Efficient Template Attacks Based on Probabilistic Multi-class Support Vector Machines

Timo Bartkewitz and Kerstin Lemke-Rust

Department of Computer Science,
Bonn-Rhine-Sieg University of Applied Sciences,
Grantham-Allee 20, 53757 Sankt Augustin, Germany
{timo.bartkewitz,kerstin.lemke-rust}@h-brs.de

Abstract. Common template attacks are probabilistic relying on the multivariate Gaussian distribution regarding the noise of the device under attack. Though this is a realistic assumption, numerical problems are likely to occur in practice due to evaluation in higher dimensions. To avoid this, a feature selection is applied to identify points in time that contribute most information to an attack. An alternative to common template attacks is to apply machine learning in form of support vector machines (SVMs). Recent works brought out approaches that produce comparable results, respectively better in the presence of noise, but still not optimal in terms of efficiency and performance. In this work we show how to adapt the SVM template approach in order to considerably reduce the effort while carrying out the attack and how to better exploit the side-channel information under the assumption of an attack model with a strict order, *e.g.* Hamming weight model.

Keywords: Power Analysis, Template Attacks, Machine Learning, Support Vector Machines.

1 Introduction

Side-channel attacks are still, even after years of intensive research, a serious threat to cryptographic devices. New attacks challenge new algorithms and respective countermeasures. Generally, side-channel attacks rely on the assumption that any electronic device provides physically observable information on secrets, usually a cryptographic key, embedded in the device. The spot from which the information is extracted is referred to as a side-channel. There exist several distinct kinds of side-channels, for instance the power consumption, electromagnetic emanation, or timings. In this work we focus on template attacks (TAs) [5], a powerful side-channel attack since they are supposed to retrieve the most information of a side-channel leakage. TAs require a profiling phase to model the noise of the device, assuming the noise to be multivariate Gaussian distributed. In the subsequent characterization phase any dependency can be found using a likelihood approach for similarities between different points in time of recorded power consumption traces. Therefore, TAs are dependent on these points in time which are assumed to contain the most information given by the maximum key-dependent variance. Besides the selection of such points which is often denoted *feature selection*, one is also concerned with potential numerical problems and the question whether the assumed noise model is adequate or not in order to mount a successful TA.

To overcome these issues recent works investigated alternatives to the multivariate Gaussian approach. Machine learning in the form of support vector machines (SVMs) is one of these promising alternatives. The SVM is a linear binary classifier that decides to which of two classes an input vector belongs, based on classified training data. Further, the SVM is independent of a certain noise distribution. In [12] the authors focused on SVMs amongst other machine learners and present attacks that predict key bits of a DES implementation. In [10, 11] the authors exclusively focused on SVMs concentrating on the applicability for template attacks. However, they did not provide an attack. In [9] the authors extend the SVM approach from a single-bit model to a multi-bit model and consequently introduce probabilistic multi-class SVMs. They showed that the SVM based template attack outperforms the Gaussian approach in the presence of noise. Nevertheless, their approach is not optimal in terms of efficiency and exploited side-channel leakage.

Our contribution: In this work we carry the SVM template attacks forward. We first show how a tailored multi-class strategy can considerably reduce the effort during the profiling and characterization phase. Second, we show how to better exploit the side-channel leakage, including the introduction of a dedicated feature selection, and compare it against several other TA approaches. We therefore assume an attack model with a strict order, *e.g.* Hamming weight model.

Organization of the paper: This paper is organized as follows: Section 2 briefly introduces template attacks based on the Gaussian approach. Section 3 provides a broad overview on support vector machines. In Section 4 we describe how SVM based template attacks can be improved by means of efficiency and performance. Finally, Section 5 reports our experimental results before we conclude in Section 6.

2 Template Attacks

Template attacks usually consist of three steps. The first step is to select points in time (often called *points of interest* or *features*) that are supposed to contain a considerable proportion of the leakage information. Afterwards templates are built involving the power consumption of a reference device, similar to the target device, that is under the full control of the attacker. Eventually, the attack on the target device is carried out by matching its power consumption leakage to the templates.

Feature Selection. The selection of points of interest within power traces is the first issue in TAs we are concerned with. There are several methods to obtain a set of points that could lead to a successful attack. Primarily, the points are selected according to their key-dependent variability, including known-key DPA [18], pair-wise distance to the mean vectors [5], or using the sum of squared pair-wise T-differences [8]. A more systematic approach is the principal subspace-based TA where the principal component analysis (PCA) is applied to transform the side-channel data into a low-dimensional subspace, figuring out the optimal linear combination of points in time which show maximum variance with respect to the side-channel leakage [1].

