariance matrix are denoted as $T_{\mathrm{cls}}$ and $T_{\mathrm{pool}}$ sequentially. In the first experiment, we did not include a reduced template attack, because if the adversary does not consider the covariance matrix, he loses information about the relationship between the interesting points. All template attack implementations were realized in the Matlab environment.

## 4. Results Evaluation

The implemented program provided two outputs: accuracy and best parameters selected for individual ML algorithm. In order to calculate accuracy, it was used a typical confusion matrix. Interested readers can consult [66] to obtain additional explanations about performance measurements for classification, e.g. *confusion matrix*, *precision*, *recall*. Examples of confusion matrices of DS1 classification for SVM-rbf and $k$-NN algorithm are shown in Tabs. 1 and 2. In the confusion matrix, accuracy is the arithmetic mean of the accuracy obtained from the $8 \times 10$ cross-validation for individual models. The $\sigma$ value represents their standard deviation. The value denoted as *mikro* is actually the accuracy computed from the confusion matrix. In other words, it is the success rate calculated for all of the 20 480 experiments carried out in DS1. It is not possible to present every obtained confusion matrices in this paper, therefore we present the results based on *mikro* value of success rate.

| Accuracy: mikro: | 96.15 % **94.38 %** | $\sigma$= 4.45 % |
|---|---|---|
| bit value: | 0 | 1 |
| pred. 0: | 9 640 | 551 |
| pred. 1: | 600 | 9 689 |

**Tab. 1.** Example of confusion matrix for SVM-rbf based on DS1, parameters selected: rbf kernel, C=50, epsilon 0.46.

| Accuracy: mikro: | 99.58 % **99.20 %** | $\sigma$= 0.63 % |
|---|---|---|
| bit value: | 0 | 1 |
| pred. 0: | 10 161 (TN) | 84 (FN) |
| pred. 1: | 79 (FP) | 10 156 (TP) |

**Tab. 2.** Example of confusion matrix for k-NN based on DS1, parameters selected: Euclidian distance, weighted vote false and k=5.

In our **first experiment**, we verified that the influence of the block **Forward selection** on resulting success rate is very low. It is clear, if the selection of interesting points is done in a correct way, the algorithm chooses always maximum of attributes. This conclusion is natural and not surprising, because selection of the interesting points from power traces is crucially important during the profiled attacks, and we used well know and verified CPA method in order to localize interesting points during dataset preparation. Based on the results obtained, we skipped this block and always chose the maximum of attributes in the following experiments.

In the **second experiment**, we were searching for the best success rate corresponding with the parameters of selected machine learning algorithm on our three datasets. In this way, we got the best possible success rate for machine learning algorithm and we could compare machine learning algorithms according to the highest value. Table 3 summarizes the success rate obtained in percentage. The penultimate rows provide the average value of the success rate calculated from three values obtained and the last rows provide the differences between average value and maximal obtained average value.

From the results, one can confirm that differences between optimized ML algorithms are negligible. Note, that the SVM with the rbf kernel had the best success rate of all SVM kernels for all datasets. The algorithm *k*-NN classified the DS1 with the highest success rate from all ML tested (DS1 has the lowest value of noise). The SVM-rbf was the best ML classifier of the DS2 and the DS3. Generally, the template attack based on the pooled covariance matrix[5] was the best in average success rate based on all tested datasets, but if we focus on the ML, the SVM-rbf together with *k*-NN were the best with almost the same success rate. The difference was only 0.45 % and we can consider the difference negligible taking into account the number of the experiments (together 28 480 of individual bit classification). MLP was the third best algorithm with only 0.57 % difference from SVM-rbf. We can conclude that the SVM-rbf, MLP and *k*-NN are the most suitable candidates for profiling power analysis attacks.

|  | *k*-NN | SVM linear | SVM rbf | SVM poly | DT |
|---|---|---|---|---|---|
| DS1 | 99.20 | 85.57 | 94.38 | 86.93 | 94.10 |
| DS2 | 97.65 | 92.65 | 99.02 | 97.92 | 83.00 |
| DS3 | 88.05 | 89.50 | 92.95 | 90.12 | 79.85 |
| $\phi$ | 94.97 | 89.24 | 95.45 | 91.66 | 85.65 |
| $\Delta$ | 0.64 | 6.37 | 0.16 | 3.95 | 9.96 |
|  | RF | MLP | LDA | $T_{\text{cls}}$ | $T_{\text{pool}}$ |
| DS1 | 90.24 | 93.52 | 85.17 | 98.44 | 98.83 |
| DS2 | 85.48 | 98.92 | 93.08 | 95.60 | 99.60 |
| DS3 | 74.95 | 92.20 | 89.45 | 47.00 | 88.40 |
| $\phi$ | 83.56 | 94.88 | 89.23 | 80.35 | 95.61 |
| $\Delta$ | 12.05 | 0.73 | 6.38 | 15.26 | 0.00 |

**Tab. 3.** The highest possible success rate of tested ML algorithms [%].

The following text summarizes the parameters selected of individual machine learning algorithms during the second experiment.

**Parameters selected for DS1:**

- MLP: training cycle 475, momentum 0.0, learning rate 0.4, normalization true.
- SVM-linear: C=0.5, epsilon 0.001.
- SVM-rbf: C=50, epsilon 0.46.
- SVM-poly: C=2.5, epsilon 0.46.
- *k*-NN: $k = 5$, weighted vote false, Euclidian distance.
- DT: maximal depth 39, confidence $1.0\,\mathrm{e}^{-7}$, no pruning true, criterion gini index.
- RF: number of trees 300, maximal depth 29, criterion gini index.

**Parameters selected for DS2:**

- MLP: training cycle 720, momentum 0.0, learning rate 0.3, normalization true.
- SVM-linear: C=17.5, epsilon 0.82.
- SVM-rbf: C=45.0, epsilon 0.28.
- SVM-poly: C=12.5, epsilon 0.28.
- *k*-NN: $k = 5$, weighted vote false, numerical measure OverlapSimilarity, kernel type multiquadric.
- DT: maximal depth 31, confidence 0.4, no pruning true, criterion gini index.
- RF: number of trees 500, maximal depth 70, criterion gini index.

**Parameters selected for DS3:**

- MLP: training cycle 640, momentum 0.6, learning rate 0.3, normalization true.
- SVM-linear: C=7.5, epsilon 0.91.

[5]Results of template based attack are informative, because template attacks were aimed at the whole byte classification.

- SVM-rbf: C=45.0, epsilon 0.28.
- SVM-poly: C=4.0, epsilon 0.37.
- *k*-NN: $k = 5$, weighted vote false, numerical measure OverlapSimilarity, kernel type multiquadric.
- DT: maximal depth 71, confidence 0.05, no pruning false, criterion gini index.
- RF: number of trees 500, maximal depth 29, criterion gini index.

