J Cryptogr Eng (2011) 1:293–302
DOI 10.1007/s13389-011-0023-x

REGULAR PAPER

Machine learning in side-channel analysis: a first study

Gabriel Hospodar · Benedikt Gierlichs ·
Elke De Mulder · Ingrid Verbauwhede ·
Joos Vandewalle

Received: 30 September 2011 / Accepted: 1 October 2011 / Published online: 27 October 2011

Abstract Electronic devices may undergo attacks going beyond traditional cryptanalysis. Side-channel analysis (SCA) is an alternative attack that exploits information leaking from physical implementations of e.g. cryptographic devices to discover cryptographic keys or other secrets. This work comprehensively investigates the application of a machine learning technique in SCA. The considered technique is a powerful kernel-based learning algorithm: the Least Squares Support Vector Machine (LS-SVM). The chosen side-channel is the power consumption and the target is a software implementation of the Advanced Encryption Standard. In this study, the LS-SVM technique is compared to Template Attacks. The results show that the choice of parameters of the machine learning technique strongly impacts the performance of the classification. In contrast, the number of power traces and time instants does not influence the results in the same proportion. This effect can be attributed to the usage of data sets with straightforward Hamming weight leakages in this first study.

Keywords Power analysis · Side-channel analysis · Cryptography · Support vector machines · Machine learning

1 Introduction

Security in electronic devices must not only rely on cryptographic algorithms proven to be mathematically secure. An attacker may use information leaked from side-channels [14] resulting from physical implementations. This is called side-channel analysis (SCA) and it may endanger the overall security of a system. The ever increasing demand for security on a number of applications, including the internet of things, financial transactions, electronic communications, and data storage, should drive designers to strongly consider the possibility of physical attacks in addition to attacks on cryptographic algorithms, as most of these applications require the use of embedded devices.

In cryptography, SCA usually aims at revealing cryptographic keys. The underlying hypothesis for SCA assumes that physical observables carry information about the internal state of a chip implementing some cryptographic algorithm. In this context, useful key-related information can often be obtained from side-channels such as processing time [13], power consumption [14], and electromagnetic emanation [8,17].

In this work, we focus on power analysis. Given a set of power traces measured from a chip implementing a cryptographic algorithm, the ultimate goal is to tell which cryptographic key has been processed internally. This problem is frequently addressed using a divide-and-conquer approach. This approach breaks down a problem into practically tractable sub-problems. For example, parts of the cryptographic key, also known as subkeys, may be attacked at a time. Each attack may be formulated as a classification problem intending to discover which subkey is linked to a given power trace.

Machine learning [16] is often used to solve classification and regression problems. Machine learning concerns computer algorithms that can automatically learn from

This work was supported in part by the European Commission's ECRYPT II NoE (ICT-2007-216676), by the Belgian State's IAP program P6/26 BCRYPT, by the K.U. Leuven-BOF (OT/06/40) and by the Research Council K.U. Leuven: GOA TENSE (GOA/11/007). Benedikt Gierlichs is a Postdoctoral Fellow of the Fund for Scientific Research, Flanders (FWO).

G. Hospodar (✉) · B. Gierlichs · E. De Mulder · I. Verbauwhede ·
J. Vandewalle
Katholieke Universiteit Leuven, ESAT-SCD-COSIC and IBBT,
Kasteelpark Arenberg 10, 3001 Leuven-Heverlee, Belgium
e-mail: gabriel.hospodar@esat.kuleuven.be

experience. As a number of SCA-related problems can be formulated as classification problems, it turns out that machine learning represents a potential, useful tool for SCA.

Machine learning techniques have already been applied in cryptanalysis [19]. Furthermore, Backes et al. [2] used machine learning techniques for acoustic side-channel attacks on printers. To the best of our knowledge, there has not been further thorough investigation on the application of machine learning techniques in SCA.

This contribution lies in the context of profiled attacks, such as Template Attacks (TAs) [5]. These attacks comprise two phases: profiling and classification. In machine learning terms, these phases are respectively known as training and testing. In TAs, multivariate Gaussian templates of noise within power traces are generated either for all possible subkeys or for the results of some particular function involving them. Subsequently, power traces are classified under a maximum likelihood approach. The TA is regarded as the strongest form of side-channel attack possible in an information theoretic sense [5]. Although TA can be seen as a machine learning technique, we refer to TA and machine learning techniques separately in this work.

In our work, the machine learning technique used is the Least Squares Support Vector Machines (LS-SVMs) [20], which is a kernel-based learning algorithm. LS-SVM classifiers have achieved considerable results in comparison to other learning techniques, including standard SVM classifiers, for classification on 20 public domain benchmark data sets [9].

We trained LS-SVM classifiers by supervised learning. In the training phase, a number of power traces were provided to the classifier to teach it the most important features of the data set. In the testing phase, one unseen power trace was presented to the classifier at a time.

To get insight into the behavior of all the LS-SVM parameters, we focused on binary classification problems. Artificial data were created based on power traces extracted from a software implementation of the Advanced Encryption Standard (AES) without countermeasures. We considered one single S-Box lookup. In this first study, there was no actual key recovery yet.

Three experiments were conducted. We first investigated the influence of changing the parameters of the LS-SVM classifiers to learn their effects. Analyses varying both the numbers of power traces and their components (time instants possibly preprocessed) were also performed. In addition, we examined the impact of three feature selection techniques [Pearson correlation coefficient approach for component selection [18], sum of squared pairwise t-differences (SOST) of the average signals [10] and principal component analysis (PCA) [12]] and one preprocessing technique (outliers removal).