Template Building. The application of templates implies two successive phases. In the first phase templates are built according to N_p selected points of interest from several measured power traces $\{\mathbf{t}_{d,k}^{i}\}_{i=1}^{N_\mathbf{t}}$ that are correlated to a function which involve both, known input data d and a key k, respectively a part of it. The traces are assumed to be drawn from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{t}_{d,k}^{i}|\boldsymbol{\mu}_{d,k}, \boldsymbol{\Sigma}_{d,k})$. Therefore, a single template $\boldsymbol{\tau}_{d,k}$ is equivalent to an estimation of the mean $\boldsymbol{\mu}_{d,k}$ and the covariance matrix $\boldsymbol{\Sigma}_{d,k}$ based on the selected points, and corresponding to different pairs of $\{d, k\}$.

Key Recovery Attack. In the second phase, given a new power trace $\mathbf{t}_{d,k}^{\mathrm{new}}$ to be characterized, the multivariate Gaussian probability density function

$$p(\mathbf{t}_{d,k}^{\mathrm{new}}|\boldsymbol{\tau}_{d,k}) = \frac{1}{\sqrt{(2\pi)^{N_p}|\boldsymbol{\Sigma}_{d,k}|}} \exp\left(-\frac{1}{2}(\mathbf{t}_{d,k}^{\mathrm{new}} - \boldsymbol{\mu}_{d,k})^\top \boldsymbol{\Sigma}_{d,k}^{-1}(\mathbf{t}_{d,k}^{\mathrm{new}} - \boldsymbol{\mu}_{d,k})\right) \quad (1)$$

is evaluated for each template. The maximum likelihood approach provides the best fit, hence the higher the probability density the better the trace $\mathbf{t}_{d,k}$ fits the respective template.

In order to avoid numerical problems in practice, mainly due to the inversion of $\boldsymbol{\Sigma}_{d,k}$, one can omit the covariances (off-diagonal values of $\boldsymbol{\Sigma}_{d,k}$) to obtain so called *reduced templates* [14]. This leads to a univariate approach since (1) can then be rewritten as the product of the probability densities at each point of interest

$$p(\mathbf{t}_{d,k}^{\mathrm{new}}|\boldsymbol{\tau}_{d,k}) = \prod_{i=1}^{N_p} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{1}{2}\frac{(t_{i,d,k}^{\mathrm{new}} - \mu_{i,d,k})^2}{\sigma_i^2}\right) \quad (2)$$

considering them as being independent and thus uncorrelated.

In practice it is often not sufficient to recover the key, or a part of it, based on a single trace to be characterized but based on a few traces. Usually, we apply Bayes' theorem [14]

$$p(k^*|\mathbf{t}_{d,k}^{\mathrm{new}}) = \frac{p(\mathbf{t}_{d,k}^{\mathrm{new}}|\boldsymbol{\tau}_{d,k}) \cdot p(k^*)}{p(\mathbf{t}_{d,k}^{\mathrm{new}})} \quad (3)$$

in order to determine by which key the traces to be characterized were generated. Here, $p(k^*)$ is the prior probability and $p(k^*|\mathbf{t}_{d,k}^{\mathrm{new}})$ the posterior probability of each key candidate k^*.

3 Binary Support Vector Machines

Support vector machines are suitable for solving classification, regression, and pattern detection problems and belong to the category of *sparse kernel machines.* Originally described in [20], the SVM is a non-probabilistic, linear, binary-class decision machine whose output is a class label. The SVM is related to supervised learning methods whose determination of the model parameters correspond to convex optimization problems. This section follows the explanations in [2].

3.1 Mathematical Background of Binary SVMs

A binary-class classification problem can be described by a linear discriminant function of the form

$$y(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b \tag{4}$$

where $\mathbf{w}$ is a *weight vector*, and b is a *bias.* An input vector $\mathbf{x}$ is assigned to class C^- if $y(\mathbf{x}) < 0$ and to C^+ otherwise. Hence, the decision boundary corresponds to the $(D-1)$-dimensional hyperplane within the D-dimensional input space, *i.e.* $y(\mathbf{x}) = 0$. Since $\mathbf{w}^\top \mathbf{x} = 0$ for every $\mathbf{x}$ lying on the decision boundary, $\mathbf{w}$ is orthogonal to every vector on the decision boundary and hence the normal vector of the hyperplane. With the same argument, bias $b = -\mathbf{w}^\top \mathbf{x}$ for every $\mathbf{x}$ on the decision boundary. Suppose the training set consists of N input vectors (row vectors) $\mathbf{x}_1, \ldots, \mathbf{x}_N$ (vectors with an index belong to the training set in the remainder) where each vector is associated to a class label $c_i \in \{-1, 1\}$. New vectors $\mathbf{x}$ are accordingly classified by the sign of $y(\mathbf{x})$. For the moment, it is assumed that the training set is linearly separable within the input space D, which means we can find a pair $(\mathbf{w}, b)$ such that (4) satisfies $c_i y(\mathbf{x}_i) > 0$ for all training vectors. That is, every training vector $\mathbf{x}_i$ is correctly classified. Naturally, we can find several pairs that separate the training set exactly but not every solution will give the smallest generalization error [2] that states the performance of classifying new vectors . In support vector machines this is solved by introducing the approach of the margin which embodies the smallest distance between the decision boundary to any input vector (Fig. 1). The best solution is given by the pair $(\mathbf{w}, b)$ for which this margin is maximized. The orthogonal distance of a vector $\mathbf{x}$ to the hyperplane is given by $y(\mathbf{x}) / \|\mathbf{w}\|$ ($\|\bullet\|$ denotes the *Euclidean norm*) and under the general constraint $c_i y(\mathbf{x}) > 0$ the maximum margin can be achieved by finding

$$\arg\max_{\mathbf{w},b}\{\min_i[c_i(\mathbf{w}^\top \mathbf{x}_i + b)]\}. \tag{5}$$