We can recognize some similarities of the parameters selected. Definitely, good choices for an attacker can be either MLP with one hidden layer and normalization true, SVM-rbf with the parameters C = 45 and epsilon 0.28 or we suggest the *k*-NN with $k = 5$ and Euclidian distance.

Practically, we can optimize every ML algorithm (using individual parameter settings) to get almost the identical classification results. The biggest difference between the tested algorithms lies in the required time that is needed to find the best setting of the concrete ML algorithm. For example, finding the best parameters of the SVM algorithm with poly kernel takes approximately eight days using parameters selected and the DS1. The difference is enormous in comparison with 6 minutes and 35 seconds that was necessary for *k*-NN optimization for the same DS1. In order to demonstrate this feature, Tab. 4 summarizes the time consumption of one executed 10-cross validation for implemented ML algorithms.

The time required to calculate one 10-cross validation for *k*-NN was less than 1 s and 320 s for SVM-rbf. Naturally, the attacker has to calculate so many numbers of CV as the number of the tested parameters, therefore the time needed is directly proportional to the number of the tested parameters and learning time. In our case, we tested only 132 combinations of the parameters for *k*-NN, but 1 111 combination for SVM with poly kernel. The algorithm *k*-NN is really easy, therefore it is not necessary to test many parameters and the algorithm does not include learning phase. These are the main reasons why in terms of time consumption, the *k*-NN algorithm provides the best performance in our implementation[6].

Our **fourth experiment** examines the classification success rate based on the number of power traces. For this purpose, we prepared new datasets from the original three DSs that differed in the number of power traces. From the first DS1, we created 10 datasets each containing one power trace more successively that corresponds to each key value. In other words, datasets created have from 256 to 2 560 of power traces with step 256. From DS2 and DS3, we created again 10 datasets containing sequentially 100 to 1 000 power traces with step 100. Data prepared were classified successively using the implemented program. The obtained results are depicted in Figs. 11, 12 and 13.

[6]Note, that our test was performed on datasets containing 1 000 and 2 560 of power traces.

| | *k*-NN | SVM linear | SVM rbf | SVM poly | DT |
|---|---|---|---|---|---|
| DS1 | 0.2 | 185 | 320 | 2 075 | 18 |
| DS2 | 0.3 | 45 | 4 | 4 | 20 |
| DS3 | 0.3 | 10 | 12 | 315 | 26 |
| | RF | MLP | LDA | $T_{\text{cls}}$ | $T_{\text{pool}}$ |
| DS1 | 5 750 | 320 | 1 | 530 | 530 |
| DS2 | 890 | 275 | 1 | 30 | 30 |
| DS3 | 750 | 90 | 1 | 27 | 27 |

