

A machine learning approach against a masked AES

Reaching the limit of side-channel attacks with a learning model

Liran Lerman · Gianluca Bontempi · Olivier Markowitch

Received: 13 February 2014 / Accepted: 12 October 2014 / Published online: 4 November 2014
© Springer-Verlag Berlin Heidelberg 2014

Abstract Side-channel attacks challenge the security of cryptographic devices. A widespread countermeasure against these attacks is the masking approach. Masking combines sensitive variables with secret random values to reduce its leakage. In 2012, Nassar et al. (DATE, pp 1173–1178. IEEE, 2012) presented a new lightweight (low-cost) boolean masking countermeasure to protect the implementation of the Advanced Encryption Standard (AES) block-cipher. This masking scheme represents the target algorithm of the DPA-Contest V4 (<http://www.dpacontest.org/home/>, 2013). In this paper, we present the first machine learning attack against a specific masking countermeasure (more precisely the low-entropy boolean masking countermeasure of Nassar et al.), using the dataset of the DPAContest V4. We succeeded to extract each targeted byte of the key of the masked AES with 7.8 traces during the attacking phase with a strategy based solely on machine learning models. Finally, we compared our proposal with (1) a stochastic attack, (2) a strategy based on template attack and (3) a multivariate regression attack. We show that an attack based on a machine learning model reduces significantly the number of traces required during the attacking step compared to these profiling attacks when analyzing the same leakage information.

Keywords Side-channel attack · Masking · Profiled attack · Machine learning · Stochastic attack · Template attack

L. Lerman (✉) · O. Markowitch
Quality and Security of Information Systems, Département
d’informatique, Université Libre de Bruxelles, Brussels, Belgium
e-mail: llerman@ulb.ac.be

L. Lerman · G. Bontempi
Machine Learning Group, Département d’informatique,
Université Libre de Bruxelles, Brussels, Belgium

1 Introduction

Embedded devices such as smart cards, mobile phones, and RFID tags are widely used in our everyday lives. These devices implement cryptographic operations allowing to secure, for example, bank transfers, buildings and cars. Several cryptographic primitives exist such as hash functions and encryption functions. During the execution of an encryption algorithm, the device processes secret information. Such secret information could be retrieved with physical attacks against the physical device by analyzing unintentional leakages that appear in power consumption [26], processing time [25], electromagnetic emanation [14] or in a combination of them [47].

Two main families of attacks against cryptographic devices exist: non-profiled and profiled attacks. Non-profiled attacks are one-phase strategy that perform the key recovery step directly on the target device. Profiled attacks are two-phase strategies that (1) characterize the leakage value of the target device on the basis of a key-related information using a clone device (similar to the target device) and then (2) attack the target device.

In 1999, Kocher et al. [26] proposed a non-profiled differential power analysis (DPA) on power consumption data (called traces). This method models the theoretic leakage for each secret information. Then the real and the predicted leakage are compared using metrics, also known as distinguishers, like the correlation coefficient (CPA) [6], the difference of means (DoM) [26], the mutual information (MIA) [15], or the Kolmogorov–Smirnov test (KS) [52]. The rationale is that the likelihood of a secret information is related to the degree of similarity between the predicted and the real leakage information.

Profiled attacks, like template attack (TA) [5] and stochastic attack (SA) [46], make a step forward in the use of sta-

tistical modeling of leakages; they estimate the conditional density function of the trace for each key-related information by creating a Gaussian parametric model during a profiling step. Thereafter, during the attacking phase, the traces are classified by a maximum likelihood approach. We refer to [51] for a recent comparison between template attack and stochastic attack. If the assumption of Gaussianity holds and sufficient data exist for accurate estimation of the parameters, Template Attack can be considered as the strongest leakage analysis in an information theoretic sense [5]. Profiled attack is particularly suitable (1) to analyze the security of a cryptographic device in the worst case scenario and (2) when the adversary is only able to observe a single use of the key (e.g. in stream ciphers) or a secret one time value (e.g. a mask in a masking scheme).

In recent years, the cryptographic community explored new approaches based on machine learning. The results show that template attacks overestimate the security of embedded devices in several scenarios. Lerman et al. [27,28] compared a template attack with a binary machine learning approach, based on non-parametric methods, against cryptographic hardware devices implementing a symmetric and an asymmetric cryptographic algorithm. Hospodar et al. [22,23] analyzed a software implementation of a portion of a block-cipher. Their experiments support the idea that non-parametric techniques can be competitive and sometimes better (i.e. less traces in the attacking phase) than template attack. Heuser et al. [21] generalized this idea by analyzing multi-class classification models in several contexts. In the same year, Bartkewitz [2] applied a multi-class machine learning model allowing to improve the attack success with respect to the binary approach. Recently, Lerman et al. [29] proposed a machine learning approach that takes into account the temporal dependencies between power values. This approach improves the success rate of an attack in a low signal-to-noise ratio with respect to classification methods. At the same time, Martinasek et al. [35] applied a neural network to extract one byte of the key of AES. Their method retrieves the secret value with probability around 0.9 using a single measured power leakage.

Together with attacks, the embedded systems industry needs countermeasures. Side-channel attacks may be counteracted by inducing a leakage independent of the secret target value. It is worth to mention that all the previously presented attacks based on machine learning were applied on unprotected cryptographic devices. Recently, Lerman et al. [30] investigated whether the results of the previous works would be still the same in a protected environment. During the attacking phase, for a specific countermeasure and for a specific device, their investigations concerned: (1) How many traces are required against a protected device with a machine learning model compared to a strategy based on template attack or on stochastic attack (2) How many traces

are required by a machine learning model attacking a protected device compared to an unprotected device (3) What is the impact of the number of traces used in the profiling step by a machine learning model when attacking a protected device.

The results presented in this paper extend the previous analyses in two main directions. We first focus on how to improve the efficiency of the attack by investigating several profiled and non-profiled attacks in protected contexts compared to an unprotected context. On one hand, we compare the previous results with multivariate regression attack and nonlinear stochastic attacks to bypass the protection. On the other hand, we add results on multivariate non-profiled attacks and machine learning models during the key recovery step. This provides a clear idea on how to improve the results of Lerman et al. [30] by proposing an original efficient strategy based solely on machine learning models. The main purpose of our paper is to see the limit of a machine learning approach (i.e. based solely on machine learning models) against a protected device and, thereafter, to compare this strategy to a broader set of attacks. In the second part, we quantify the information that the profiled models retrieve with a large learning set (i.e. in asymptotic contexts). Such a study is of particular importance when the adversary has no constraint on the number of measured traces during the profiling step.

As in Lerman et al. [30], our requirements are fast-execution, low-memory usage and high success rate of the attack. The purpose is to put ourselves in a realistic attack scenario before deployment or certification process (that are expensive and time consuming).

We made a detailed assessment of the proposed strategy by considering several public datasets with different number of traces during the profiling phase and the attacking phase. These traces were collected on a smart card that implements the block-cipher AES protected by a lightweight masking scheme. All our datasets were extracted from the public dataset of the DPAContest V4 [11], “an initiative towards an international benchmarking reference” [11].

The rest of the paper is organized as follows. Section 2 discusses side-channel attacks, non-profiled attacks, profiled attacks, masking countermeasures and strategies against countermeasures. Section 3 introduces our original attack based on a machine learning approach against a masking scheme. Section 4 illustrates the power of our proposal with a large number of experiments. Section 5 concludes this paper with several perspectives of future work.

2 Preliminaries

In this section we introduce the basic definitions of side-channel attacks, non-profiled attacks, profiled attacks, masking countermeasures and strategies against countermeasures.