Finding a direct solution would be too complex. Since rescaling of $\mathbf{w}$ and b does not affect the distance from any input vector to the hyperplane, we set

$$c_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, N \tag{6}$$

which means that for vectors that lie on the margin around the decision boundary the equality holds. Consequently, the optimization problem has been reduced

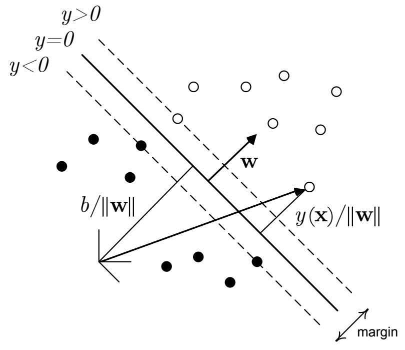

Fig. 1. Geometry of the separating hyperplane in support vector machines

to maximize $\|\mathbf{w}\|^{-1}$ which is equivalent to minimizing $\|\mathbf{w}\|^2$. This is a *quadratic programming* problem that can be solved by applying the method of *Lagrange multipliers* a_i, with respect to the *Karush-Kuhn-Tucker* (KKT) conditions [2]. The optimal solution of this Lagrangian optimization problem yields a representation of (4), *s.t.* $\mathbf{w} = \sum_{i=1}^{N} a_i c_i \mathbf{x}_i$ where the vectors $\mathbf{x}_i$ for which $a_i > 0$ are called *support vectors*. Hence, to classify new vectors $\mathbf{x}$ we obtain

$$y(\mathbf{x}) = \sum_{i \in \mathcal{S}}^{N} a_i c_i \mathbf{x}_i^{\top} \mathbf{x} + b \quad \text{where} \quad b = \frac{1}{N_{\mathcal{S}}} \sum_{i \in \mathcal{S}} (c_i - \sum_{j \in \mathcal{S}} a_j c_j \mathbf{x}_i^{\top} \mathbf{x}_j) \tag{7}$$

and $\mathcal{S}$ is the set of indices of the support vectors, respectively $N_{\mathcal{S}}$ the number of support vectors.

3.2 Non-linear Classification: Introduction of a Kernel

So far it was assumed that the training set is linearly separable within the input space D. If that does not hold the training set may be separable in the higher dimensional *feature space* $F > D$, not to be confused with feature selection. Therefore, the input vectors are transformed into that feature space by $\Phi(\mathbf{x})$ which gives a vector product of the form $\Phi(\mathbf{x}_1)^{\top}\Phi(\mathbf{x}_2)$ for $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^D$. A direct solution is computationally very intensive. Hence, a kernel function is applied that behaves exactly like the vector product in F without even knowing the concrete feature space, but also without having influence on the resulting dimension.

3.3 Non-separable Case: Introduction of a Soft-Margin

A linear separation, in the input space or in the feature space, can lead to a poor generalization (large generalization error) in the case of overlapping class distributions. Therefore, it makes sense to allow for misclassification of some training vectors to achieve a separation anyway. To do so, slack variables and a trade-off parameter γ, often denoted *box constraint*, are introduced to penalize misclassification leading to two kinds of support vectors (see [2] for details). Those where $a_i < \gamma$, support vectors which lie on the margin, and those where $a_i = \gamma$ which are support vectors that are either correctly classified but inside the margin or misclassified. For $\gamma \to \infty$ the penalty prohibits misclassified vectors and thus recovers the strict margin.

3.4 SVM Training and Classification

Training a support vector machine means solving the Lagrangian optimization problem for the given training set. There exist a specific approach called *sequential minimal optimization* (SMO) [16] that breaks down the optimization problem into many smallest problems where each of which only considers two Lagrange multipliers at a time. The Lagrange multipliers are jointly optimized under a linear equality constraint. The subsequent multipliers to be optimized are then chosen heuristically. The SVM classification is done through the evaluation of (7) making use of the parameters of the SMO training.

4 Template Attacks Using Support Vector Machines

The approach is similar to common template attacks. The posterior key probabilities are successively updated with each characterization trace applying Bayes' theorem. Since TAs are a multi-class classification problem it is essential to turn the actual binary SVM into a probabilistic multi-class SVM. Contrary to previous work [9] which follows a general probabilistic approach, we subsequently present probabilistic multi-class SVMs tailored to fit template attacks.