**Tab. 4.** Time consumption of 10-cross validation [s].

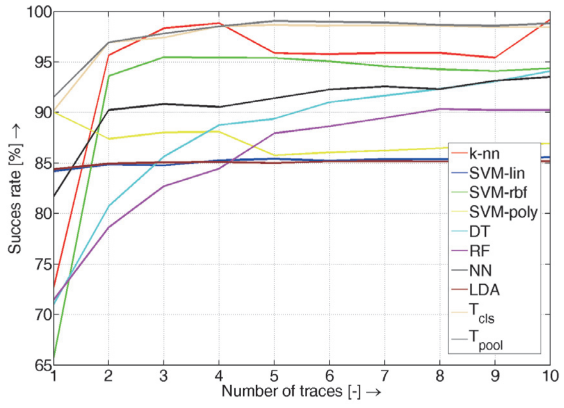

**Fig. 11.** Classification results DS1.

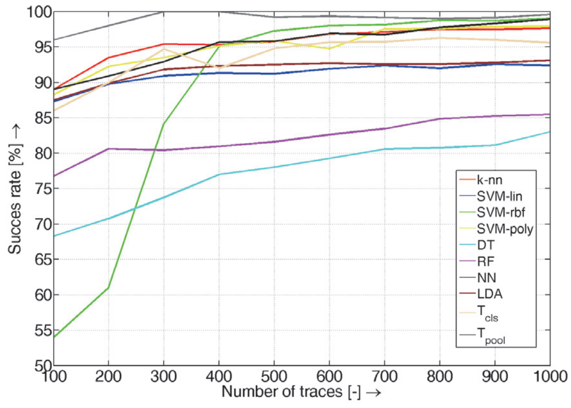

**Fig. 12.** Classification results DS2.

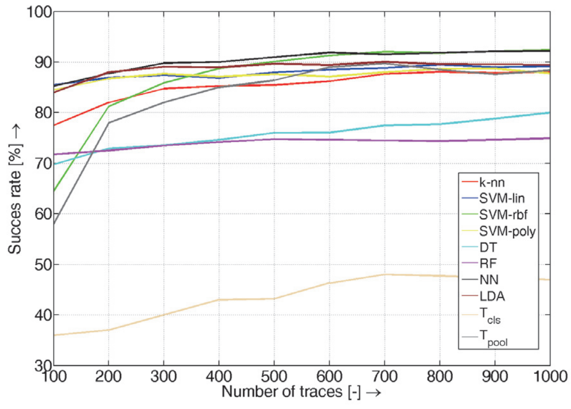

**Fig. 13.** Classification results DS3.

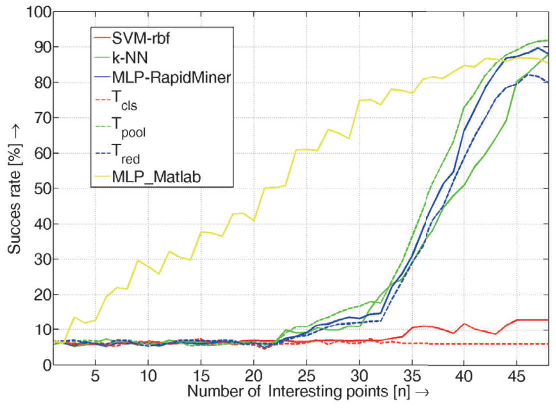

**Fig. 14.** Success rate of the secret *offset* revelation based on 100 power traces of DS2.

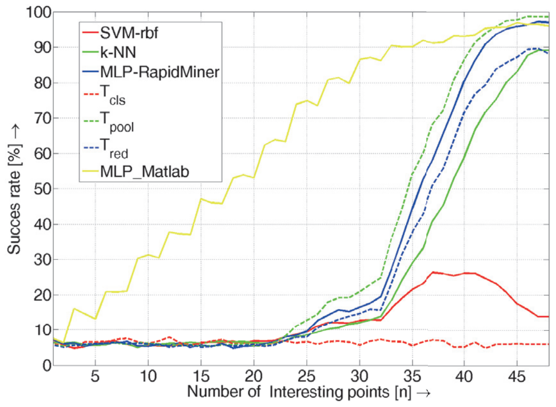

**Fig. 15.** Success rate of the secret *offset* revelation based on 250 power traces of DS2.

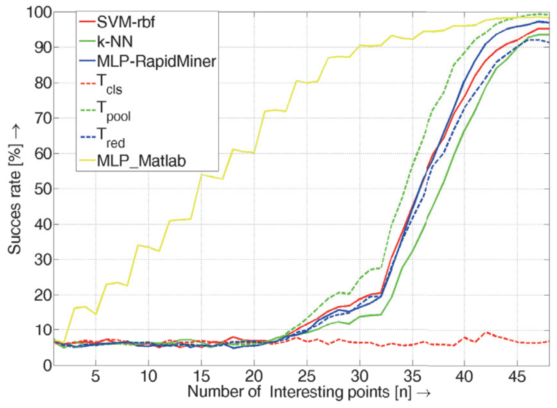

**Fig. 16.** Success rate of the secret *offset* revelation based on 500 power traces of DS2.

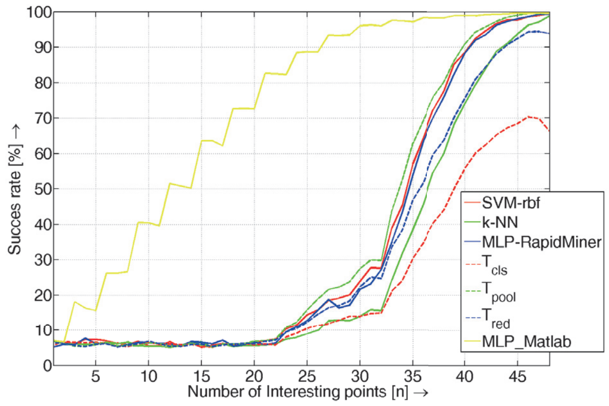

**Fig. 17.** Success rate of the secret *offset* revelation based on 1 000 power traces of DS2.

Displayed graphs confirm the theoretical assumption of the increasing success rate with the increasing number of power traces in profiling phase. The success rate is precipitously increasing until a maximal value and after that the value of the success rate stays almost constant. An interesting fact is that the number of power traces required to achieve the maximum value was comparable for every ML algorithms (especially for the best three, SVM-rbf, MLP and *k*-NN). Generally, about 500 power traces of DS1, 300 power traces of DS2 and 200 power traces of DS2 were necessary to achieve the maximal value. From a comparison of Fig. 12 and Fig. 13, we can see the influence of choosing the interesting points because these DSs were prepared based on identical power traces and aimed on the secret *offset* classification (in other words, datasets differ only in method of selecting interesting points). The shift of the maximum success rate values around 10 % is obvious. During the DS3 classification, the classical template attack provides really low values of calculated probabilities, therefore the first order success rate was worse when compared with other approaches.

In our **fifth experiment**, we investigate a success rate of the masks revelation depending on the number of interesting points and the number of power traces. Moreover, we investigate the influence of multiclass classification. In comparison with our previous experiments, we modified the program in such a way that ML classified instance to 256 classes (whole byte classification). In other words, ML and TA classified secret *offset* directly (not successively bit by bit). For this purpose, we prepared datasets based on DS2 that differed in the number of power traces and interesting points. We prepared learning sets that contain 100, 250, 500 and 1 000 of power traces successively and a test set that contains 1 500 instances in order to test profiling attacks. Figures 14, 15, 16 and 17 report the success rate to predict the right *offset* value as a function of the number of interesting points selected for the best profiled attacks: SVM-rbf, NN, *k*-NN, $T_{\text{cls}}$ and $T_{\text{pool}}$. The experiments described in [48], [49] implemented MLP approach in Matlab using Netlab. In order to extend our research on testing different implementation, we involved also the MLP approach implemented in Netlab [67] (denoted MLP_Matlab) and we involved to this experiment reduced template attack denoted as $T_{\text{red}}$. Reduced template attack is calculated according the equation (4) but the covariance matrix is equal to the identity matrix (reduced templates contain only the mean vector).