Our approaches were compared to TAs in terms of effectiveness to provide a known reference. TAs strongly rely on parametric estimations of Gaussian distributions. Machine learning techniques are able to bypass restrictive assumptions about probability density distributions.

This paper is organized as follows: Sect. 2 describes the feature selection techniques used in this work. In Sect. 3, our selected machine learning technique is explained. The results of the experiments are presented in Sect. 4. The preprocessing technique is also explained in this section. Section 5 presents the conclusions. Section 6 is dedicated to future work.

2 Feature selection

Generally, machine learning approaches make use of a preliminary step before tackling the classification problem to be solved. This step is called feature selection. It filters out and/or preprocesses the components of a given data set in order to extract the intrinsic parts of the data containing the most relevant pieces of information under some criteria. The two main advantages of using the feature selection step are (1) it allows for the reduction of the computational burden of classifiers with respect to processing and memory issues; and (2) it avoids confusing or teaching the wrong features of the data to the classifier. High-dimensional data can prevent classifiers from working in practice.

Power traces have thousands or millions of samples in time, which are here regarded as components. In this work, each component is represented by t. Many of them do not carry relevant information related to the targeted subkeys. They rather represent noise and ideally should not be presented to the classifier.

The following sections are presented: the Pearson correlation coefficient approach for component selection (mostly used in this work); the SOST of the average signals; and the PCA.

2.1 Pearson correlation coefficient approach

A straightforward way to select the N most relevant components of the power traces concerns finding the N points which have the largest correlations with respect to a function of the targeted subkey, as shown by Rechberger and Oswald [18].

The Pearson correlation coefficient is given by

$$\rho(t) = \frac{\operatorname{cov}(x(t), y)}{\sqrt{\operatorname{var}(x(t)) \cdot \operatorname{var}(y)}}, \quad -1 \leq \rho(t) \leq 1,$$

where $\operatorname{cov}(\cdot,\cdot)$ represents the covariance, $\operatorname{var}(\cdot)$ represents the variance, $x(t)$ is the vector with the component t of all power traces in the training set, and y is a vector containing the target values.

2.2 Sum of squared pairwise T-differences (SOST)

Chari et al. [5] originally proposed the following method to choose the most relevant components of the power traces, as part of TAs: let M_i be the statistical average of the power traces associated with the subkey i, $i = 1, \ldots, S$. The N most relevant components would be those at which large differences show up after computing the sum of pairwise differences between the average signals M_i.

Gierlichs et al. [10] showed that this feature selection method was not optimal. Better results were achieved using the SOST of the average signals. SOST is based on the T-Test—a statistical tool to distinguish signals. The T-Test is expressed by the following ratio:

$$T = \frac{M_i(t) - M_j(t)}{\sqrt{\frac{\sigma_i^2(t)}{n_i} + \frac{\sigma_j^2(t)}{n_j}}}. \tag{1}$$

The denominator of the formula weights the difference between $M_i(t)$ and $M_j(t)$ according to the variabilities $\sigma_i^2(t)$ and $\sigma_i^2(t)$, in relation to the number of signals n_i and n_j associated with the sets i and j. Equation (1) can also be seen as an analogy to the signal-to-noise ratio, in which the difference between the averages is the signal and the denominator is a measure of dispersion, being interpreted as noise. This noise may make the distinction between the distributions with averages $M_i(t)$ and $M_j(t)$ hard, but it should vanish if n_i and n_j are large.

The most relevant components will be those at which the SOST,

$$\mathrm{SOST}(t) = \sum_{j>i=0}^{S} \left(\frac{M_i(t) - M_j(t)}{\sqrt{\frac{\sigma_i^2(t)}{n_i} + \frac{\sigma_j^2(t)}{n_j}}} \right)^2,$$

presents the N highest peaks.

2.3 Principal component analysis (PCA)

Principal component analysis [12] is a well-known orthogonal, non-parametric transformation that provides a way to efficiently select relevant information from data sets. The core idea is to compute a new basis that better expresses the data set, revealing its intrinsic structure.

By assuming linearity, the set of new plausible bases is significantly reduced and the problem turns into finding an appropriate change of basis. PCA also assumes that mean and variance are sufficient statistics.

Presuming that the directions with largest variances contain most relevant information, PCA sorts the transformed, most important components of a vector with regard to their variances.

In addition to maximizing the signal, measured by the interclass variance, PCA also intends to minimize the redundancy within components of the data set. This can be achieved by setting all off-diagonal terms of the covariance matrix of the transformed data to zero, making the components uncorrelated to each other.

After estimating the covariance matrix $\mathbf{C}$ of the original data set $\mathbf{x} \in \mathbb{R}^M$, where M is the number of components of $\mathbf{x}$, the $N < M$ eigenvectors related to the largest eigenvalues λ_t from the eigenvector decomposition $\mathbf{C}u_t = \lambda_t u_t$ should be selected. The transformed, lower dimensional variable will be given by $z_t = u_t^{\mathrm{T}}(x_t - \mu_t)$, $t = 1, \ldots, N$, where μ_t is the mean of the t-th component of $\mathbf{x}$. The error on the new data set resulting from the dimensionality reduction is determined by $\sum_{t=N+1}^{M} \lambda_t$. For the transformed variable, t does not have a time connotation anymore.

3 Classification

The classification technique used in this work is the LS-SVM. LS-SVM tackles linear systems rather than solving convex optimization problems, typically quadratic programs, as in standard support vector machines (SVM) [7]. This is done by both introducing a least squares loss function and working with equalities, instead of the intrinsic inequalities of SVM formulations. One advantage of this reformulation is complexity reduction.