2.1 Side-channel attacks

During the execution of an encryption algorithm, the cryptographic device processes a function f (targeted by the adversary)

$$f: \mathcal{P} \times \mathcal{O} \rightarrow \mathcal{F}$$

$$s = f_O(p) \tag{1}$$

called a sensitive variable [44] where

- $O \in \mathcal{O}$ (and $\mathcal{O} = \{O_0, O_1, \dots, O_{K-1}\}$ is a key-related information where $\mathcal{O} = \{0, 1\}^{l_1}$, l_1 is the size of the secret value used in f (e.g. one byte of the secret key) and K is the cardinality of \mathcal{O})
- $p \in \mathcal{P}$ (and $\mathcal{P} = \{p_0, p_1, \dots, p_{P-1}\}$ represents a public information where $\mathcal{P} = \{0, 1\}^{l_2}$, l_2 is the size of the public value used in f (e.g. one byte of the plaintext) and P is the cardinality of \mathcal{P})
- $\mathcal{F} = \{0, 1\}^{l_3}$ is the codomain of f where l_3 is the size of the output of f .

Note that l_1, l_2 and l_3 depend on the cryptographic algorithm and the device architecture. We assume that the adversary wants to retrieve the secret value used when the cryptographic device (executing a known encryption algorithm) encrypts known plaintexts.

Prouff [43] showed that nonlinear functions are less robust against side-channel attacks than linear functions. As a result, usually, the target function f represents a nonlinear function such as a substitution box (SBox) of the block-cipher, e.g.

$$f_O(p) = \text{SBox}(p \oplus O) \tag{2}$$

where \oplus is the bitwise exclusive-or.

Let

$${}^jT_i = \left\{ {}^jT_i \in \mathbb{R} \mid t \in [1; n] \right\} \tag{3}$$

be the j -th leakage information (called trace) associated to the i -th target value. We consider the leakage information jT_i of the device at time t depending on the output of $f_{O_i}(p)$ such that

$${}^jT_i = y_i + \epsilon \tag{4}$$

$$= L(f_O(p)) + \epsilon \tag{5}$$

where $y_i = L(f_O(p))$, $\epsilon \in \mathbb{R}$ is the noise following a Gaussian distribution with zero mean and L is the leakage model

$$L: \mathcal{F} \rightarrow \mathcal{Y} \tag{6}$$

$$y = L(f_O(p))$$

where $\mathcal{Y} = \{y_1, y_2, \dots\} \subset \mathbb{R}$ (also known as the set of classes). Examples of models L are the identity, the Hamming weight (HW) and the Hamming distance [33].

2.1.1 Non-profiled attacks

Non-profiled attacks are commonly used to target a cryptographic device. These attacks estimate the output value of $f_O(p)$ for each possible target value O . Then, the estimated leakage model \hat{L} transforms this output value to allow, *in fine*, to compare the real and the predicted leakage information with a distinguisher D (e.g. the Pearson correlation). Mathematically, a univariate non-profiled attack returns the target value \hat{O} that maximizes

$$\hat{O} = \arg \max_{O \in \mathcal{O}} |D(\hat{T}(O), T)| \tag{7}$$

where

- $|x|$ designates the absolute value of x
- $T = [{}^1T, \dots, {}^N T]$ represents a list of traces measured at time t (i.e. ${}^i T \in \mathbb{R} \forall i \in \{1; N\}$)
- $\hat{T}(O) = [\hat{L}(f_O(p_{[0]})), \dots, \hat{L}(f_O(p_{[N]}))]$ refers to a list of estimated leakages parameterized with an estimated key O and known plaintexts $p_{[i]}$ associated to ${}^i T$ (i.e. $\hat{L}(f_O(p_{[0]})) \in \mathcal{Y}$).

This paper focuses on correlation power analysis where the distinguisher represents the Pearson correlation estimator.

The multivariate non-profiled attack generalizes the univariate attack by considering several time instants related to the target information. According to [19], there are two multivariate approaches: (a) apply an attack to a combination of power leakages, or (b) apply an attack to multiple sample points independently and then combine their results. In our experiments, we considered both approaches.

2.1.2 Profiled attacks

Let $\Pr[A]$ be the probability of A and let $\Pr[A | B]$ be the probability of A given B. The profiled attack strategy represents a more efficient attack by deeper leakage estimations. It estimates (with a set of traces called learning set) a template $\Pr[{}^jT_i | y_i; \theta_i]$ (where θ_i is the parameter of the probability density function) for each target value during the profiling step (also known as learning step). The learning set is measured on a controlled device similar to the target chip. In our experiments, we used the same cryptographic device but we refer to [37] that studies practical issues when the controlled and the target devices differ.

Once a template is estimated for each target value, during the attacking step the adversary classifies a new trace T (measured on the target device) using the a posteriori probability returned by a model $A(T)$

$$\hat{y} = A(T) = \arg \max_{y_i \in \mathcal{Y}} \Pr [y_i | T] \tag{8}$$

$$= \arg \max_{y_i \in \mathcal{Y}} \frac{\Pr [T | y_i] \times \Pr [y_i]}{\Pr [T]} \tag{9}$$

$$= \arg \max_{y_i \in \mathcal{Y}} \hat{\Pr} [T | y_i; \hat{\theta}_i] \times \hat{\Pr} [y_i] \tag{10}$$

where the a priori probabilities $\hat{\Pr} [y_i]$ are estimated by the user accordingly.

If a set T of traces (where $T = [^1T, \dots, ^N T]$ and $^i T \in \mathbb{R}^n \ \forall i \in \{1; N\}$) for a constant secret key are available, the adversary uses the equation (or the log-likelihood rule)

$$\hat{O} = \arg \max_{y_i \in \mathcal{Y}} \prod_{j=1}^N \Pr [^j T | y_i] \times \Pr [y_i] \tag{11}$$

Several approaches exist to estimate the probability $\Pr [T_i | y_i]$ such as the parametric template attack [5], the stochastic attack [46], the multivariate regression model [49] and the non-parametric machine learning models [22,27].

Template attacks Template attacks [5] assume that $\Pr [T_i | y_i]$ follows a Gaussian distribution for each target value, i.e.

$$\hat{\Pr} [T_i | y_i; \hat{\theta}_i] = \hat{\Pr} [T_i | y_i; \hat{\mu}_i, \hat{\Sigma}_i] \tag{12}$$

$$= \frac{e^{-\frac{1}{2}(T_i - \hat{\mu}_i) \hat{\Sigma}_i^{-1} (T_i - \hat{\mu}_i)^\top}}{\sqrt{(2\pi)^n \det(\hat{\Sigma}_i)}} \tag{13}$$

where $\det(\Sigma)$ denotes the determinant of the matrix Σ while $\hat{\mu}_i \in \mathbb{R}^n$ and $\hat{\Sigma}_i \in \mathbb{R}^{n \times n}$ are respectively the expected value and the covariance of the n variate traces associated to the i -th target value.