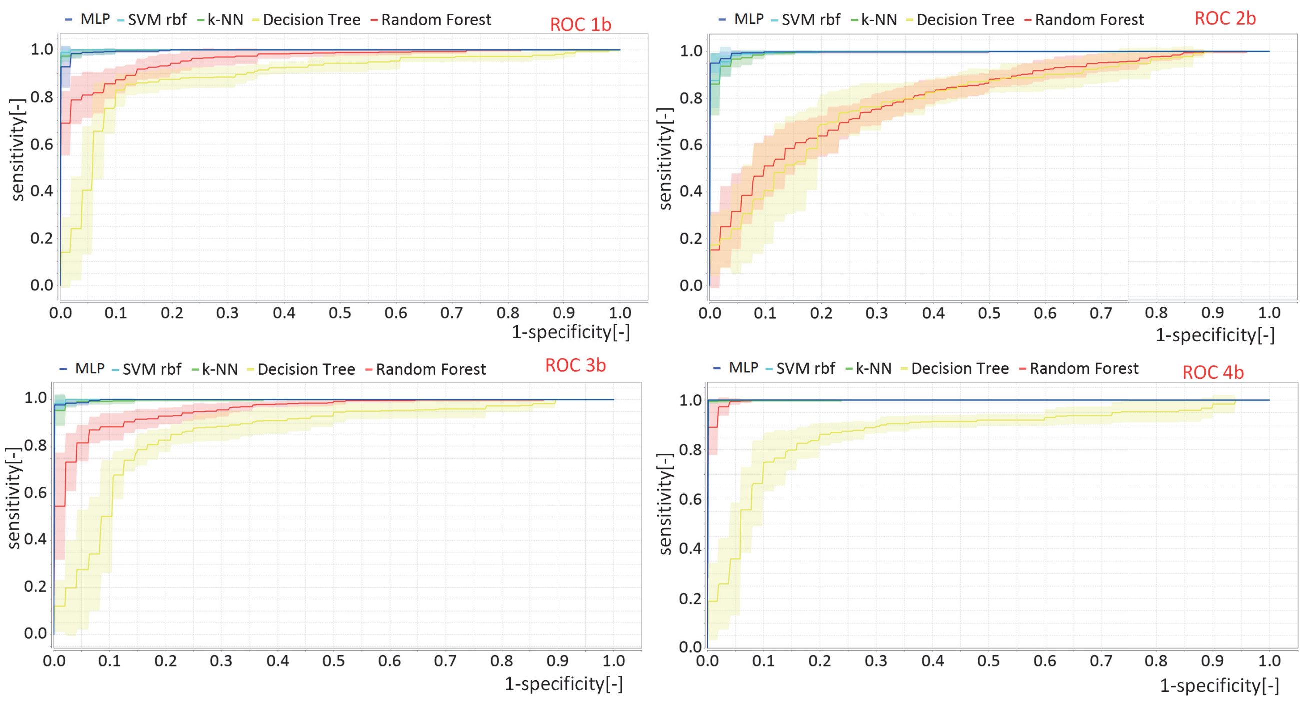

**Fig. 18.** ROC analysis for individual bits of DS2.

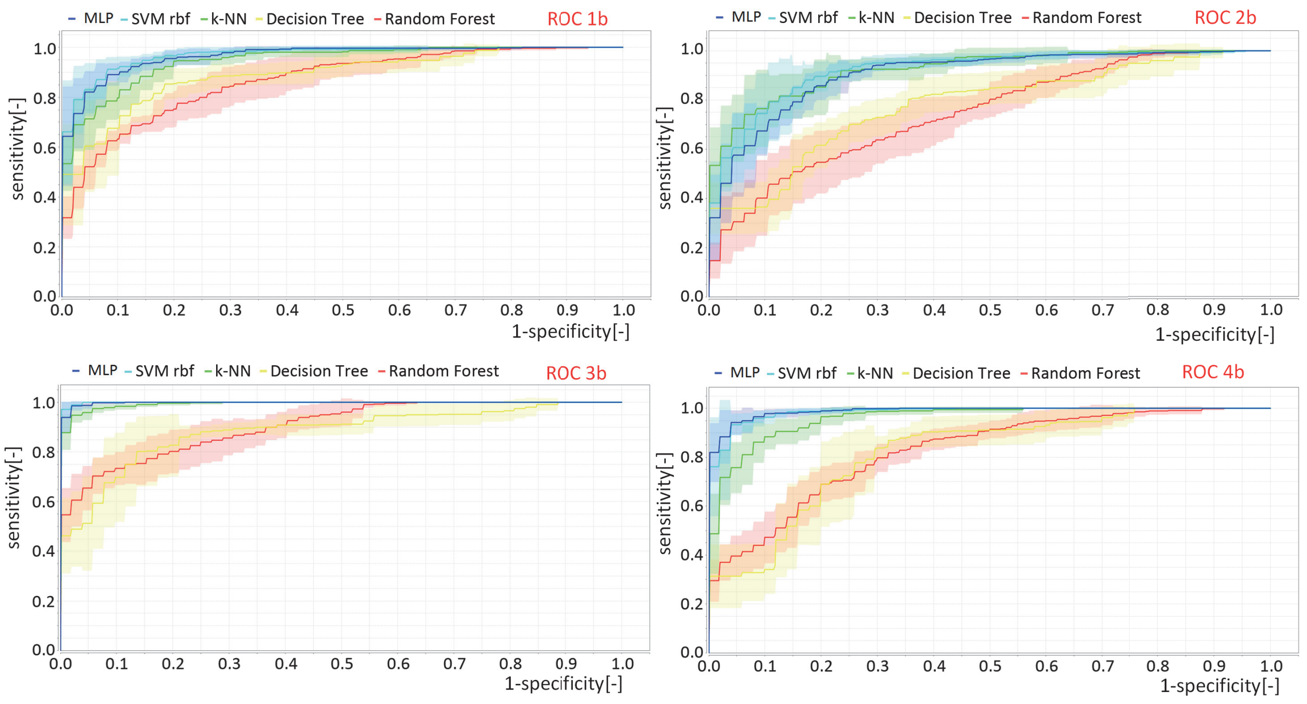

**Fig. 19.** ROC analysis for individual bits of DS3.

One can extract the following observations. First, as expected, the higher the number of traces in the learning set, the higher the accuracy. For example, maximal success rate achieved was 70 % and 99 % for learning set containing 100 and 1 000 power traces successively. Second, the number of the selected points in each trace influences the success rate: the higher the number of interesting points, the higher the success rate of every attack implementations. The main finding is that the rise in success rate of the attacks based on ML occurs much earlier than for every TA attack. We can observe success rate of 72 % for the MLP and of 7 % for TAs for 20 interesting points and 1 000 power traces.

It is remarkable that if learning set is small (in our experiment less than 1 000 of power traces), the classic template attack is practically inapplicable. It provides the success rate somewhere around 7 %. This is caused by the numerical problems that are connected with covariance matrix. These numerical problems occur during the inversion which needs to be done in equation (4). In our case, the values calculated were very small and that leads to bad classification results.

The obtained results confirmed that generally the ML approach is much more effective profiling power analysis attack in terms of small number of power traces and interesting points. It is pretty surprising that the MLP_Matlab approach is better in comparison with the second implementation. It is caused by more precise settings. The fact is that the template attack based on the pooled covariance matrix and the ML approaches (NN and SVM) are practically the same for the larger learning sets. The obtained success rates were 99.9 % and 99.6 % for $T_{\text{pool}}$ and MLP successively. Furthermore, the results obtained confirmed that $k$-NN is more similar to classic template attack. The success rate lies between $T_{\text{cls}}$ and $T_{\text{pool}}$. This approach is much more efficient than the classic template attack for smaller datasets, on the other hand, $T_{\text{pool}}$ is better, because it takes into account the relation between the interesting points selected. In practice, the $k$-NN approach corresponds with the reduced template attack, that does not take the covariance matrix into account. Naturally, we realized the same experiment for DS3 and we evaluated the results obtained. Not surprisingly, the results were practically the same, therefore we did not include those in the article.

In our **last experiment**, we performed ROC (Receiver Operator Characteristic) analysis for the chosen profiled attacks based on machine learning algorithms. This method is commonly used in medical decision making, and in ML in order to illustrate the performance of a binary classifier as its discrimination threshold is varied [68]. Therefore ROC graphs are useful to organize, select classifiers and visualize their performance.

The results of accuracy for individual models are calculated based on the confusion matrices (see section 4 and example in Table 2 for $k$-NN). The numbers along the major diagonal represent the correct decisions, and the numbers off diagonal represent the errors, the confusion between the various classes. In other words, each column of the table corresponds to the correct values of the class (in our case bit value 1 or 0) and each row corresponds to the predicted values.

For the following analysis, we denote:

- True positive (TP): the model predicted bit value 1 and the actual bit value was 1.
- False positive (FP): the model predicted 1 and the actual value was 0.
- True negative (TN): the model predicted 0 and the actual value was 0.
- False negative (FN): the model predicted 0 and the actual bit value was 1.