(LS-)SVM classifiers are originally formulated to perform binary classification. In the training phase, the (LS-)SVM classifier constructs a hyperplane in a high-dimensional space aiming to separate the data according to the different classes. This data separation should occur in such a way that the hyperplane has the largest distance to the nearest training data points of any class. These particular training data points define the so-called margin.

Let $D_n = \{(x_k, y_k) \colon x_k \in \mathbb{R}^N, y_k \in \{-1, +1\}; k = 1, \ldots, n\}$ be a training set, where x_k and y_k are, respectively, the k-th input (power trace after feature selection) and output (subkey-related) patterns.

The classifier in the primal weight space takes the following form:

$$y = \mathrm{sign}[w^{\mathrm{T}}\varphi(x) + b],$$

where $\mathrm{sign}(x) = -1$ if $x < 0$, else $\mathrm{sign}(x) = 1$, and $\varphi(x) \colon \mathbb{R}^N \to \mathbb{R}^{N_f}$ maps the N-dimensional input space into a higher, possibly infinite, N_f-dimensional space. Both the weights $w \in \mathbb{R}^{N_f}$ and bias $b \in \mathbb{R}$ are parameters of the classifier. These parameters can be found by solving the following optimization problem having a quadratic cost function and equality constraints:

$$\min_{w,b,e} L(w,e) = \frac{1}{2} w^{\mathrm{T}} w + \frac{\gamma}{2} \sum_{k=1}^{n} e_k^2$$

$$\text{s.t. } y_k[w^{\mathrm{T}} \varphi(x_k) + b] = 1 - e_k, \quad k = 1, \ldots, n, \qquad (2)$$

which is a modification of the basic SVM formulation. In Eq. (2), $e = [e_1, \ldots, e_n]^{\mathrm{T}}$ is a vector of error variables, tolerating misclassification, and γ is the regularization parameter, determining the trade-off between the margin size maximization and the training error minimization.

After constructing the Lagrangian,

$$L(w, b, e; \alpha) = \frac{1}{2} w^{\mathrm{T}} w + \gamma \frac{1}{2} \sum_{k=1}^{n} e_k^2 - \sum_{k=1}^{n} \alpha_k \{ y_k [w^{\mathrm{T}} \varphi(x_k) + b] - 1 + e_k \},$$

and taking the conditions for optimality, by setting

$$\frac{\partial L}{\partial w} = 0, \quad \frac{\partial L}{\partial b} = 0, \quad \frac{\partial L}{\partial e_k} = 0, \quad \frac{\partial L}{\partial \alpha_k} = 0, \quad k = 1, \ldots, n,$$

the classifier formulated in the dual space is given by

$$y(x) = \mathrm{sign}\left(\sum_{k=1}^{n} \alpha_k y_k K(x, x_k) + b \right),$$

where $K(x, x_k) = \varphi(x)^{\mathrm{T}} \varphi(x_k)$ is a positive definite kernel matrix, $\alpha_k \in \mathbb{R}$ are the Lagrange multipliers, or support values. Both α_k and b are the solutions of the following linear system

$$\begin{pmatrix} 0 & y^{\mathrm{T}} \\ y & \Omega + \frac{1}{\gamma} I_n \end{pmatrix} \begin{pmatrix} b \\ \alpha \end{pmatrix} = \begin{pmatrix} 0 \\ 1_n \end{pmatrix},$$

with $1_n = (1, \ldots, 1)^{\mathrm{T}}$ and $\Omega_{kl} = y_k y_l \varphi(x_k)^{\mathrm{T}} \varphi(x_l)$. The solution is unique when the matrix corresponding to the linear system has full rank.

According to Mercer's theorem [1], a positive definite K guarantees the existence of the feature map φ, which is often not explicitly known.

From $\frac{\partial L}{\partial e_k} = 0$, we have $\alpha_k = \gamma e_k$, meaning that the support values are proportional to the errors corresponding to the training data points. As $\alpha_k \neq 0$, $k = 1, \ldots, n$, every data point is a support vector, implying lack of sparseness. High α_k values suggest high contributions of training data points on the decision boundary created by the classifier to distinguish the different classes.

3.1 Practicalities

Roughly, when working with (LS-)SVMs one usually chooses the kernel K between the linear kernel,

$$K(x, x_k) = x_k^{\mathrm{T}} x,$$

and the radial basis function (RBF) kernel,

$$K(x, x_k) = \exp\left\{ \frac{-||x - x_k||_2^2}{\sigma^2} \right\},$$

where $||\cdot||_2$ is the L_2-norm and $\sigma^2 \in \mathbb{R}^+$ is a parameter to be chosen. Other kernel options, such as the polynomial kernel and the multilayer perceptron (MLP) kernel, are beyond the scope of this work.

Kernel-based models usually depend on parameters controlling both their accuracies and complexities. When using linear kernels, the only parameter to be tuned is γ. If $\gamma \to 0$, the solution favors margin maximization, putting less emphasis on minimizing the misclassification error. RBF kernels require tuning an additional parameter σ^2, which is directly related to the shape of the decision boundary.

Considering RBF kernels, both parameters γ and σ^2 should be optimized for the classifier to maximize the success rates concerning the analysis of unknown power traces from the testing data set. A combination of a cross-validation and a grid search algorithm is recommended in the literature [20] for parameter tuning. Cross-validation can help preventing overfitting. However, its computation may be computationally costly.

Many classification problems have more than two classes. As (LS-)SVMs are designed to perform binary classification, a typical approach to cope with a multi-class problem is to split the problem up into binary classification problems using some coding technique. For example, a multi-class problem comprising p classes may be broken down into $\lceil \log_2 p \rceil$ binary classification tasks. Subsequently, the results of the binary classifiers should be combined to reconstruct a valid result for the original multi-class problem. In this work we only deal with binary classification problems.