Stochastic attacks Stochastic attacks [46] (also known as linear regression attack) model the leakage information that depends on the secret value $y_i = f_O(p)$ at time t with a regression model ${}_t h$, i.e.

$${}^j_t T_i = L(y_i) + \epsilon \tag{14}$$

$$= {}_t h(y_i) + {}_t R \tag{15}$$

$$= {}_t c + \sum_{u=1}^U {}_t \alpha_u g_u(y_i) + {}_t R \tag{16}$$

where ${}_t R \in \mathbb{R}$ is a residual Gaussian noise at time t , $\{{}_t c, {}_t \alpha_1, \dots, {}_t \alpha_U\} \in \mathbb{R}^{U+1}$ is the parameter of the regression model ${}_t h$ and $\{g_1, \dots, g_U\}$ is the basis used in the regression. Each g_u is a monomial of the form $\prod_{j \in \mathcal{J}} \text{Bit}_j(y_i)$ where $\text{Bit}_j(y_i)$ returns the j -th bit of y_i and $\mathcal{J} \subset \{1, 2, \dots, l_3\}$. In other words, we can vary the degree of a stochastic model from 1 through to l_3 to vary its complexity. For example, in a linear basis, each function g_u equals to

$$g_u(y_i) = \text{Bit}_u(y_i) \tag{17}$$

Then, the attacker assumes that $\Pr [T | y_i]$ follows the Gaussian distribution $\mathcal{N}(h(y_i), \Sigma)$ where $h(y_i)$ equals to $\{{}_1 h(y_i), {}_2 h(y_i), \dots, {}_n h(y_i)\}$ and $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix of the residual term. An extended version of stochastic attack removes the profiling step [10]. However, this approach is out of the scope of this work.

Multivariate regression attack The multivariate regression model [49] describes the relationship between a set of n features representing a trace $T_i = \{{}_1 T_i, \dots, {}_n T_i\} \in \mathbb{R}^n$ and the target value such that

$$y_i = c + \sum_{j=1}^n \alpha_j {}_j T_i + \epsilon \tag{18}$$

where $\alpha_j \in \mathbb{R} \ \forall j \in \{1; n\}$.

Non-parametric machine learning attack Non-parametric (supervised) machine learning models make no assumptions about the density distribution functions.

The machine learning field regroups several learning algorithms. For example, random forest model (RF) [3] builds a set of decision trees that classifies a trace based on a voting system. Support vector machine (SVM) [7] discriminates traces associated to different target values with hyperplanes in optionally a higher dimensional space than the original dimension. Unfortunately, no statistical modeling algorithms can be entitled to be the universally best one as formalized by the no-free-lunch theorem.

We refer to [2,21–23,27–29] for a detailed introduction to the non-parametric machine learning models.

2.2 Masking countermeasure

Based on secret sharing, the masking countermeasure aims to reduce the unintentional leakage information of a cryptographic device [4]. For this, the method masks a public information p with d uniformly distributed random values $v = \{v_0, v_1, \dots, v_{d-1}\} \in \mathcal{V}^d$ changing at each execution where $\mathcal{V} = \{0, 1\}^{l_4}$ and l_4 is the size of each random value. This approach is called a masking scheme of order d . From a theoretical point of view, the security level of a masked implementation against side-channel attacks increases exponentially with d [4] when the amount of noise in the traces is sufficiently high [48].

To deal with the mask values, the encryption scheme (E) is modified and satisfies the relation

$$E'(\text{plaintext}, \text{key}, \text{masks}) = E(\text{plaintext}, \text{key}) \tag{19}$$

where E' is the modified encryption algorithm. In other words, E and E' output the same ciphertext when the same pair $\{\text{key}; \text{plaintext}\}$ is used. However E and E' manipulate

different internal values. If correctly implemented, the leakage of the masked implementation becomes statistically independent of the key.

The public cryptographic literature provides plenty of masking schemes based on the one time pad. The Boolean masking computes the combination of p and v with the bit-wise exclusive-or operation [4]. The multiplicative masking performs the combination with a multiplication in the field $GF(2^n)$ [1, 18]. The affine masking combines the advantages of the previous schemes in a security point of view with the cost of a heavy operation during the encryption step [50]. Our paper deals with the most common method—the Boolean masking scheme—as the DPAContest V4 uses it.

The main issue in designing masking scheme lies in propagating the mask values throughout nonlinear functions (i.e. $SBox(p \oplus v) \neq SBox(p) \oplus SBox(v)$). Several approaches exist to modify the SBoxes part. One of the easiest approach computes a table look-up which associates to each masked input $p \oplus v_{input}$ the output value $SBox(p) \oplus v_{output}$ where v_{input} and v_{output} are mask values [36]. However, such fast approach may require a lot of memory to store the tables. This is the proposed target masking scheme of the DPAContest V4.

2.3 Strategies against countermeasures

Recently, Moradi et al. [38] show that an adversary can successfully target a masked scheme when the implementation of the countermeasure contains mistakes. More precisely, a non-profiled CPA attack with an appropriate model retrieves the secret key with less than 200 traces.

When the masked implementation is correctly implemented, potentially, an adversary can retrieve the secret information using an attack of order $d + 1$ (where the attacker considers $d + 1$ targets: the set of d random mask values and a key-related information). More precisely, the $(d + 1)$ -order non-profiled attack combines $d + 1$ points in each trace associated to the mask values (e.g. in a masking scheme of order 1, the adversary combines instants associated to the target value $HW(f_O(p) \oplus v_0)$ and to $HW(v_0)$ where v_0 represents the mask value). Then, after this combination, a classical non-profiled attack is performed. It turns out that in a d -order masking scheme and with a correlation power analysis, the combination of $d + 1$ different instants (related to the d mask values and to the target value) correlates to the targeted sensitive variable but the masking scheme can still affect the success of the attack as the combination does not remove completely the dependence with the mask values [40].

In a secure implementation context, it is necessary that the mask values remain secret. Indeed, once the mask value is revealed or removed, the attacker is able to execute an efficient non-profiled or profiled attack.

In 2008, Schindler [45] extended the stochastic attack to a masking context by taking into account the mask value v in the deterministic part when targeting the secret information $y_i = f_O(p \oplus v)$, e.g.

$${}_t^j T_i = L(y_i) + \epsilon \tag{20}$$

$$= {}_t h(y_i) + {}_t R \tag{21}$$

$$= \begin{cases} {}_t c + \sum_u {}_t \alpha_u g_u(f_O(p \oplus v)) + {}_t R & t \in T_1 \\ {}_t c + \sum_u {}_t \alpha_u g_u(v) + {}_t R & t \in T_2 \end{cases} \tag{22}$$

where T_1 and T_2 represent sets of time instants correlated respectively to $f_O(p \oplus v)$ and to v . During the attacking step, the adversary replaces the probability $\hat{\Pr}[T | y_i; \hat{\theta}_i]$ in (10) with

$$\Pr[T | y_i = f_O(p)] = \sum_{v \in \mathcal{V}^d} \Pr[T | y_i = f_O(p \oplus v)] \tag{23}$$

where $\Pr[T | y_i = f_O(p \oplus v)]$ follows the Gaussian distribution $\mathcal{N}(h(y_i), \Sigma)$. The main advantage of this approach is that we need a smaller set of measurements during the profiling step compared to template attack applied to masking [45].

Oswald et al. [40] evaluated several approaches to attack a masked implementation with a combination between template attack and correlation power analysis. In the same year, Gierlichs et al. [16] extended these practical proposals with a theoretical analysis. The first approach (called Templates Before Preprocessing) uses template attack to extract the values of the estimated leakage information of the $d + 1$ masked information (e.g. $HW(f_O(p \oplus v))$ and $HW(v_i)$) before combining them and to apply a correlation power analysis. The second approach (called templates during preprocessing) forces a bias into the mask values by removing traces associated to certain mask values. For this, the template attack extracts mask-related information and keeps a subset of traces associated to a subset of mask values. Then a correlation power analysis on the selected traces reveals the key. The third approach (called templates after preprocessing) uses template attack to extract the unmasked sensitive value (e.g. $HW(f_O(p))$) and performs a correlation power analysis on the extracted unmasked sensitive value. The last approach (called template-based DPA) performs a template attack against the masking implementation by replacing $\Pr[T | y_i]$ in Eq. 10 with

$$\hat{\Pr}[T | y_i] = \sum_{v \in \mathcal{V}^d} \hat{\Pr}[T | y_i \wedge v] \times \hat{\Pr}[v] \tag{24}$$

As a result, we need $\text{card}(\mathcal{Y}) \times \text{card}(\mathcal{V}^d)$ templates (where $\text{card}(x)$ represents the cardinality of the set x), one for each possible combination of y_i and v .