4.1 Probabilistic Support Vector Machines

In order to use SVMs in an aggregated probabilistic approach, probabilistic decisions of the class label c for new vectors are necessary. Maintaining the sparseness property of SVMs it is proposed in [17] to fit a logistic sigmoid function to the outputs $y(\mathbf{x})$ of an already trained binary SVM to give the posterior conditional probability

$$p(c = 1|\mathbf{x}) = \frac{1}{1 + \exp(A \cdot y(\mathbf{x}) + B)} \tag{8}$$

that $\mathbf{x}$ belongs to the class $c = 1$. Clearly, $p(c = -1|\mathbf{x}) = 1 - p(c = 1|\mathbf{x})$. A second training set should be involved to avoid severe overfitting (this will be discussed in Section 5). The parameters A and B are found by minimizing the cross-entropy error of the training set

$$\arg\min_{A,B} - \sum_{i=1}^{N} t_i \log(p_i) + (1 - t_i) \log(1 - p_i) \tag{9}$$

where $t_i = (\mathrm{sign}[y(\mathbf{x}_i)] + 1)/2$ and $p_i = p(c = 1|\mathbf{x}_i)$. Nevertheless, solving this optimization problem in a numerical stable way is proposed in [13].

4.2 Our Probabilistic SVM Multi-class Approach

Previous work [7] investigates methods based on two strategies to turn a binary SVM into a multi-class SVM. The *one-versus-all* strategy uses binary SVMs for separating one multi-class from the joint set of all other multi-classes, whereas the *one-versus-one* strategy applies binary SVMs for pair-wise distinguishing the multi-classes. In [9] the authors preferred the *one-versus-one* strategy in order to mount their attacks. In this work, we suggest our own method tailored for template attacks with regards to the attack model.

Suppose the TA classification problem implies M distinct classes. Then, given a new input vector $\mathbf{x}$, we aim for posterior conditional probabilities $p(\Omega_l|\mathbf{x})$ for $l = 1, \ldots, M$ and Ω_l is the l_{th} multi-class label. The classification is enabled by a training set $\widetilde{\mathbf{X}}$ (traces) where for each training vector $\tilde{\mathbf{x}}_j$ for $j = 1, \ldots, N$ the correct multi-class label Ω_l, relying on a certain attack model, is known.

Attack Model with a Strict Order. We assume that the side-channel leakage leads to splitting the measurement samples into a strict order according to the known multi-class labels at the leakage-dependent points in time. More precisely, let $x_{\Omega_l,t}$ be an arbitrary scalar at point t of a trace that belongs to label Ω_l, then for each leakage-dependent point t we assume a strict ordering of the labels, *i.e.* either $x_{\Omega_1,t} < x_{\Omega_2,t} < \ldots < x_{\Omega_M,t}$ or, alternatively, $x_{\Omega_1,t} > x_{\Omega_2,t} > \ldots > x_{\Omega_M,t}$. A well-known example of such an ordered leakage is the Hamming weight model [14]. With such an attack model we train $M-1$ SVMs using the training set $\widetilde{\mathbf{X}}$ and introduce binary-class helper labels $c_{i,j}$ to the training vectors, such that

$$c_{i,j} = \begin{cases} 1, & \{\tilde{\mathbf{x}}_j | \tilde{\mathbf{x}}_j \text{ belongs to } \Omega_l \text{ with } l \leq i \\ -1, & else \end{cases}, \quad \begin{array}{l} j = 1, \ldots, N \\ l = 1, \ldots, M \end{array} \tag{10}$$

when the i_{th} SVM is under training, $i = 1, \ldots, M-1$. These helper labels convert the attack model multi-classes to binary-classes as requested by the SVM classification model. Using the i_{th} SVM afterwards to classify a new vector $\mathbf{x}$ we get the multi-class overlapping probabilities $p(\bigcup_{l=1}^{i} \Omega_l|\mathbf{x})$. That is, the i_{th} SVM gives the probability that $\mathbf{x}$ belongs to the classes before the i_{th} hyperplane. Figuratively, we construct the hyperplanes between the multi-class labels from the left to the right as depicted in Figure 2. Recalling that the probabilities rely on a distance measure between $\mathbf{x}$ and the separating hyperplane, each consecutive probability $p(\bigcup_{l=1}^{i+1} \Omega_l|\mathbf{x}), p(\bigcup_{l=1}^{i+2} \Omega_l|\mathbf{x}), \ldots, p(\bigcup_{l=1}^{M-1} \Omega_l|\mathbf{x})$ is even higher once $\mathbf{x}$ was classified to belong to a multi-class before the i_{th} hyperplane with a non-negligible probability (*cf.* Fig. 2). This is due to the fact that the distance grows in a positive manner and thus the probability grows since the binary-class separation regarding $\mathbf{x}$ becomes even clearer. However, this is of course an undesired result. We can overcome this drawback by training, again, $M-1$ SVMs but this time starting with the last multi-class label Ω_M, *i.e.* using (10) with reversed signs. With this approach, going from the left to the right first and then vice versa, we surround the correct multi-class label by two consecutive hyperplanes. In fact we do not need to train new SVMs since the separating hyperplanes

$$\begin{array}{ccc|cccccc}
1 & 1 & 1 & -1 & -1 & -1 & -1 & & -1 \\
\Omega_1 & \Omega_2 & \Omega_3 & \Omega_4 & \Omega_5 & \Omega_6 & \Omega_7 & \dots & \Omega_M
\end{array}$$