4 Results

This section investigates the potential that LS-SVMs have to work as robust power traces analyzers. We chose to get started with a thorough investigation on the machine learning technique behavior. To this end, several tests were performed using both linear and RBF kernels. We are one step behind of the actual discovery of cryptographic subkeys. In this work, our attacks distinguish between two different classes related to the output of one AES S-Box.

4.1 Experimental settings

Preliminary tests were performed on real measurements from an implementation consisting of the subpart of the AES algorithm composed by the XOR between an 8-bit subkey and the input word, followed by the application of one S-Box. The LS-SVM supervised learning classifiers have been implemented using the LS-SVMlab1.7 [4].

Our data set contains 5,000 power traces with 2,000 components each. In this work, attacks distinguish only between two different classes, to help us initially build a solid understanding about the techniques involved. These two different classes were chosen in three ways.

The first two approaches considered the relationship between the power traces and the internal state of the cryptographic algorithm to be represented by the Hamming weight model [15]. The *threshold* approach divided the data set into two classes depending on the Hamming weights of the outputs of the S-Box (less than or greater than 4). The *intercalated* approach divided the data set depending if the Hamming weights were even or odd. The third approach, named *bit(4)*, focused on the fourth least significant bit of the output of the S-Box, since it was the bit leaking more information. Both *intercalated* and *bit(4)* approaches were created so that their classes would not be as trivially separable as those from the *threshold* approach.

In the training phase, a number of inputs from the training set were provided to the classifier to teach it the most important features of the data set. In the testing phase, one input of the test set was presented to the classifier at a time. Success rates were calculated as percentages of correct classifications among the power traces from the test sets.

Section 4.2 investigates the impact that the variation of the parameters γ and σ^2 have on the success rates. In Sect. 4.3, we analyzed the influence of varying both the numbers of traces and components on the classification. The most relevant components of the power traces were selected by the Pearson correlation coefficient approach in these two sections. Lastly, Sect. 4.4 examined whether removing outliers or using either the SOST or PCA feature selection techniques, instead of the Pearson correlation coefficient approach, would increase the success rates. All these sections provide a brief comparison of our results to TAs in terms of effectiveness.

4.2 Influence of the LS-SVM parameters

In this part, the training and test sets comprised, respectively, 3,000 and 2,000 power traces. At first, only two components, out of 2,000, were selected by picking those having the largest Pearson correlation coefficients and not belonging to the same clock cycle. Working with two-dimensional inputs allowed visualization of the decision boundaries created by the classifiers with respect to the components.

The parameters γ and σ^2, in case of using the RBF kernel, assumed the following values: 0.1, 1 and 10. After analyzing all their combinations, we verified that the success rates for the *threshold* approach were as high as 99.3%, regardless of the type of kernel used. This is because the classes assigned by the *threshold* approach are easily distinguishable.

The *intercalated* approach led to results sensitive to both the type of kernel and the tuning parameters. When using RBF kernels, the success rates showed a direct relation to σ^2. Success rates as high as 99.0%, 94.0% and 82.7% were achieved for $\sigma^2 = 0.1$, 1 and 10, respectively. Although γ did not influence the results as much as σ^2, high values of γ slightly increased the results. Linear kernels yielded success rates around 49.9%, performing as well as random guesses. The reason behind this is that a linear function cannot separate nonlinear, intercalated data.

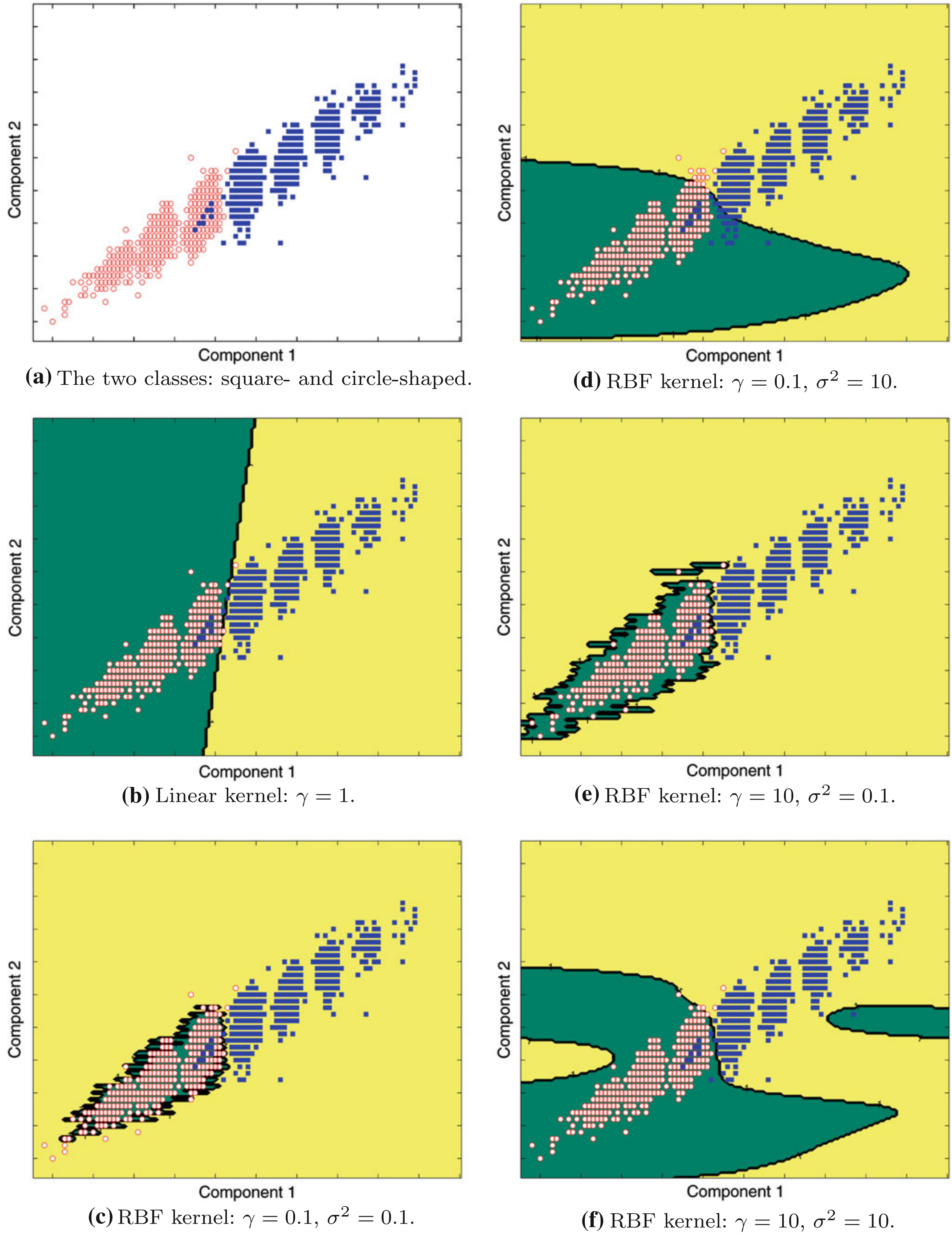
The *bit(4)* approach achieved success rates of 74.0%, regardless of both the kernel type and the values of the tuning parameters. These results, despite being worse than those from both the *threshold* and *intercalated* approaches, are yet clearly better than random, considering that only one out of eight bits from the output of the S-Box was taken into account.