3 Machine learning approach against masking countermeasure

We propose a new approach that uses a machine learning approach to: (1) bypass the problem of combining mask-related information that still keeps a dependence to mask values (unlike the d -order non-profiled attack, the templates before preprocessing and the templates after preprocessing); (2) keep all traces in the attacking step (unlike the templates during preprocessing); (3) reduce the number of templates from $|\mathcal{Q}| \times |\mathcal{V}^d|$ to $|\mathcal{V}^d|$ (compared to the template-based DPA) leading to several advantages.

From a theoretical point of view, the main issues are that: (i) the number of required data increases with the number of templates (cf. we need one learning set per template that leads to a gigantic workload in the profiling step [45]) and (ii) the imbalanced class problem [24] arises in the Template-based DPA according to the density distribution of $L(f_O(p))$ (unlike our proposal).

From a practical perspective, in the case of the DPAContest V4, the adversary has no control on the attacked device. As a result, we (empirically) estimated that template-based DPA needs a large number of measurements in the profiling step—at least 40,000 traces each of 435,002 samples, representing more than 2^{34} bytes of information—to have at least one trace per template with probability 0.99 when the Hamming weight leakage model is chosen. For the same problem, our proposal needs at least 200 traces (i.e. a realistic attack scenario). In practice, we need at least 48,698 traces for template-based DPA and at least 35 for our proposal when considering the dataset of the DPAContest V4.

Our approach applies a profiled attack to extract the mask values (during the *mask recovery step*) before a non-profiled attack that retrieves the secret key (during the *key recovery step*). Note that this approach is generalizable to the case where a profiled attack is used to extract the secret key. Furthermore, we assume to be in the worst case scenario where the adversary knows the mask values used during the profiling phase. Our requirements are fast-execution, low-memory usage and high success rate (i.e. realistic attack scenarios). Efficient methods to perform profiled attacks have been proposed recently [2, 21–23, 27, 28]. These methods use a machine learning model that returns the target value after a learning (profiling) step. Concerning the non-profiled attack, several approaches exist. One of the most efficient methods represents the correlation power analysis that does not require any estimation probability density function. Note that our method can be extended to other (nonlinear) distinguishers.

During the profiling phase, we make three main steps on a controlled device: (1) we collect a set of traces \mathbf{T} with an oscilloscope; (2) we select p points $\{t_1, t_2, \dots, t_p\}$ that are significantly correlated to known mask values; (3) we build a

profiled model $A(T)$ that returns the (combination of) d estimated mask values based on a trace T at time $\{t_1, t_2, \dots, t_p\}$. We also build a learning model against the target function y_i if the adversary uses it during the *mask recovery step*. During the attacking phase, we then use $A(T)$ for each collected traces (on a target device) to take into account the estimated mask values during the *mask recovery step*. Example 1 illustrates our proposal with a specific case.

Example 1 Suppose that we target a masking scheme of order 1 and that we build one support vector machine [3]—a machine learning model— $A(T)$ during the profiling step. Our target value represents the output of a masked nonlinear function $SBox(p \oplus v \oplus O)$. The correlation power analysis step returns the key that maximizes

$$\hat{O} = \arg \max_{O \in \mathcal{O}} |\rho(\hat{\mathbf{T}}(O), \mathbf{T})| \quad (25)$$

where

- ρ represents a Pearson correlation estimator.
- $\mathbf{T} = [{}_t^1 T, \dots, {}_t^N T]$ represents a list of traces measured at time t
- $\hat{\mathbf{T}}(O) = [\text{HW}(f_O(p_{[1]} \oplus A({}_t^1 T))), \dots, \text{HW}(f_O(p_{[N]} \oplus A({}_t^N T)))]$ refers to a list of estimated leakages parameterized with an estimated key O and known plaintexts $p_{[i]}$ associated to \mathbf{T} .

Figure 1 summarizes our approach. First, during a preliminary phase, we implement the cryptographic device with its countermeasure before collecting a set of traces. Secondly, during the profiling phase, we reduce the number of points per trace with a feature selection algorithm [8] before selecting the best profiled model. In the third phase, we use the selected profiled model with a non-profiled attack or a profiled attack to extract the secret key value. This phase allows to estimate the resistance of cryptographic devices during the post-attacking phase (called the *Security Level Estimation* in Fig. 1) based for example on the number of traces required to extract the key.

The success of this approach is strictly related to the quality of our profiled model: the lower the error between the correct and the estimated mask values by the profiled model, the higher is the correlation between the real and the predicted traces for the correct key during the attacking phase. As a result, reducing the error of the profiled model leads to reduce the number of traces required to distinguish the correct key from the others. In the ideal case, our model extracts the mask values with probability one and, as a result, correlation power analysis returns the key with the same number of traces as an unprotected implementation during the attacking phase.

Several previous works showed that a machine learning approach improves the success of attacks with respect to tem-

Fig. 1 Strategy against a masked implementation

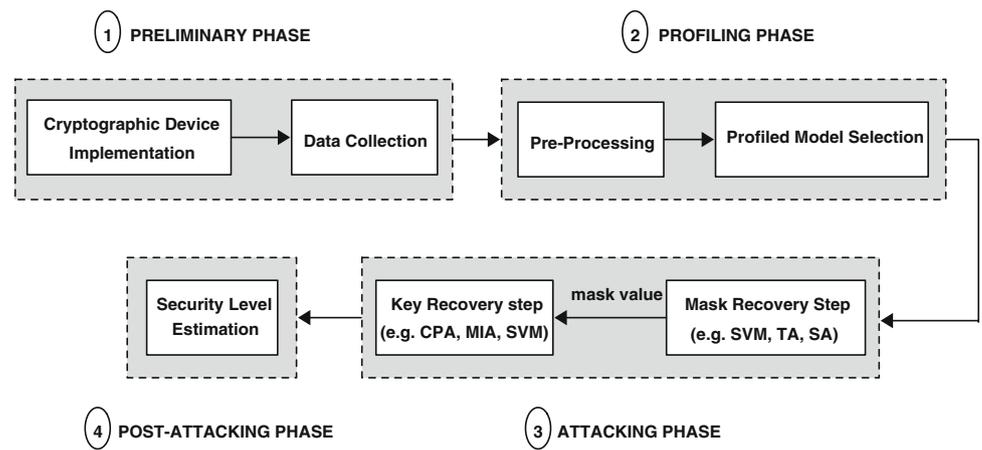


plate attack [2, 21–23, 27, 28]. Therefore, we expected that the machine learning approach induces a reduction of the number of traces during the attacking phase compared to a strategy based on template attack.

In parallel, the quantity of traces used during the profiling step should influence the number of traces required during the attacking phase. Indeed, the variance of models decreases as the training set size increases [20]. Therefore, the higher the number of traces in the learning set, the lower is the error between the correct and the estimated mask values and the lower the number of traces required during the attacking phase. Several experiments have been performed to verify these intuitions.

4 Experiments and discussion

This section regroups all our experiments. In the following, the notation A/B is used in a masking context to denote the configuration with the learner A used in the *Mask Recovery step* and a profiled attack or a non-profiling attack B used in the *Key Recovery step*. In an unprotected context, the notation $/A$ indicates a profiled attack or a non-profiling attack used in the *Key Recovery step*.