Fig. 2. For instance, the *third* support vector machine is trained which corresponds to the hyperplane separating $\bigcup_{l=1}^{3} \Omega_l$ and $\bigcup_{l=4}^{M} \Omega_l$. The binary-class helper labels $c_{3,j}$ for the training vectors $\mathbf{x}_j$ are given on top. Training vectors that belong to the multi-classes before the hyperplane are classified with helper label 1, all others with -1.

did not change, but merely use the complementary probabilities deferred by one multi-class due to the surrounding. Suppose $\mathbf{x}$ indeed belongs to Ω_i, then $\mathbf{x}$ is right-bounded by the hyperplane i and left-bounded by the hyperplane $i-1$ and thus related to the probabilities $p(\bigcup_{l=1}^{i} \Omega_l|\mathbf{x})$ and $1 - p(\bigcup_{l=1}^{i-1} \Omega_l|\mathbf{x})$. Hence, we suggest using

$$p(\Omega_i|\mathbf{x}) = p(\bigcup_{l=1}^{i} \Omega_l|\mathbf{x}) \cdot \left[1 - p(\bigcup_{l=1}^{i-1} \Omega_l|\mathbf{x})\right] \tag{11}$$

$$= \frac{1 - \frac{1}{1+\exp(A_{i-1}\cdot y_{i-1}(\mathbf{x})+B_{i-1})}}{1 + \exp(A_i \cdot y_i(\mathbf{x}) + B_i)}, \quad 1 < i < M, \tag{12}$$

$$p(\Omega_1|\mathbf{x}) = p(\bigcup_{l=1}^{1} \Omega_l|\mathbf{x}) = \frac{1}{1 + \exp(A_1 \cdot y_1(\mathbf{x}) + B_1)}, \tag{13}$$

$$\text{and } p(\Omega_M|\mathbf{x}) = 1 - p(\bigcup_{l=1}^{M-1} \Omega_l|\mathbf{x}) = 1 - \frac{1}{1 + \exp(A_{M-1} \cdot y_{M-1}(\mathbf{x}) + B_{M-1})} \tag{14}$$

being the posterior conditional class probabilities.

Attack Model without a Strict Order. Technically speaking, if an attack model with a strict order is not applicable, *i.e.* scalars that belong to different multi-classes are equal on average, a separation is impossible no matter which method is used. Nevertheless, in practice one could try to combine (leave out) equivalent multi-classes or use a one-versus-one strategy. The latter case means separating each pair (Ω_i, Ω_j) for $i < j \leq M$ which in turn results in training $M(M-1)/2$ SVMs. The posterior conditional class probabilities are then given by

$$p(\Omega_i|\mathbf{x}) = \prod_{j \neq i} p[(\Omega_i, \Omega_j)|\mathbf{x} \text{ belongs to } \Omega_i]. \tag{15}$$

4.3 SVM Based Templates

As in the multivariate Gaussian approach, templates are built on the reference traces first, here called the training set, by training $M-1$ or $M(M-1)/2$ support vector machines where M denotes the number of classes according to our attack model. Afterwards, we fit the sigmoid function with classification values from the SVMs involving a second set of reference traces. Hence, a single template contains the Lagrange multipliers $\mathbf{a}_i$, the support vectors $\mathbf{X}_{\mathcal{S}i} \subset \widetilde{\mathbf{X}}$, and the bias b_i plus the values A_i and B_i for the templates $i, \ldots, M-1$, respectively for $i, \ldots, M(M-1)/2$. Please note that in the SVM approach templates are to be characterized by the class separators and not by the class representatives.

4.4 Considering Feature Selection

Primarily, a prior feature selection is a dimensionality reduction to help figuring out the most discriminative features in a given data set. However, this generally means a loss of information that in turn affects the prediction performance of classifiers.

For the Gaussian TA approach a priorly executed feature selection is certainly essential since it avoids numerical problems in practice which render the evaluation of probability densities impossible. Further, it is assumed that the exploitable side-channel leakage is hidden locally in the variability of only a few points in time with respect to such probability densities [14]. Hence, the loss of information is accepted and considered as loss of noise, respectively loss of information that marginally contributes to the attack.