The *sensitivity* of a classifier is estimated as:

$$sensitivity = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{5}$$

The *specificity* of a classifier is estimated as:

$$specificity = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}. \tag{6}$$

Generally, ROC graphs are two-dimensional graphs in that *sensitivity* is plotted on the $Y$ axis and $1 - specificity$ is plotted on the $X$ axis, therefore they depict relative tradeoffs between benefits (TP) and costs (FP). It is well known that the best possible prediction model would yield a point $(0, 1)$ in the upper left corner of the ROC space. This point is also called a “perfect classification”, because it represents 100 % sensitivity (no FN) and 100 % specificity (no FP). The perfect classier produces a curve that runs vertically upwards from the origin $(0, 0)$ up to the point $(0, 1)$ and from this point horizontally to the right. A completely random guess (considering binary classification with the success rate of 50 %) would give a line along a diagonal from the origin (0,0) to the top right corner (1,1).

Based on previous results we involved MLP, SVM-rbf, $k$-NN, DT and RF into the ROC analysis successively. We implemented ML using optimal parameters that were discovered using the second experiment (please consult in Section 4 - part of *the second experiment*). We calculated ROC based on whole datasets prepared and 10-fold cross validation. The results of ROC analysis corresponding with each bit classification of DS1 are depicted in Fig. 20 in Appendix. The semitransparent areas indicate the standard deviation that results over the different cycles of 10-fold cross validation. Solid line indicates the average result of cross validation performed. It can be observed that some of the bits of the secret key can be perfect distinguished. This group includes the first bit, the third bit and the eighth bit. In these cases, each model provides almost perfect classification. Moreover, the remaining group of bits was also classified with high performance, but the differences between ML models were more significant. It is remarkable that the ROC curve plot of $k$−NN was the closest to the perfect classifier for each remaining bit and the second best model was SVM-rbf. Model based on $k$−NN provided perfect classification even though other models not (classification of bit 4, 5, 6 and 7). On the other hand, it was interesting that according the ROC comparison MLP was the worst for DS1. Naturally, the observation corresponds with the results of the second experiment (please consult this observation in Tab. 3). Based on our experiments and experience with MLP in profiled power analysis attack, we concluded that it is caused by a small number of interesting points[7].

[7]Since 2010, we have implemented mainly MLP approach instead of standard template attack in our profiling power analysis attacks that have conducted in researches and DPA Contests.

The results of ROC analysis corresponding with each bit classification of DS2 are depicted in Fig. 18 and shows that each 4 bits of secret *offset* were classified with great performance. Based on ROC plots of $k$-NN, SVM-rbf and MLP, it can be observed that these models are really close to the perfect classification. The difference between these models is really negligible for DS2.

ROC curves corresponding with each bit classification of DS3 are depicted in Fig. 19. As in previous case, $k$-NN, SVM-rbf and MLP provided the best plot in ROC space. Moreover, the influence of the most crucial step of profiling PA, that lies in selecting of the interesting points, is demonstrated by comparing Fig. 18 and Fig. 19, because datasets DS2 and DS3 were prepared from identical raw power traces. In Fig. 19, the plot of ROC curves are more distant from (0,1), therefore the selection was performed less precisely and the model created provided lower performance. We conclude that if selection of interesting points is properly performed, every possible distinguisher will provide more or less similar results regardless of the underlying technique deployed (either classic templates or machine learning after optimization).

## 5. Conclusion

In this paper, we provided an extensive comparison of widely used machine learning algorithms in power analysis such as SVM, decision tree, MLP including the new approach based on $k$-NN. We implemented a verification program that chose optimal settings of the parameters of individual machine learning algorithms in order to obtain the best classification accuracy. Based on the obtained results, we can consider SVM-rbf, MLP and $k$-NN as the most suitable candidates for profiling power analysis attacks (in terms of classification accuracy). Generally, we can optimize every ML algorithm using parameter settings to get almost identical classification results. On the other hand, optimization of individual ML algorithm can be time consuming (possible difference can be enormous based on the selected parameters and algorithm, for example 6 minutes and 35 seconds was needed for $k$-NN optimization and 8 days for SVM-poly optimization).

Moreover, we investigated a success rate of the masks revelation depending on the number of the interesting points and the number of power traces. As expected, the higher the number of traces and points in the learning set, the higher the accuracy of every power analysis attacks implemented. The main finding was that the sharp rise in success rate of the ML attacks (MLP and SVM) occurs much earlier than for every TA attacks. We can conclude that the ML approach is much more effective profiling power analysis attack in terms of a small number of power traces and interesting points. In other words, it is better to use profiling power analysis attack based on the MLP if the adversary has only limited power traces measured than to realize attack based on templates.

From every experiments realized, we see really good potential in $k$-NN algorithm. The approach proposed based on simplest $k$-NN algorithm can provide important advantages to the attacker compared with other profiling attacks . We summarize these observations in the followings points:

- the basic principle of the method is very simple, profiling phase constitutes only the storing of data measured in the memory (the attacker has to realize this in any case),
- it is not necessary to prepare (calculate) templates, the attacker can save time and memory,
- the $k$-NN approach is implemented by default in many program environments, therefore it is no problem with attack implementation,
- the success rate is comparable with the template attack, the $k$-NN approach corresponds with the reduced TA that does not take into account the covariance matrix,
- the attacker can use more interesting points compare with TA where it is limited due to memory limitation resulting from the covariance matrix,
- the $k$-NN does not include a learning phase compared with other MLs, therefore the attacker can work more efficiently (fast response to every changes related to the power traces measured, number of interesting points, size etc.).

We hope that this method will have continuance in profiled power analysis attacks, because from the basic principle follows a good assumption of the multivariate normal distribution classification (interesting points of power traces). We demonstrated this assumption with simple example and performed experiments.

## Acknowledgments

This Research described in this paper was financed by the National Sustainability Program under grant LO1401. For the research, infrastructure of the SIX Center was used.

## References

[1] JOYE, M., OLIVIER, F., Side-channel analysis. In *Encyclopedia of Cryptography and Security, 2nd ed.* 2011, p. 1198–1204. DOI: 10.1007/978-1-4419-5906-5_516

[2] KOCHER, P. C., JAFFE, J., JUN, B. Differential power analysis. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*. London (UK), 1999. p. 388–397.