4.1 Target implementation

The experiments were carried out on electromagnetic emission leakages that are freely available on the DPAContest V4 website [11] to easily reproduce the results. The target cryptographic device (an Atmel ATmega-163 smart card) implements in software the masked block-cipher AES-256 in encryption mode without any mode of operation. Each trace has 435,002 samples associated to the same secret key and measured during the first round. The masking scheme is a variant of the “Rotating Sbox Masking” [39]; an additive Boolean masked scheme with masked SBox. According to its authors, it has a low-cost design and keeps performances and

complexity close to the unprotected scheme (in a hardware context) while being resistant against several side-channel attacks. The purpose of the DPAContest is to retrieve the first 128 key bits. As we target the first 128 key bits and since the first round of AES-128 and AES-256 is the same, in the following we focus on AES-128.

The “Rotating Sbox Masking”, modified for the DPAContest, uses sixteen public masks $v = \{v_0, v_1, \dots, v_{15}\}$. This strategy is based on a low-entropy masking scheme that aims to limit the amount of entropy. At the beginning of each encryption a random secret value (called offset) is drawn randomly between 0 and 15. Then, the i -th byte of the plaintext is xored with the mask $v_{i+\text{offset}}$, i.e.

$$p_i \oplus v_{i+\text{offset}} = s_i = \text{one byte of the STATE} \tag{26}$$

where $i + \text{offset}$ is computed modulo 16. Sixteen customized SBox (called MaskedSubBytes) are precomputed such that $\text{MaskedSubBytes}_i(s_j) = \text{SBox}(s_j \oplus v_i) \oplus v_{i+1}$, $i \in \{1, 2, \dots, 16\}$. As a result, the hardware and software designers precompute once the sixteen SBoxes and store them in memory.

After the linear part of AES, an additional operation is processed to ensure that the State input for the next round equals

$$\left(\text{MC} \circ \text{SR} \circ \text{SBox} \circ \begin{Bmatrix} s_0 \oplus v_{0+\text{offset}+r} \oplus O_0^r \dots \\ s_1 \oplus v_{1+\text{offset}+r} \oplus O_1^r \dots \\ s_2 \oplus v_{2+\text{offset}+r} \oplus O_2^r \dots \\ s_3 \oplus v_{3+\text{offset}+r} \oplus O_3^r \dots \end{Bmatrix} \right) \oplus \begin{Bmatrix} v_{0+\text{offset}+r+1} \dots v_{12+\text{offset}+r+1} \\ v_{1+\text{offset}+r+1} \dots v_{13+\text{offset}+r+1} \\ v_{2+\text{offset}+r+1} \dots v_{14+\text{offset}+r+1} \\ v_{3+\text{offset}+r+1} \dots v_{15+\text{offset}+r+1} \end{Bmatrix} \tag{27}$$

where $r \in \{0, \dots, 8\}$ (for AES-128) is the round number, O_i^r is the $(i + 1)$ -th byte of the r -th subkey while SR and MC represent respectively the ShiftRow and the MixColumns operations of AES. For the last round (where $r = 9$ for

AES-128), the masking scheme insures that its output value equals to

$$\left(\text{SR} \circ \text{SBox} \circ \begin{pmatrix} s_0 \oplus v_{0+\text{offset}+r} \oplus O_0^r \dots \\ s_1 \oplus v_{1+\text{offset}+r} \oplus O_1^r \dots \\ s_2 \oplus v_{2+\text{offset}+r} \oplus O_2^r \dots \\ s_3 \oplus v_{3+\text{offset}+r} \oplus O_3^r \dots \end{pmatrix} \right) \oplus \begin{pmatrix} v_{0+\text{offset}+r+1} \dots v_{12+\text{offset}+r+1} \\ v_{1+\text{offset}+r+1} \dots v_{13+\text{offset}+r+1} \\ v_{2+\text{offset}+r+1} \dots v_{14+\text{offset}+r+1} \\ v_{3+\text{offset}+r+1} \dots v_{15+\text{offset}+r+1} \end{pmatrix} \quad (28)$$

Finally, the last mask values are removed with a xor operation.

The DPAContest V4 uses a SASEBO-W platform that contains an ATmega163 8-bit smart card. The smart card is powered at 2.5V and clocked a 3.57 MHz. A LeCroy wave-runner 6100A oscilloscope was used. The bandwidth is 200 MHz and the sampling rate equals 500 MS/s. We refer to [11, 39] for additional information on this masking scheme and on the acquisitions setup.

4.2 Experimental results

For the sake of fairness, we compared different attacks based on the same target value and the same dataset: each attack extracts first the offset value before applying a key recovery step. Note that an adversary targeting the offset or the mask value leads to the same result in our case: the (Pearson) correlation between them equals one. We suggest to target the mask value when the setting differs.

All our experiments were executed on a MacBook Pro with 2.66 GHz Intel Core 2 Duo, 8 GB 1,067 MHz DDR3 with the tenth major release of OS X (i.e. OS X Mavericks version 10.9). The attack process lasted roughly 3 weeks without considering the step of collecting the traces.

4.2.1 Finding the offset value on traces

Before proceeding with the quantitative analysis, we report here a preliminary visualization phase that allowed us to find the points that are the highest correlated with the secret offset. For the sake of time and memory (and due to the big data context), we computed the efficient Pearson correlation between each instant of 1,500 traces and the offset values (see Fig. 2). However, we suggest to test several other methods in a non-big data scenario (or when the adversary has enough time, memory and money) such as strategies based on mutual information (e.g. minimum redundancy maximum relevance [42]), the sum of squared pairwise T-differences filter [17] and the principal component analysis [41]. It is worth emphasizing that several instants are (significantly) correlated with the target value. Visualization suggests that

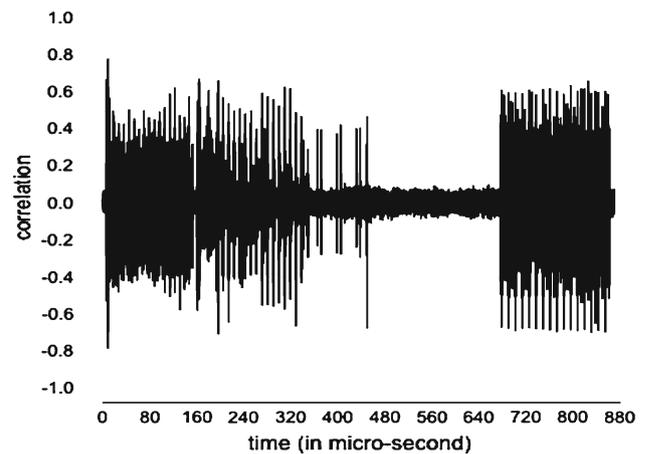


Fig. 2 Correlation between offset and power values at each time in the first round of the masked AES

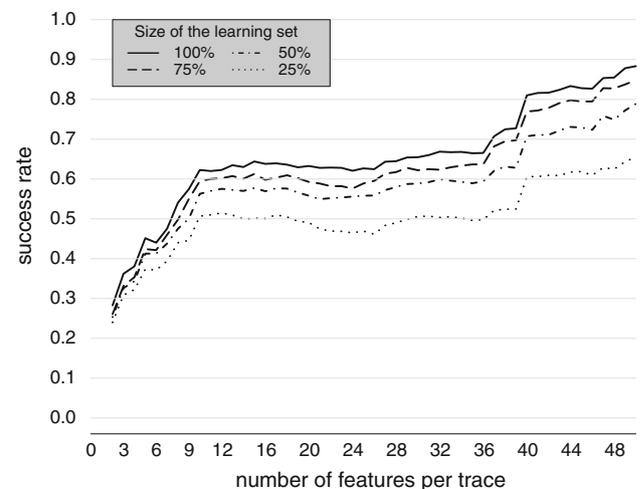


Fig. 3 Support vector machine

apart from the central part of traces there is a high amount of information about the offset value available in each trace. As a consequence, we should expect that the profiled model would output the right offset value with a high probability.