This looks different for the SVM approach where we aim for inter-class separation instead of intra-class densities. On the one side numerical problems due to high dimensioned data do not occur and on the other dimensionality reduction methods such as PCA likely jeopardize the optimal performance of SVMs in other applications [21]. In contrast to previous works [9–12] we suppose using the linear kernel with a dedicated subsequent feature selection, called *normal-based feature selection*, is optimal while presuming a linear attack model in template attacks (*cf.* Section 4.2). The normal-based feature selection retains points in time according to the weight vector $\mathbf{w}$ (*cf.* Section 3.1). It can be shown [3] that a feature j which corresponds to a higher absolute value $|w_j|$ are more influential in determining the optimal margin and thus improves classification performance. Since we train several SVMs according to the attack model we disregard features by setting the respective weights of $\mathbf{w}$ to zero instead of removing them from the data set. One may argue that prior feature selection has the same effect but the Lagrange multipliers $\mathbf{a}_i$ that significantly determine the weight vector are still found using the complete data set.

5 Experimental Results

For our experiments we used a Microchip PIC18F2520 microcontroller [15] running at 3.68 MHz. The power traces were acquired with a PicoScope *5203* and a sampling rate at 125 MS/s using a 1 Ω-resistor in the ground line. The traces were compressed with peak extraction [14] representing the full substitution layer (S-boxes) in the first round of the AES (Advanced Encryption Standard). Therefore, we chose the attack model to be the Hamming weight of the S-box output. We thus have nine classes with a strict order (*cf.* Sec. 4.2).

To produce comparable results to [9] we choose the *guessing entropy* [19] to evaluate the attack performance. It states the average position of the correct key within a descending ranking of the probabilities of all possible keys. However, in order to introduce our adaptations we used our own SVM implementation as described throughout this paper (see Appendix A for pseudo code) instead of the C-SVC implementation of the LIBSVM library [4]. C-SVC also applies SMO for training and the same probability model for classification, thus both implementations are comparable.

In the following, we validate our SVM template attack against variants of it, the SVM TA from [9], the common TA, and the common TA with prior PCA. Our approach implies the here proposed multi-class method and the linear kernel with the subsequent normal-based feature selection. As variants we replaced the normal-based feature selection with prior feature selection, namely known-key DPA (kkDPA) where the points in time with the highest correlation were taken, respectively with the application of PCA. Further, we include the common template attack with both known-key DPA and prior PCA. The SVM TA from [9] uses the RBF kernel and known-key DPA. They also suggest an empirical determined cost factor (box constraint) of $\gamma = 10$, respectively $\gamma = 1$ for noisy measurements, and an empirical determined termination criterion of 0.02 that states the fraction of vectors that are allowed to violate the KKT conditions. In our experiments, however, we involve $\gamma = 1$ and termination criterion of zero, as recommended in [6], except for the attack from [9]. In order to simulate noisy measurements we add Gaussian noise to the power traces, in particular Gaussian noise with a standard deviation of $\sigma_{n_g} = 5$. We determined the intrinsic noise of our measured power traces to be $\sigma_{n_0} \approx 0.7$.

Initially, we want to show how to disregard features with the help of normal-based feature selection. As can be seen in Figure 3 the weights are linearly

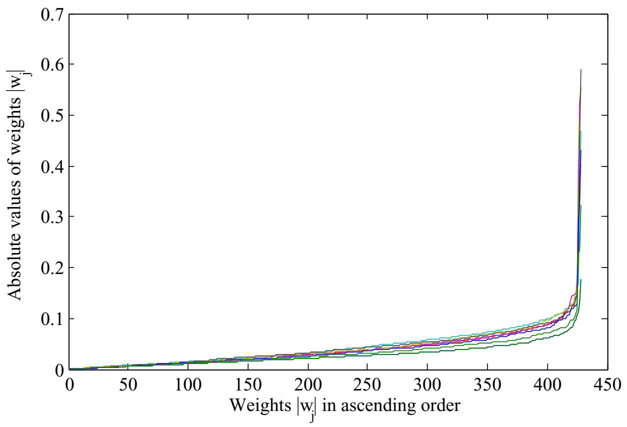

Fig. 3. Absolute values of the weights in ascending order. The weight vectors were obtained from training 8 SVMs (HW attack model).

increasing except the last few weights which increase exponentially. These are the weights we retain and thus all the others were set to zero. In our experiments we retain 8 weights. We also used 8 components from PCA and 8 points in time with the highest correlation from known-key DPA.

Our performance comparison considers each template attack involving the required profiling, respectively characterization traces to reach a guessing entropy of one. Thus, it states the minimum effort to always recover the correct key with respect to our experiments. Furthermore, we depict the guessing entropies of the attacks while our attacks possesses a guessing entropy of one.

Table 1. Comparison of template attacks depicting the required amount of characterization traces along an increasing profiling base (traces per HW) to reach a guessing entropy of one. The lower table depicts the guessing entropies while our TA reaches a guessing entropy of one. The traces were involved with their intrinsic noise $\sigma_{n_0} \approx 0.7$.