Figures 1, 2 and 3 show how the classification in the *threshold*, *intercalated* and *bit(4)* approaches, respectively, occurs in relation to the tuning parameters. The horizontal and vertical axes are, respectively, the two considered power traces components. Figures 1, 2 and 3a present the two classes for each approach: the square-shaped points belong to one class, whereas the circle-shaped points belong to the other class. Figures 1, 2 and 3b concern the decision boundaries (dark lines scattering the space in two regions: light and dark colored) of the linear kernel-based classifiers for $\gamma = 1$. Varying γ for the linear kernel does not influence the decision boundary considerably. Figures 1, 2 and 3c–f concern the decision boundaries of the RBF kernel-based classifiers for the combinations of $\gamma \in \{0.1, 10\}$ and $\sigma^2 \in \{0.1, 10\}$.

We verified that underfitting arises for low values of γ, while overfitting occurs for high values of γ. This conclusion is supported by Eq. (2). When using the RBF kernel, low values of σ^2 make the decision boundary fit the data, while high values of σ^2 spread the decision boundaries.

Particularly, the orientations of the decision boundaries shown in Fig. 2d for $\gamma = 0.1$ and $\sigma^2 = 10$ seem counterintuitive. The reason behind this is twofold: (1) the low value of γ favored underfitting; and (2) the relatively high value of σ^2 favored the spread of the decision boundary in the wrong direction, which has been poorly chosen due to underfitting. However, as γ increases, the direction of the decision boundaries tend to fit the orientation of the data more suitably, as shown in Fig. 2f in comparison to Fig. 2d.

TAs using two templates led to success rates of 99.6%, 50.3%, and 73.7% for the *threshold*, *intercalated*, and *bit(4)* approaches, respectively. Except for the *intercalated* approach, results were similar to those obtained with LS-SVMs using RBF kernels. Results on the *intercalated* approach were poor, as in LS-SVMs with linear kernels. The Gaussian-based templates did not fit well the distributions of



(a) The two classes: square- and circle-shaped.

(d) RBF kernel: $\gamma = 0.1$, $\sigma^2 = 10$.