4.2.2 Model selection

This section assesses and compares several classifiers that extract the secret offset value. We considered five different types of multi-class classification models: support vector machine (SVM), random forest (RF), template attack (TA), stochastic attack (SA) and multivariate regression analysis (MRA). We used two disjoint sets: a learning set of 1,500 traces to estimate the parameters of each model and a validation set of 1,500 traces to measure their success rate in predicting the right offset value. During the feature selection step, in each trace, we selected 50 instants that are the

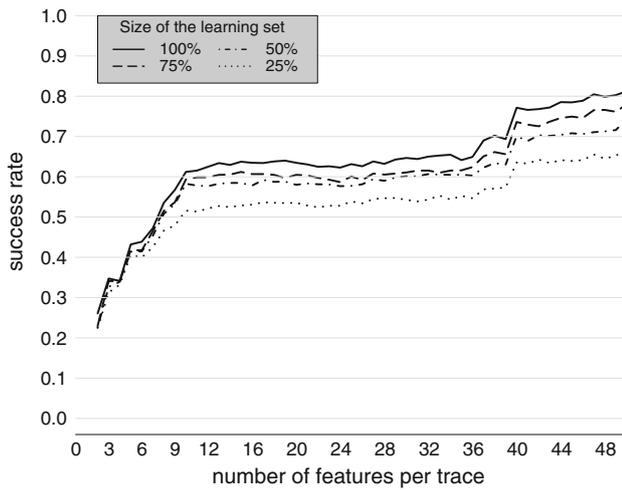


Fig. 4 Random forest

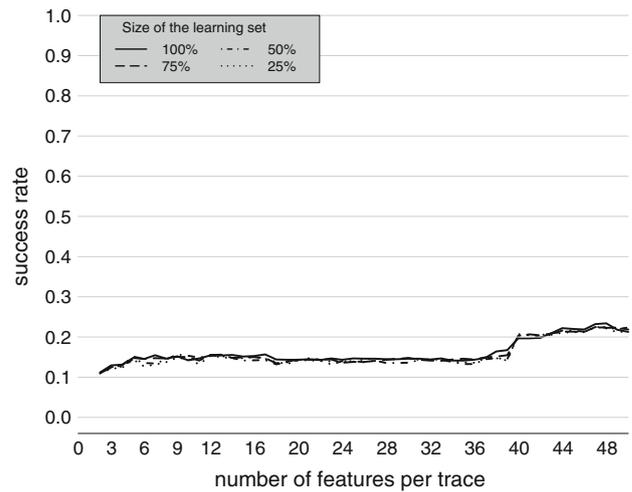


Fig. 7 Multivariate regression analysis

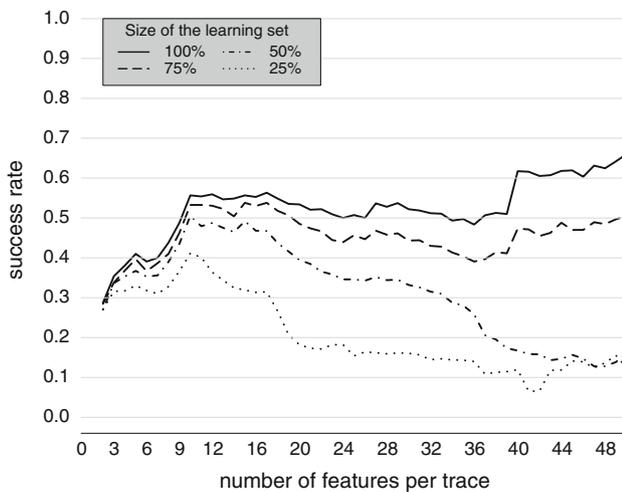


Fig. 5 Template attack

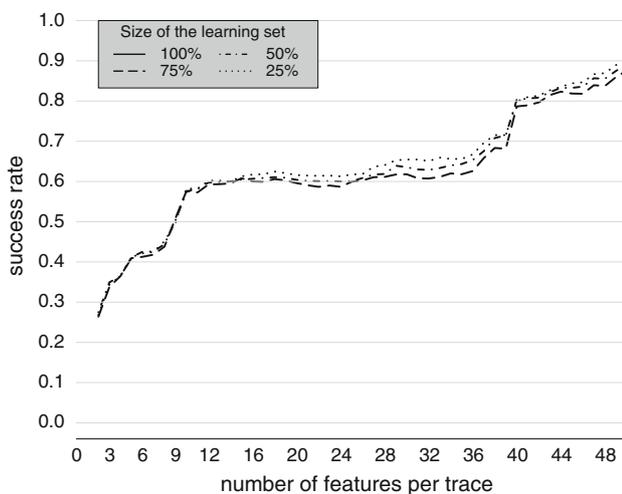


Fig. 6 Stochastic attack with degree 1

highest linearly correlated with the offset value.¹ We did not consider other feature selection methods (such as “Principal Component Analysis” [41] or “minimum Redundancy Maximum Relevance” [42]) due to their massive memory requirements or time consuming while our dataset contained $1,500 \times 435,002\text{bytes} > 2^{29}$ bytes.² We refer to [32] for an analysis of practical issues that a security evaluator faces when performing side-channel attacks. In spite of the low feature selection complexity, we observed a high success rate of the models.

Figures 3, 4, 5, 6 and 7 report the success rate to predict the right offset value as a function of the number of points (that were selected from the sorted 50 instants) used in each trace for respectively support vector machine, random forest, template attack, stochastic attack (with degree 1) and multivariate regression analysis. We can deduce the following observations. First, as expected, the higher the number of traces in the learning set (from 25 to 100 % of 1,500 traces), the higher is the accuracy, with an exception for multivariate regression analysis. This can be explained by the fact that the low complexity of the multivariate regression analysis (low variance) requires a small learning set to reach its best result. Secondly, the number of selected points in each trace influences the success rate: the higher the number of features, the higher is the success rates for support vector machine, random forest, stochastic attack and multivariate regression analysis. It is interesting to remark that in a small learning set setting (i.e. less than 75 % of the entire learning set) the template attack reduces its success rate when the number of features goes beyond a certain size. This is presumably

¹ The 50 instants are sorted in descending order with respect to their correlation coefficient in absolute value.

² Each sample of the trace is an 8-bit value. The limit of R—the used program language—is 2^{31} bytes for a matrix.

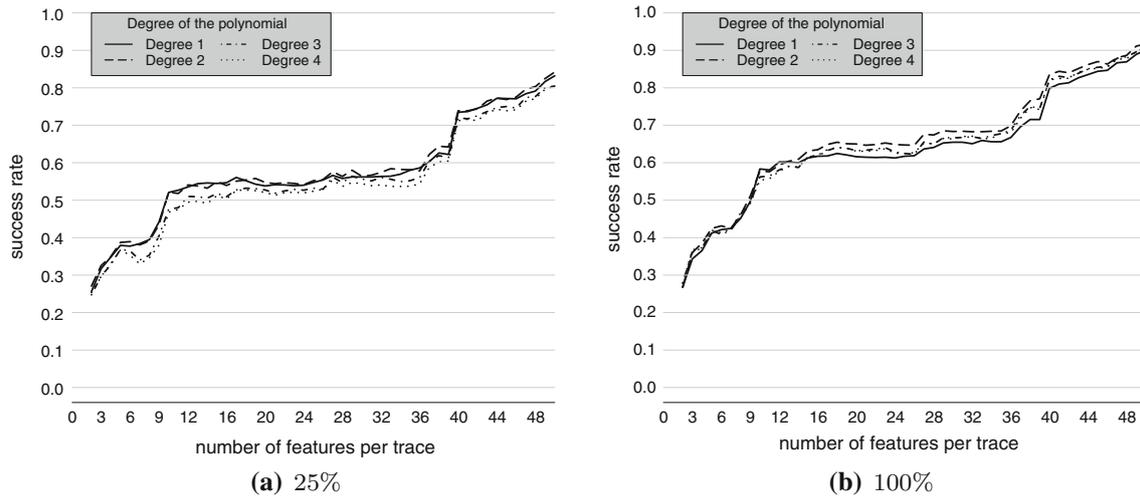


Fig. 8 Stochastic attack with different degrees (from 1 to 4) using 25 % (a) and 100 % (b) of 1,500 traces in the learning set