Profiling	SVM based TA				Common TA	
base	Our TA	linear & kkDPA	linear & PCA	Heuser et al. [9]	kkDPA	PCA
10	33	98	71	289	–	–
20	22	67	33	147	92	53
40	21	63	26	139	63	37
60	19	61	21	121	59	23
80	15	44	17	116	55	19
100	13	39	15	111	51	16
10	1	2.59	1.17	18.54	–	–
20	1	2.57	1.15	16.75	7.51	1.62
40	1	2.58	1.15	18.72	6.58	1.15
60	1	2.58	1.08	18.10	4.91	1.10
80	1	2.08	1.08	17.53	3.46	1.10
100	1	2.08	1.08	18.43	3.74	1.08

Table 1 shows the results considering the original traces. It is observable that template attacks based on SVMs perform better than common TAs. As expected each attack requires less characterization traces with an increasing profiling base where best performance is almost reached with a profiling base containing 100 traces per Hamming weight. However, with a too small profiling base (10 traces per HW) the common attacks fail due to numerical problems caused by the matrix inversion with the Gauss-Jordan algorithm. Our approach performs well, especially with a small profiling base, whereas the attack proposed in [9] using the RBF-kernel performs only suboptimal. This observation is even more aggravated when comparing the guessing entropies. When our TA reaches a guessing entropy of one the other attacks reduce the key space to at most four except the RBF kernel based attack that reduces the key space to about 18.

Next, we evaluate the performance in the presence of higher noise. Table 2 depicts that in this case PCA is not the optimal choice for feature selection. Both TA approaches lead to inferior results when using PCA instead of known-key

Table 2. Comparison of template attacks depicting the required amount of characterization traces along an increasing profiling base (traces per HW) to reach a guessing entropy of one. The lower table depicts the guessing entropies while our TA reaches a guessing entropy of one. The traces were involved with added noise σ_{n_g}, thus $\sigma_{n_1} \approx 5.7$.

Profiling	SVM based TA				Common TA	
base	Our TA	linear & kkDPA	linear & PCA	Heuser et al. [9]	kkDPA	PCA
10	81	121	149	1100	–	–
20	62	93	112	365	650	920
40	56	84	94	312	158	344
60	51	73	84	153	112	146
80	46	62	79	144	96	124
100	43	58	73	138	92	121
10	1	1.18	1.66	41.25	–	–
20	1	1.14	1.62	13.54	15.524	14.37
40	1	1.12	1.54	7.8	11.22	12.02
60	1	1.12	1.52	7.75	4.91	5.55
80	1	1.12	1.46	7.57	2.95	3.58
100	1	1.12	1.36	7.58	2.96	2.81

DPA. Our approach still performs well but the results altogether are also a bit closer now. The RBF kernel based attack performs slightly better and can finally reduce the key space to eight.

Eventually, our findings indicate that SVM based template attacks do not perform best with the RBF kernel. Admittedly, our results concerning the RBF kernel were obtained using our multi-class strategy instead of the usual one-versus-one strategy but we suggest this has no crucial negative impact. Actually, the good performance of a linear kernel should not be surprising since template attacks usually imply a linear classification problem whereas the RBF kernel is appropriate for non-linear problems. The linear kernel also performs well with prior feature selection, *i.e.* known-key DPA and PCA but the normal-based feature selection is very simple, and furthermore it provides better results with a small profiling base. The computational effort of our SVM based attack is in the range of seconds and hence negligible when compared to common TAs.

6 Conclusion

In this work we showed how to improve the performance of template attacks based on support vector machines. Although previous works already demonstrated the advantages of such template attacks, their approaches were not optimal in the sense of efficiency and performance. First of all we proposed a multi-class method tailored for TAs which lead to training less SVMs under a attack model with a strict order, *e.g.* the Hamming weight model. Next, we showed that the subsequent feature selection after the training called normal-based feature selection together with the linear kernel leads to superior results than using it with a prior feature selection, namely known-key DPA or PCA, respectively.

Acknowledgements. The work described in this paper was supported by the German Federal Ministry of Education and Research (BMBF) through the EXSET project (promotional reference 01IS10026B). The information in this document reflects the views only of the authors.