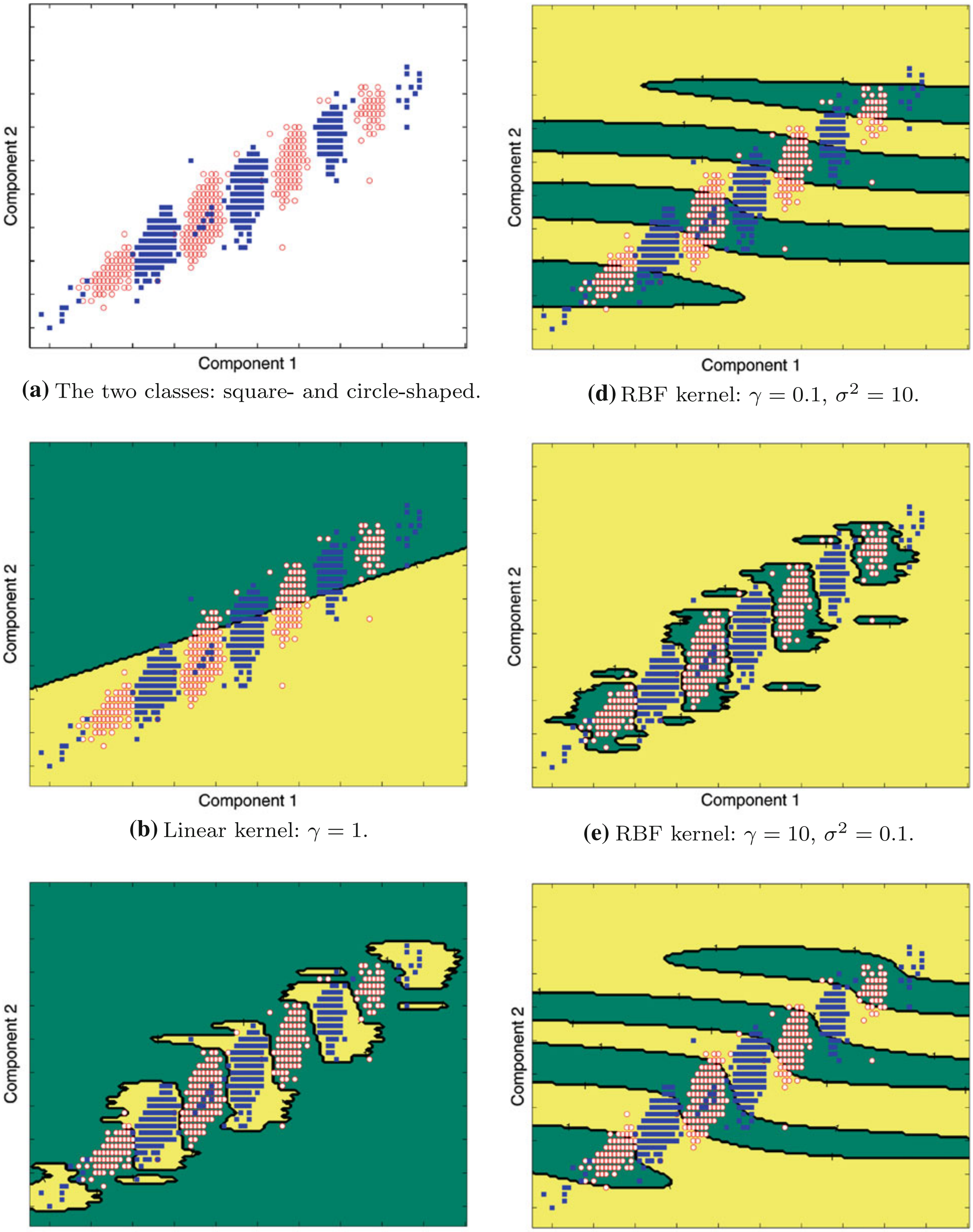
(b) Linear kernel: $\gamma = 1$.

(e) RBF kernel: $\gamma = 10$, $\sigma^2 = 0.1$.

(c) RBF kernel: $\gamma = 0.1$, $\sigma^2 = 0.1$.

(f) RBF kernel: $\gamma = 10$, $\sigma^2 = 10$.

Fig. 1 *Threshold* approach: decision boundaries of the LS-SVM classifiers



(a) The two classes: square- and circle-shaped.

(d) RBF kernel: $\gamma = 0.1$, $\sigma^2 = 10$.

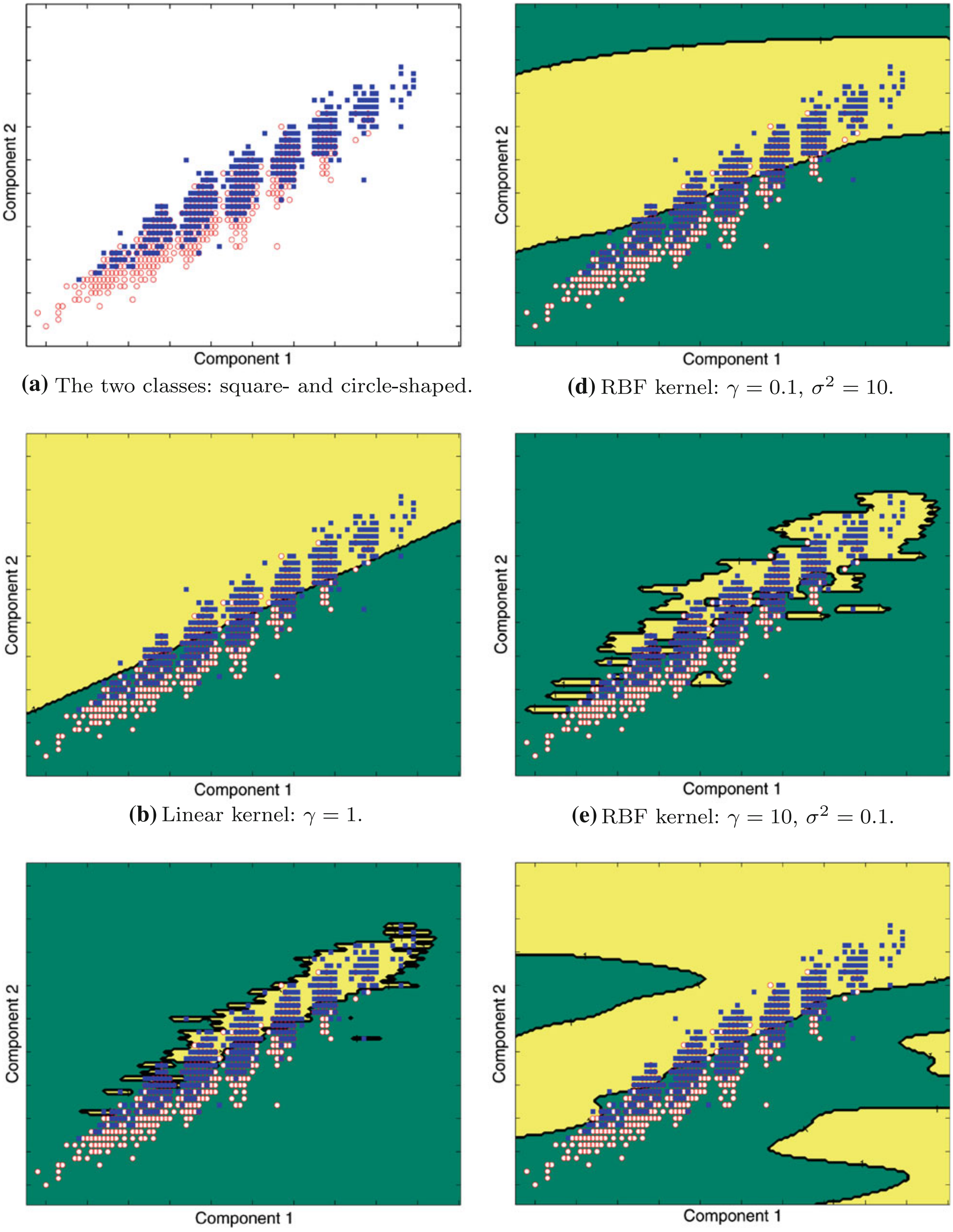
(b) Linear kernel: $\gamma = 1$.