[3] MANGARD, S., OSWALD, E., POPP, T. *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. 1st ed. Secaucus (USA): Springer US, 2007. ISBN: 978-0-387-30857-9. DOI: 10.1007/978-0-387-38162-6

[4] MARTINASEK, Z., CLUPEK, V., TRASY, K. General scheme of differential power analysis. In *36th International Conference on Telecommunications and Signal Processing (TSP), 2013*. Rome (Italy), July 2013. p. 358–362. DOI: 10.1109/TSP.2013.6613952

[5] CHARI, S., RAO, J., ROHATGI, P. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*. Redwood Shores (USA), 2003. p. 13–28. DOI: 10.1007/3-540-36400-5_3

[6] CHOUDARY, O., KUHN, M. G. Efficient template attacks. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS*. Berlin (Germany), November 2013. p. 253–270. DOI: 10.1007/978-3-319-08302-5_17

[7] SCHINDLER, W., LEMKE, K., PAAR, C. A stochastic model for differential side channel cryptanalysis. In *Proceedings of 7th International Workshop on Cryptographic Hardware and Embedded Systems - CHES 2005*. Edinburgh (UK), September 2005. p. 30–46. DOI: 10.1007/11545262_3

[8] SCHINDLER, W. On the optimization of side-channel attacks by advanced stochastic methods. In *Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography*, Les Diablerets (Switzerland), January 2005. p. 85–103. DOI:10.1007/978-3-540-30580-4_7

[9] RECHBERGER, C., OSWALD, E. Practical template attacks. In *Information Security Applications*, 2005. p. 440–456. ISBN: 978-3-540-24015-0. DOI: 10.1007/978-3-540-31815-6_35

[10] HANLEY, N., TUNSTALL, M., MARNANE, W. Using templates to distinguish multiplications from squaring operations. *International Journal of Information Security*, 2011, vol. 10, no. 4, p. 255–266. ISSN: 1615-5262. DOI: 10.1007/s10207-011-0135-4

[11] ARCHAMBEAU, C., PEETERS, E., STANDAERT, F.-X., et al. Template attacks in principal subspaces. In *Cryptographic Hardware and Embedded Systems - CHES 2006*. Yokohama (Japan), 2006, p. 1–14. DOI: 10.1007/11894063_1

[12] BAR, M., DREXLER, H., PULKUS, J. Improved template attacks. In *COSADE 2010 - First International Workshop on Constructive Side-Channel Analysis and Secure Design*. 2010, p. 81–89.

[13] BHASIN, S., DANGER, J.-L., GUILLEY, S., et al. Side-channel leakage and trace compression using normalized inter-class variance. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy*. New York (USA), 2014, p. 1–9. DOI: 10.1145/2611765.2611772

[14] GIERLICHS, B., LEMKE-RUST, K., PAAR, C. Templates vs. stochastic methods. In *Cryptographic Hardware and Embedded Systems - CHES 2006*. Yokohama (Japan), 2006, p. 15–29. DOI:10.1007/11894063_2

[15] 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 *Proceedings of the 12th Conference on Topics in Cryptology*. San Francisco (USA), 2012, p. 383–397. DOI: 10.1007/978-3-642-27954-6_24

[16] AKKAR, M.-L., BEVAN, R., DISCHAMP, P., et al. Power analysis, what is now possible... In *Advances in Cryptology - ASIACRYPT 2000*. Kyoto (Japan), 2000, p. 489–502. DOI: 10.1007/3-540-44448-3_38

[17] HAJNY, J., MALINA, L. Anonymous credentials with practical revocation. In *Satellite Telecommunications (ESTEL), IEEE First AESS European Conference on*. Rome (Italy), 2012, p. 1–6. DOI: 10.1109/ESTEL.2012.6400081

[18] BRIER, E., CLAVIER, C., OLIVIER, F. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004*. Cambridge (USA), 2004, p. 16–29. DOI: 10.1007/978-3-540-28632-5_2

[19] OSWALD, E., MANGARD, S., PRAMSTALLER, N. Secure and efficient masking of AES - a mission impossible?, 2004. [Online] Cited 2015-07-17. Available at: http://eprint.iacr.org/2004/134

[20] GOLIC, J., TYMEN, C. Multiplicative masking and power analysis of AES. In *Cryptographic Hardware and Embedded Systems CHES 2002*. Redwood Shores (USA), 2003, p. 198–212. DOI: 10.1007/3-540-36400-5_16

[21] OSWALD, E., MANGARD, S., PRAMSTALLER, N., et al. A side-channel analysis resistant description of the AES S-box. In *Fast Software Encryption*. Paris (France), 2005, p. 413–423. DOI: 10.1007/11502760_28

[22] CANRIGHT, D., BATINA, L. A very compact "perfectly masked" S-box for AES. In *Proceedings of the 6th International Conference on Applied Cryptography and Network Security*. New York (USA), 2008, p. 446–459. DOI: 10.1007/978-3-540-68914-0_27

[23] NASSAR, M., SOUISSI, Y., GUILLEY, S., et al. RSM: A small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In *Design, Automation Test in Europe Conference Exhibition (DATE*. Dresden (Germany), 2012, p. 1173–1178. DOI: 10.1109/DATE.2012.6176671

[24] YE, X., EISENBARTH, T. On the vulnerability of low entropy masking schemes. In *Smart Card Research and Advanced Applications*. Berlin (Germany), 2014, p. 44–60. DOI: 10.1007/978-3-319-08302-5_4

[25] BHASIN, S., DANGER, J.-L., GUILLEY, S., et al. A low-entropy first-degree secure provable masking scheme for resource-constrained devices. In *Proceedings of the Workshop on Embedded Systems Security*. New York (USA), 2013, p. 1–10. DOI: 10.1145/2527317.2527324

[26] PROUFF, E., RIVAIN, M. A generic method for secure Sbox implementation. In *Information Security Applications*. Jeju Island (Korea), 2007, p. 227–244. DOI: 10.1007/978-3-540-77535-5_17

[27] BHASIN, S., BRUNEAU, N., DANGER, et al. Analysis and improvements of the DPA contest v4 implementation. In *Security, Privacy, and Applied Cryptography Engineering*. Pune (India), 2014, p. 201–218. DOI: 10.1007/978-3-319-12060-7_14

[28] GROSSO, V., STANDAERT, F.-X., PROUFF, E. Low entropy masking schemes, revisited. In *Smart Card Research and Advanced Applications*. Berlin (Germany), 2014, p. 33–43. DOI: 10.1007/978-3-319-08302-5_3

[29] GUILLEYHO, S. DPA contest v4, 2013. [Online] Cited 2015-07-17. Available at: http://www.dpacontest.org/v4/rsm_doc.php

[30] MESSERGES, T. Using second-order power analysis to attack DPA resistant software. In *Cryptographic Hardware and Embedded Systems CHES 2000*. Worcester (USA), 2000. DOI: 10.1007/3-540-44499-8_19

[31] MANGARD, S., PRAMSTALLER, N., OSWALD, E. Successfully attacking masked AES hardware implementations. In *Cryptographic Hardware and Embedded Systems CHES 2005*. Edinburgh (UK), 2005, p. 157–171. DOI: 10.1007/11545262_12

[32] ANDERSON, J. R., MICHALSKI, R. S., CARBONELL, J. G., et al. *Machine Learning: An Artificial Intelligence Approach*. Springer-Verlag Berlin Heidelberg, 1983. ISBN: 978-3-662-12405-5

[33] 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: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. Amsterdam (Netherlands), 2007, p. 3–24. ISBN: 978-1-58603-780-2

[34] HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. Unsupervised learning. In *The Elements of Statistical Learning*. 2009, p. 485–585. ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7_14

[35] QUISQUATER, J.-J., SAMYDE, D. Automatic code recognition for smart cards using a Kohonen neural network. In *Proceedings of the 5th Conference on Smart Card Research and Advanced Application Conference - Volume 5*. Berkeley (USA), 2002, p. 6–6.