References

1. Archambeau, C., Peeters, E., Standaert, F.-X., Quisquater, J.-J.: Template Attacks in Principal Subspaces. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 1–14. Springer, Heidelberg (2006)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)
3. Brank, J., Grobelnik, M., Milic-Frayling, N., Mladenic, D.: Feature Selection Using Linear Support Vector Machines. No. MSR-TR-2002-63, Microsoft Research (2002)
4. Chang, C.C., Lin, C.J.: LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
5. Chari, S., Rao, J., Rohatgi, P.: Template Attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
6. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge (2000)
7. Duan, K.-B., Keerthi, S.S.: Which Is the Best Multiclass SVM Method? An Empirical Study. In: Oza, N.C., Polikar, R., Kittler, J., Roli, F. (eds.) MCS 2005. LNCS, vol. 3541, pp. 278–285. Springer, Heidelberg (2005)
8. Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. Stochastic Methods. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 15–29. Springer, Heidelberg (2006)
9. Heuser, A., Zohner, M.: Intelligent Machine Homicide. In: Schindler, W., Huss, S.A. (eds.) COSADE 2012. LNCS, vol. 7275, pp. 249–264. Springer, Heidelberg (2012)
10. Hospodar, G., De Mulder, E., Gierlichs, B., Vandewalle, J., Verbauwhede, I.: Least Squares Support Vector Machines for Side-Channel Analysis. In: COSADE 2011. CASED, Darmstadt (2011)
11. Hospodar, G., Gierlichs, B., De Mulder, E., Verbauwhede, I., Vandewalle, J.: Machine Learning in Side-channel Analysis: A First Study. Journal of Cryptographic Engineering 1, 293–302 (2011)
12. Lerman, L., Bontempi, G., Markowitch, O.: Side Channel Attack: An Approach Based on Machine Learning. In: COSADE 2011. CASED, Darmstadt (2011)
13. Lin, H.T., Lin, C.J., Weng, R.C.: A Note on Platt's Probabilistic Outputs for Support Vector Machines, vol. 68, pp. 267–276. Kluwer Academic Publishers, Hingham (2007)
14. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)

15. Microchip Technology Inc.: PIC18F2420/2520/4420/4520 Data Sheet (2008)
16. Platt, J.C.: Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In: Advances in Kernel Methods, pp. 185–208. MIT Press, Cambridge (1999)
17. Platt, J.C.: Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In: Advances in Large Margin Classifiers, pp. 61–74. MIT Press, Cambridge (1999)
18. Rechberger, C., Oswald, E.: Practical Template Attacks. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 440–456. Springer, Heidelberg (2005)
19. Standaert, F.-X., Malkin, T.G., Yung, M.: A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 443–461. Springer, Heidelberg (2009)
20. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
21. Zhang, Y., Schneider, J.G.: Projection Penalties: Dimension Reduction without Loss. In: Fürnkranz, J., Joachims, T. (eds.) ICML 2010, pp. 1223–1230. Omnipress (2010)

A Algorithms for SVM Based Template Attacks

Required adaptations for an attack model without a strict order infer from Section 4.2.

Algorithm 1. SVM Template Building

```
Input: Training set $\widetilde{\mathbf{X}}$ with $N$ traces related to labels $\Omega_1, \ldots, \Omega_M$, and constraint $\gamma$
Output: $M-1$ templates ($\mathbf{a}_i$, $b_i$, $\mathbf{X}_{\mathcal{S}i} \subset \widetilde{\mathbf{X}}$, $A_i$, $B_i$)
1: for $i = 1$ to $M-1$ do
2:    for $j = 1$ to $N$ do
3:       if $\tilde{\mathbf{x}}_j$ belongs to $\Omega_l$ with $l \leq i$ then $c_j \leftarrow 1$ else $c_j \leftarrow -1$
4:    end for
5:    $\mathbf{a}_i$, $b_i$, $\mathbf{X}_{\mathcal{S}i} \subset \widetilde{\mathbf{X}} \leftarrow$ SMO-training [6] with $\widetilde{\mathbf{X}}$, $(c_1, \ldots, c_N)$, and $\gamma$
6:    $A_i$, $B_i \leftarrow$ Sigmoid-training [13] with $\mathbf{a}_i$, $b_i$, $\mathbf{X}_{\mathcal{S}i} \subset \widetilde{\mathbf{X}}$, and $\widetilde{\mathbf{X}}$
7: end for
```

Algorithm 2. SVM Template Classification

```
Input: $M-1$ templates ($\mathbf{a}_i$, $b_i$, $\mathbf{X}_{\mathcal{S}i} \subset \widetilde{\mathbf{X}}$, $A_i$, $B_i$), and new trace $\mathbf{x}$
Output: Posterior probabilities $p(\Omega_1|\mathbf{x}), \ldots, p(\Omega_M|\mathbf{x})$
1: for $i = 2$ to $M-1$ do
2:    $p(\Omega_i|\mathbf{x})$ acc. to (12) with ($\mathbf{a}_j$, $b_j$, $\mathbf{X}_{\mathcal{S}j} \subset \widetilde{\mathbf{X}}$, $A_j$, $B_j$) for $j \in \{i-1, i\}$
3: end for
4: $p(\Omega_1|\mathbf{x})$ acc. to (13) with ($\mathbf{a}_1$, $b_1$, $\mathbf{X}_{\mathcal{S}1} \subset \widetilde{\mathbf{X}}$, $A_1$, $B_1$)
5: $p(\Omega_M|\mathbf{x})$ acc. to (14) with ($\mathbf{a}_{M-1}$, $b_{M-1}$, $\mathbf{X}_{\mathcal{S}M-1} \subset \widetilde{\mathbf{X}}$, $A_{M-1}$, $B_{M-1}$)
```