(e) RBF kernel: $\gamma = 10$, $\sigma^2 = 0.1$.

(c) RBF kernel: $\gamma = 0.1$, $\sigma^2 = 0.1$.

(f) RBF kernel: $\gamma = 10$, $\sigma^2 = 10$.

Fig. 2 *Intercalated* approach: decision boundaries of the LS-SVM classifiers



(a) The two classes: square- and circle-shaped.

(d) RBF kernel: $\gamma = 0.1$, $\sigma^2 = 10$.

(b) Linear kernel: $\gamma = 1$.

(e) RBF kernel: $\gamma = 10$, $\sigma^2 = 0.1$.

(c) RBF kernel: $\gamma = 0.1$, $\sigma^2 = 0.1$.

(f) RBF kernel: $\gamma = 10$, $\sigma^2 = 10$.

Fig. 3 *Bit(4)* approach: decision boundaries of the LS-SVM classifiers

Table 1 Success rates (SR) for the *bit(4)* approach

Method	Number of components	SR (%)
LS-SVM: RBF	3	74.7
LS-SVM: RBF	4	67.1
LS-SVM: RBF	6	52.7
LS-SVM: Linear	3	75.5
LS-SVM: Linear	4	75.5
LS-SVM: Linear	6	75.1
TA	3	73.0
TA	4	75.8
TA	6	75.0

the *intercalated* data. This was because the *intercalated* data were actually drawn from a Gaussian mixture.

4.3 Varying the number of traces and components

Since the classes defined by the *threshold* approach are trivially separable, this section deals only with the *intercalated* and *bit(4)* approaches. The first part of the experiments consisted in varying the number of power traces within the training and test sets. Out of 5,000, 4,000, 3,000, 2,000, 1,000, and 500 randomly chosen power traces, 70% were assigned to the training set and the remaining were assigned to the test set. The parameters γ and σ^2 were kept constant. For the *intercalated* approach: $\gamma = 10$ and $\sigma^2 = 1$. For the *bit(4)* approach: $\gamma = 1$ and $\sigma^2 = 0.1$.

Interestingly enough, the results did not vary noticeably when contrasted with those from Sect. 4.2. However, in the *bit(4)* case, the success rates dropped approximately 10% when using the minimum number of power traces, though. For these first experiments with binary classification problems a relatively small amount of power traces seemed to be enough, as far as the relevant components of the power traces have been selected. TAs behaved likewise.

The second part of the experiments in this section attempted to raise the success rates (SR) in the *bit(4)* approach by increasing the number of components. Table 1 shows the result for $\gamma = 1$ and $\sigma = 0.1$ using, respectively, 3,500 and 1,500 power traces for the training and test sets.

When using the RBF kernel, the success rates dropped as more components were included. It means that the added components did not bring additional, valuable information to the classifier. On the other hand, they represented noise, making the classification task even harder. The contemplation of a search method for more suitable parameters may help improve the results. In both the linear kernel and TA cases, the success rates did not vary as a function of the considered number of components.

4.4 Outliers removal, SOST and PCA

In this part, we examined whether removing outliers or using either the SOST or PCA feature selection techniques, instead of the Pearson correlation coefficient approach, would increase the success rates.

Traces were drawn as outliers if the value of one of its selected components subtracted by the mean of this component yielded a value larger than 2.7 times the related standard deviation. The threshold 2.7 was chosen ad hoc. It assured 99.3% of data coverage. Still focusing on the *bit(4)* approach, this preprocessing technique did not improve the previous results.

Concerning the SOST, even with this feature selection technique not choosing exactly the same components as those from the previous section, the success rates also did not change in comparison with the prior results.

The scenario did not change remarkably when using PCA either. By keeping the features representing more than 98.0% of the variance of all the features, the results remained similar to the ones from Sect. 4.2. When saving, respectively, 98.0%, 99.0%, and 99.5% of the variance, the number of selected features were 4, 5, and 7, respectively.

5 Conclusions

Side-channel analysis is a powerful and feasible way of attacking secure systems. In this novel work, we started a comprehensive study on the application of machine learning in SCA. One of the main motivations to use machine learning techniques is their outstanding results in different domains.

We focused on power analysis. Power traces often leak meaningful amounts of information about the processed cryptographic key. Power traces extracted from a software implementation of the AES without countermeasures were used for initial analysis. To help us better understand the behavior of the machine learning technique, we created three simple arrangements of class distributions: *threshold*, *intercalated*, and *bit(4)*. All of them contained two different classes.