[36] KUR, J., SMOLKA, T., SVENDA, P. Improving resiliency of java card code against power analysis. In *Mikulasska kryptobesidka, Sbornik prispevku*, 2009, p. 29–39.

[37] YANG, S., ZHOU, Y., LIU, J., et al. Back propagation neural network based leakage characterization for practical security analysis of cryptographic implementations. In *Information Security and Cryptology - ICISC 2011*. Seoul (Korea), 2011, p. 169–185. DOI: 10.1007/978-3-642-31912-9_12

[38] 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*. Darmstadt (Germany), 2011, p. 29–41.

[39] LERMAN, L., BONTEMPI, G., MARKOWITCH, O.. Power analysis attack: an approach based on machine learning. *International Journal of Applied Cryptography*, 2014, vol. 3, no. 2, p. 97–115. ISSN: 1753-0563. DOI: 10.1504/IJACT.2014.062722

[40] HOSPODAR, G., GIERLICHS, B., DE MULDER, E., et al. Machine learning in side-channel analysis: a first study. *Journal of Cryptographic Engineering*, 2011, vol. 1, no. 4, p. 293–302. DOI: 10.1007/s13389-011-0023

[41] HOSPODAR, G., DE MULDER, E., GIERLICHS, B., et al. Least squares support vector machines for side-channel analysis. In *COSADE 2011 - Second International Workshop on Constructive Side-Channel Analysis and Secure Design*. Darmstadt (Germany), 2011, p. 293–302.

[42] HEUSER, A., ZOHNER, M. Intelligent machine homicide - breaking cryptographic devices using support vector machines. In *Constructive Side-Channel Analysis and Secure Design: Third International Workshop, COSADE*. Darmstadt (Germany), 2012, p. 249–264. DOI: 10.1007/978-3-642-29912-4_18

[43] BARTKEWITZ, T., LEMKE-RUST, K. Efficient template attacks based on probabilistic multi-class support vector machines. In *Smart Card Research and Advanced Applications*. Graz (Austria), 2013, p. 263–276. DOI: 10.1007/978-3-642-37288-9_18

[44] MLADENIC, D., BRANK, J., GROBELNIK, M., et al. Feature selection using support vector machines. In *The 27th Annual International ACM SIGIR Conference (SIGIR 2004)*. 2004, p. 234–241.

[45] LERMAN, L., BONTEMPI, G., TAIEB, S. B., et al. A time series approach for profiling attack. In *Proceedings of the Third International Conference Security, Privacy, and Applied Cryptography Engineering SPACE*. Kharagpur (India), 2013, p. 75–94. DOI: 10.1007/978-3-642-41224-0_7

[46] HE, H., JAFFE, J.,ZOU, L. *Side channel cryptanalysis using machine learning*. Stanford, 2012.

[47] LERMAN, L., MEDEIROS, S. F., BONTEMPI, G., et al. A machine learning approach against a masked AES. In *Proceedings of the 12th International Conference of Smart Card Research and Advanced Applications CARDIS*. Berlin (Germany), 2013, p. 61–75. DOI: 10.1007/978-3-319-08302-5_5

[48] MARTINASEK, Z., ZEMAN, V. Innovative method of the power analysis. *Radioengineering*, 2013, vol. 22, no. 2, p. 586–594. ISSN: 1210-2512

[49] MARTINASEK, Z., HAJNY, J., MALINA, L. Optimization of power analysis using neural network. In *Proceedings of the 12th International Conference Smart Card Research and Advanced Applications CARDIS*. Berlin (Germany), 2013, p. 94–107. DOI: 10.1007/978-3-319-08302-5_7

[50] LERMAN, L., MEDEIROS, S. F., VESHCHIKOV, et al. Semi-supervised template attack. In *Proceedings of the 4th International Workshop Constructive Side-Channel Analysis and Secure Design COSADE*. Paris (France), p. 184–199. DOI: 10.1007/978-3-642-40026-1_12

[51] CHOU, J.-W., CHU, M.-H., TSAI, Y.-L., et al. *Advances in Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2013. (An unsupervised learning model to perform side channel attack). ISBN: 978-3-642-37452-4. DOI: 10.1007/978-3-642-37453-1_34

[52] HEYSZL, J., IBING, A., MANGARD, S., et al. Clustering algorithms for non-profiled single-execution attacks on exponentiations. In *Proceedings of the 12th International Conference Smart Card Research and Advanced Applications CARDIS 2013*. Berlin (Germany), 2013, p. 79–93. DOI: 10.1007/978-3-319-08302-5_6

[53] ZHANG, Z., WU, L., WANG, A., et al. Improved leakage model based on genetic algorithm. *IACR Cryptology ePrint Archive*, 2014.

[54] PERIN, G., IMBERT, L., TORRES, L., et al. Attacking randomized exponentiations using unsupervised learning. In *Proceedings of the 4th International Workshop Constructive Side-Channel Analysis and Secure Design COSADE*. Paris (France), 2014, p. 144–160. DOI: 10.1007/978-3-319-10175-0_11

[55] AUMONIER, S. Generalized correlation power analysis. In *Proceedings of the Ecrypt Workshop Tools for Cryptanalysis*, 2007.

[56] JAP, D., BREIER, J. Overview of machine learning based side-channel analysis methods. In *14th International Symposium on Integrated Circuits (ISIC)*. 2014, p. 38–41. DOI: 10.1109/ISICIR.2014.7029524

[57] WHITNALL, C., OSWALD, E. Robust profiling for DPA-style attacks. In *17th International Workshop Cryptographic Hardware and Embedded Systems - CHES*. Saint-Malo (France), 2015, p. 3–21. DOI: 10.1007/978-3-662-48324-4_1

[58] STANDAERT, F.-X., MALKIN, T. G., YUNG, M. A unified framework for the analysis of side-channel key recovery attacks. In *Proceedings of the 28th Annual International Conference on Advances in Cryptology: the Theory and Applications of Cryptographic Techniques*. Berlin, (Germany), 2009, p. 443–461. DOI: 10.1007/978-3-642-01001-9_26

[59] FEI, Y., LUO, Q., DING, A. A statistical model for DPA with novel algorithmic confusion analysis. In *Cryptographic Hardware and Embedded Systems, CHES 2012*. Leuven (Belgium), 2012, p. 233–250. DOI: 10.1007/978-3-642-33027-8_14

[60] TSOUMAKAS, G., KATAKIS, I. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*. Idea Group Publishing, vol. 3, no. 3, 2007, p. 1–13. ISSN: 1548-3924