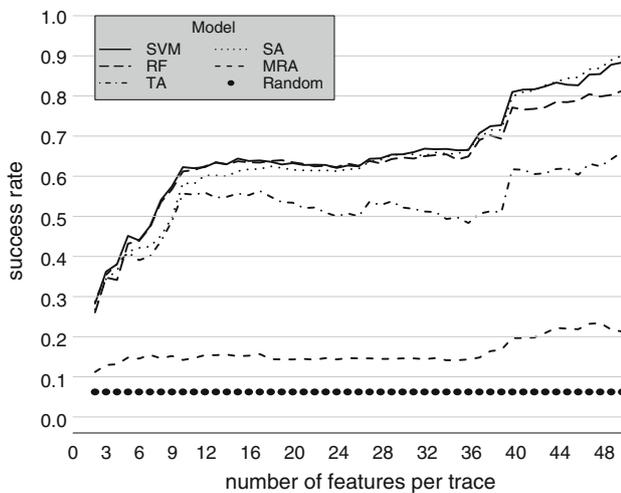


Fig. 9 SVM vs RF vs TA vs SA vs MRA vs random

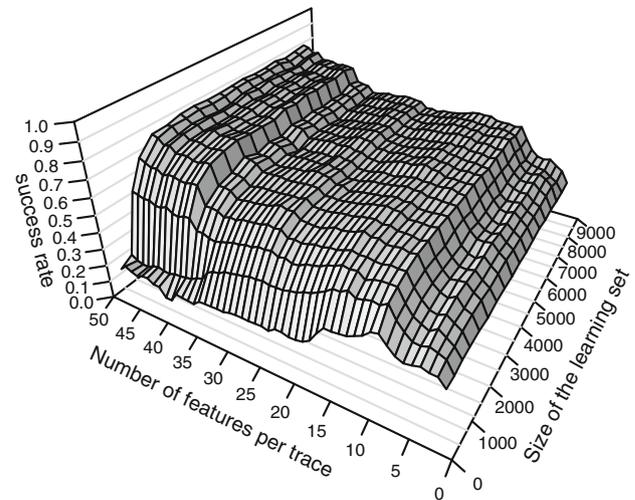


Fig. 10 Template attack

due to the ill-conditioning of the covariance matrix when the number of features is too large.

Figure 8 shows what happens when we build a stochastic attack with different degrees (from 1 through to 4). In our setting, the linear stochastic attack reaches similar results as the nonlinear stochastic attack. This observation is not surprising since, for computational reasons, we applied a feature selection that searches for linear dependencies. As a result, we consider the linear model in the following. However, the results may change if the size of the profiling set increases as the nonlinear stochastic attack (that has more parameters to estimate than the linear approach) requires more traces to estimate its parameters.

Figure 9 combines the success rate of a random model (i.e. $\frac{1}{16}$) and the five previous models (i.e. support vector machine, random forest, template attack, linear stochastic attack and

multivariate regression analysis) by choosing the best size for the learning set that is 100 % of 1,500 traces. In our setting, multivariate regression analysis has the worst success rate. Therefore, we do not consider this model in the following of our experiments. The success rates of support vector machine, random forest and stochastic attack are similar and greater than the success rate of template attack. We could argue that template attack requires a larger learning set. Figures 10 and 11 highlight the impact on the success rate of respectively template attack and support vector machine by increasing the size of the learning set until 8,500 traces.³ The success rate of template attack remains lower than support vector machine.

³ Note that the first four sizes represent 25, 50, 75 and 100 % of 1,500 traces.

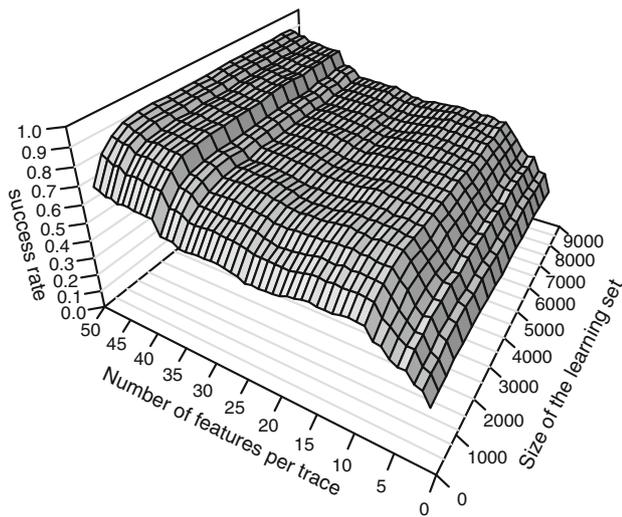


Fig. 11 Support vector machine

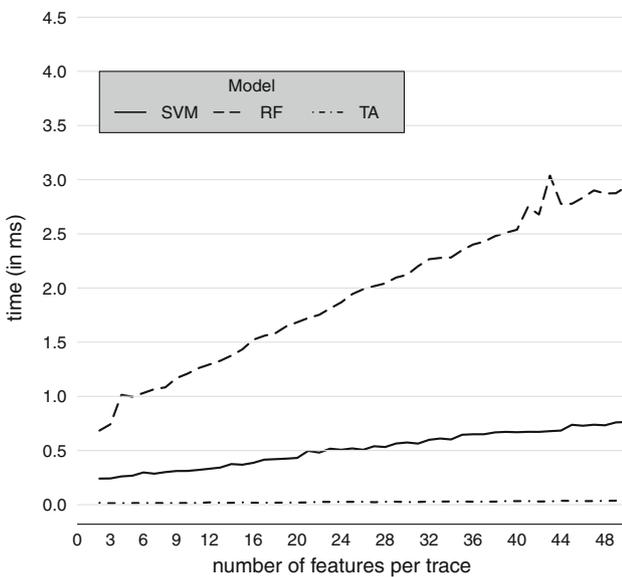


Fig. 12 Time to process a trace in the learning step (in ms) per number of feature selected

Note that we did not select the best meta-parameter values for support vector machine and random forest (such as the number of trees in the random forest) but only the best number of features (from 2 to 50) to predict the target value. The default values of the implementation of support vector machine [9] and random forest [31] were used.⁴ As a consequence, we do not claim that the Support Vector Machine configurations and the random forest configurations are necessarily the best one for profiled attack for our experiments. However, our experiments show that a profiled attack based

⁴ Support vector machine had a radial kernel with a gamma equals to the inverse of the data dimension and a cost of 1. Random forest had 500 trees.