In this work, we chose the LS-SVM as our classification technique. We performed three experiments. Firstly, experiments examining the influence of the parameters of the machine learning technique on the accuracy of the classifiers were performed on data sets with obvious Hamming weight leakage. Secondly, we investigated the consequences of varying both the number of traces and their components. Thirdly, we ran tests considering feature selection (Pearson correlation coefficient approach, SOST and PCA) and preprocessing (outliers removal) techniques. The results were compared to the relatively simple, strong and well-known TAs.

The success rates obtained in the *threshold* approach were high regardless of the type of kernel used, mainly because the data set was easily separable. Results for the *intercalated* approach showed that the RBF kernel is more suitable for nonlinear problems. Since the choice of the LS-SVM parameters directly affected the results, applying an automatic parameter tuning technique should be considered in future works.

The influence of varying the number of power traces for training could only be noticed when using as few as 500 power traces, which yielded lower success rates. When increasing the number of components of the inputs of the classifiers, the results got worse due to lack of valuable information within the additional components. Likewise, the application of a preprocessing technique did not contribute. All the feature selection techniques inspected in this work performed similarly.

TAs performed similarly to LS-SVM classifiers using linear kernels. This similarity was clarified in the analysis on the *intercalated* approach, which generated comparable (poor) results for both LS-SVM classifiers with linear kernels and TAs. LS-SVM classifiers with RBF kernels were able to produce reasonable results in the *intercalated* approach. They outperformed TAs in this case.

6 Future works

LS-SVM It would be interesting to look further at the functioning of kernel-based learning algorithms. It may be that other possibly tailored kernels improve the results. As a short reminder, the kernel is responsible for mapping the data into a feature space to facilitate the distinction between different classes.

Efficiency comparison Future works should include an assessment of the efficiency of machine learning approaches in comparison to TAs. We observed that LS-SVMs are significantly heavier than TAs, especially in the training phase. This observation is likely to hold for other elaborated machine learning techniques.

Other cryptographic algorithms and implementations Any implementation of any cryptographic algorithm may be attacked. The more attacks are performed, the more can be learned about using machine learning techniques in SCA. Here the attacker is free to attack either software or hardware implementations of cryptographic algorithms such as the AES, 3DES, etc.

Attacks on protected implementations A more realistic attack scenario should consider attacks on protected implementations. Some designers make use of countermeasures such as masking [6], for instance. Also, more noisy data should be considered.

Other machine learning techniques One is basically free to use any off-the-shelf machine learning technique or even some other approach. Examples of other machine learning techniques include artificial neural networks [3,11]. Machine learning techniques that learn in an unsupervised way (by not making use of any labeled data, e.g. clustering) are able to perform non-profiled attacks, which were out of the scope of this work.

References

1. Aizerman, M.A., Braverman, E.A., Rozonoer, L.: Theoretical foundations of the potential function method in pattern recognition learning. In: Automation and Remote Control, vol. 25, pp. 821–837 (1964)
2. Backes, M., Dürmuth, M., Gerling, S., Pinkal, M., Sporleder, C.: Acoustic side-channel attacks on printers. In: USENIX, p. 20 USENIX Association, USA (2010)
3. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, USA (1995)
4. Brabanter, K.D., Karsmakers, P., Ojeda, F., Alzate, C., Brabanter, J.D., Pelckmans, K., Moor, B.D., Vandewalle, J., Suykens, J.: LS-SVMlab toolbox user's guide version 1.7. http://www.esat.kuleuven.be/sista/lssvmlab/ (2010)
5. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: CHES, vol. LCNS 2523, pp. 13–28. Springer, USA (2002)
6. Coron, J.S., Goubin, L.: On boolean and arithmetic masking against differential power analysis. In: CHES, pp. 231–237. Springer, London (2000)
7. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**, 273–297 (1995)
8. Gandolfi, K., Naccache, D., Paar, C., G, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: concrete results. In: CHES, vol. 2162, pp. 251–261. Springer, Berlin (2001)
9. Gestel, T.V., Suykens, J., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., Moor, B.D., Vandewalle, J.: Benchmarking least squares support vector machine classifiers. Mach. Learn. **54**, 5–32 (2004)
10. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. stochastic methods. In: CHES, vol. LCNS 4249, pp. 15–29. Springer, Japan (2006)
11. Haykin, S.: Neural Networks: A Comprehensive Foundation. Macmillan College Publishing Company, Englewood Cliffs (1998)
12. Jolliffe, I.T.: Principal Component Analysis. Springer, Berlin (1986)
13. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Crypto 96—Advances in Cryptology, pp. 104–113. Springer, UK (1996)
14. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Crypto 99—Advances in Cryptology. LCNS, vol. 1666, pp. 388–397. Springer, USA (1999)
15. Messerges, T.S., Dabbish, E.A., Sloan, R.H.: Examining smart-card security under the threat of power analysis attacks. IEEE Trans. Comput. **51**, 541–552 (2002)
16. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
17. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In: Proc. Smart Card Programming and Security. LCNS, vol. 2140, pp. 200–210 (2001)
18. Rechberger, C., Oswald, E.: Practical template attacks. In: WISA, vol. 3325, pp. 440–456. Springer, Korea (2004)
19. Rivest, R.L.: Cryptography and machine learning. In: Advances in Cryptology ASIACRYPT, pp. 427–439. Springer, Berlin (1993)
20. Suykens, J., Gestel, T.V., Brabanter, J.D., Moor, B.D., Vandewalle, J.: Least Squares Support Vector Machines. World Scientific, Singapore (2002)