[61] READ, J., PFAHRINGER, B., HOLMES, G., et al. Classifier chains for multi-label classification. *Machine Learning*, 2011, vol. 85, no. 3, p. 333–359. DOI: 10.1007/s10994-011-5256-5

[62] ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 1992, vol. 46, no. 3, p. 175–185. DOI: 10.2307/2685209

[63] EVERITT, B., LANDAU, S., LEESE, M., et al. *Cluster Analysis*. 5th ed. Wiley, 2011. ISBN: 9780470749913. DOI: 10.1002/9780470977811

[64] MORADI, A., GUILLEY, S., HEUSER, A. Detecting hidden leakages. In *Proceedings of the 12th International Conference Applied Cryptography and Network Security: ACNS 2014*. Lausanne (Switzerland), 2014, p. 324–342. DOI: 10.1007/978-3-319-07536-5_20

[65] HOFMANN, M., KLINKENBERG, R. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. CRC Press, 2013. ISBN: 9781482205497

[66] SOKOLOVA, M., LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 2009, vol. 45, no. 4, p. 427–437. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2009.03.002

[67] NABNEY, I. T. *NETLAB: Algorithms for Pattern Recognition*. Advances in Pattern Recognition. Springer-Verlag New York, Inc., New York (USA), 2002. ISBN: 978-1-85233-440-6

[68] FAWCETT, T. An introduction to ROC analysis. *Pattern Recognition Letters*, 2006, vol. 27, no. 8, p. 861–874. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2005.10.010

## About the Authors...

**Zdeněk MARTINÁSEK** received MSc. (Ing.) at the Department of Telecommunications at the Faculty of Electrical Engineering and Communication at Brno University of Technology in 2008. He received Ph.D. at the same Department. He also helps to cover pedagogically Master's program course. The area of his professional interests is cryptography, power analysis, sensors and modern data communication.

**Václav ZEMAN** received MSc. (Ing) at Faculty of Electrical Engineering and Communication at Brno University of Technology in 1991. He received Ph.D. at the Department of Telecommunications the at same Faculty in 2003. Now he is the Associate Professor (doc. 2005) at Faculty of Electrical Engineering and Communication at Brno University of Technology. He publishes in the cryptology area and communication systems area. Now, he is a lecturer and he deals with cryptology and information system security.

**Lukáš MALINA** is currently with the Department of Electrical Engineering and Communication, Brno University of Technology. He received the MSc degree in 2009 and Ph.D. in 2014 from Brno University of Technology, Czech Republic. His research interests include security, lightweight cryptography, group signatures and privacy.

**Josef MARTINÁSEK** is currently with the Institute of Structural Mechanics, Faculty of Civil Engineering, Brno University of Technology. He received the MSc degree in 2008. His research interests include data-mining and static of concrete structures.

## 6. Appendix

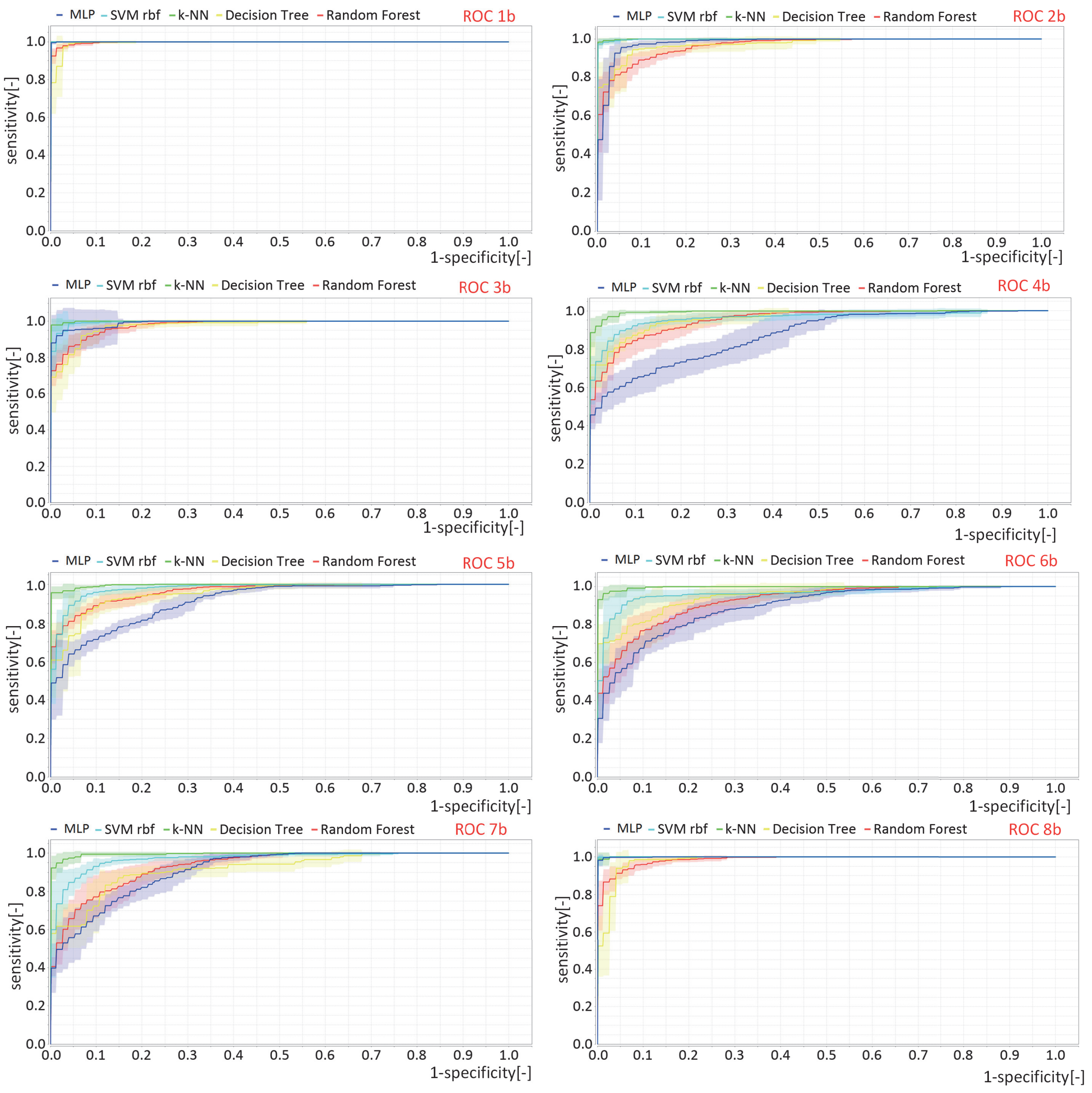

**Fig. 20.** ROC analysis for individual bits of DS1.