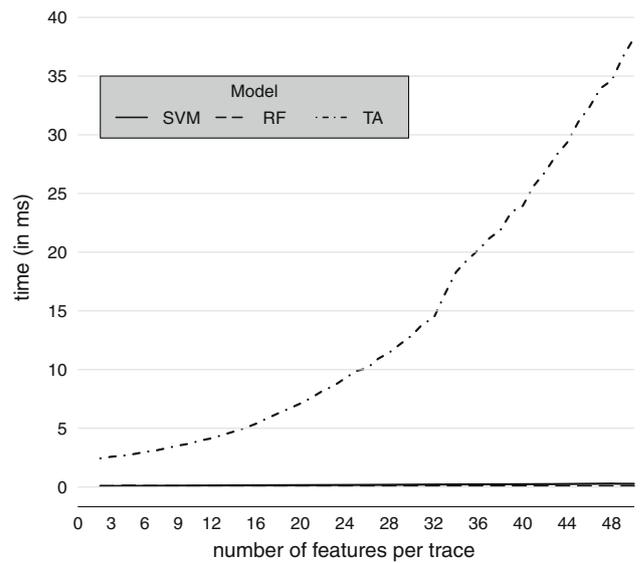


Fig. 13 Time to process a trace in the attacking step (in ms) per number of feature selected

on a machine learning model extracts more information on the offset value than a strategy based on template attack for the presented task.

Based on the above considerations, and to choose the best learning model, we looked at the learning time and the prediction time of the offset, based on one trace, as a function of the number of selected points (see Figs. 12, 13). Template attack has the lowest learning time while its prediction time increases exponentially in the number of selected features. Support vector machine has a lower learning time and a reasonable prediction time compared to random forest. As a result, in the attacking step, we use only a support vector machine as the machine learning model. We do not report the results for stochastic attack or for multivariate regression analysis as we used unoptimized and nonpublic implementations. According to the previous results, we selected 50 features for support vector machine, template attack and stochastic attack (of degree 1) leading to a success rate of respectively 0.88, 0.66 and 0.90.

4.2.3 Key recovery step

During the attacking step we considered four settings targeting the Hamming weight of the MaskedSubBytes. In the first setting, the correlation power analysis extracts the secret key on an unmasked implementation (i.e. the non-profiled attack always receives the correct offset value). The second setting targets the masked implementation where a support vector machine extracts the mask value and where a correlation power analysis searches the secret key. In the third and fourth experiments, we exchanged the support vector machine by respectively the template attack and the stochastic attack. We repeated ten times each setting with a different set of traces

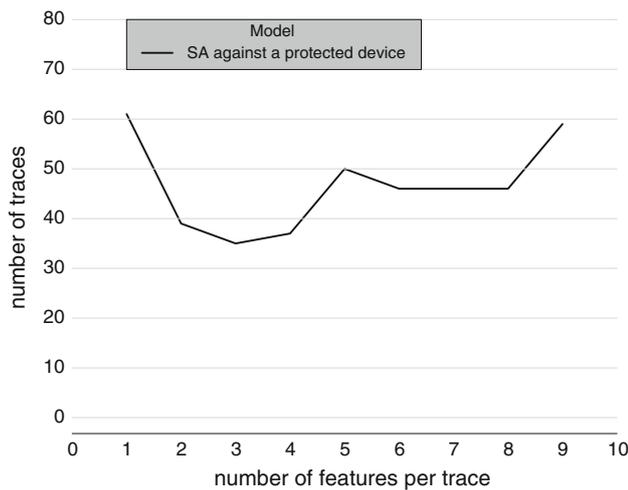


Fig. 14 Number of traces during the attacking phase in function of the number of features

during the attacking phase while the learning set remains the same.

Figure 15 summarizes the number of key bytes found as a function of the number of traces used (in average) during the non-profiling phase for each setting. We found the key with 16.3 traces (with less than 5 s of execution time) for the unmasked implementation. For the masked implementation, we extracted the key with 26 traces (with less than 20 s of execution time) using the support vector machine, with 27.8 traces (with less than 80 s of execution time) using the stochastic attack and with 56.4 traces (with less than 45 s of execution time) using the template attack. Figure 16 shows the minimum, the maximum and the average number of traces used to find the key. Compared to each strategy applied on the protected device, the support vector machine (combined with the correlation power analysis) leads to the closest results to an unprotected configuration.

For the sake of completeness, we also implemented the state-of-the-art stochastic attack on the masking scheme without a non-profiling step as proposed by Schindler [45] (see Eq. 20). Figure 14 shows the number of traces needed on a validation set in function of the number of features used. The stochastic attack needs more than 40 traces to find the key when the model considers more than 5 features and it reaches the minimum with 3 features. According to Fig. 15, stochastic attack needs 107 traces in average to extract the 16 key bytes on a testing set (with less than 180 s of execution time) (Fig. 16).

4.2.4 Attacking step with a multivariate key recovery step

To improve our best attack (i.e. the combination of support vector machine and correlation power analysis), we have studied how to minimize the number of traces used during the key recovery step by mixing n sample points. As discussed

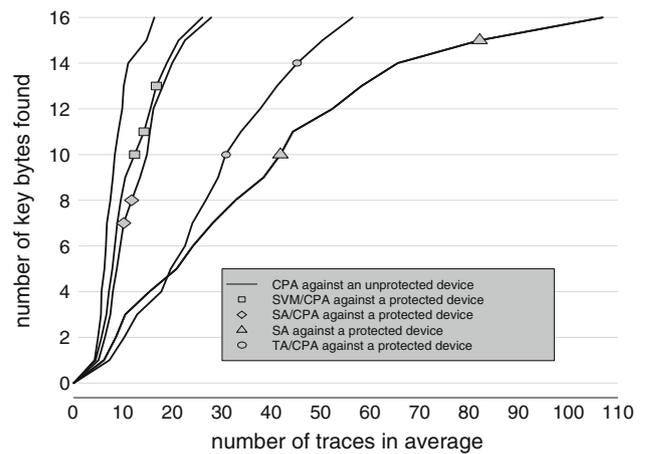


Fig. 15 Comparison of attacks against unprotected and protected AES

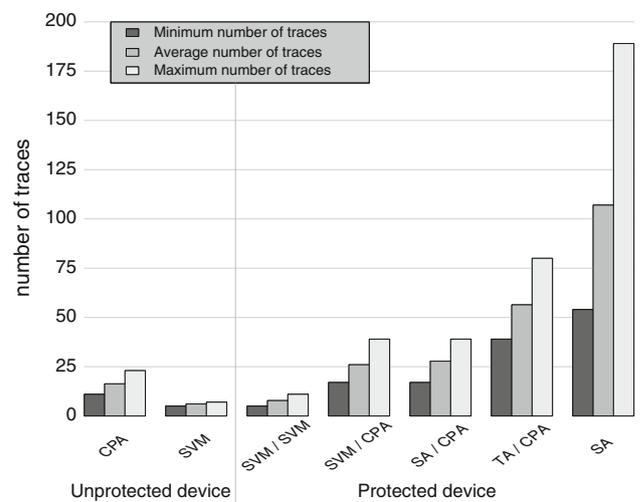


Fig. 16 Minimum, maximum and average number of traces used by each attack to find the key

in Sect. 2.1.1, we can either (a) apply a strategy (based on a non-profiling attack or a profiling attack) to a combination of power leakages, or (b) apply a non-profiling attack to multiple sample points independently and then combine their results. This section considers both methods.

We tested three combination functions which we note as COMB: the first combines the result of each correlation power analysis with the mean function, the second uses the weighted mean while the last uses the max function. Mathematically, the attack returns the target value \hat{O} that maximizes

$$\hat{O} = \arg \max_{O \in \mathcal{O}} \text{COMB}(|D_0|, \dots, |D_n|) \tag{29}$$

where D_i denotes a distinguisher applied on the set of traces measured at time i . We define the mean function as

$$\text{COMB}(|D_0|, \dots, |D_n|) = \frac{1}{n} \sum_i^n |D_i| \tag{30}$$

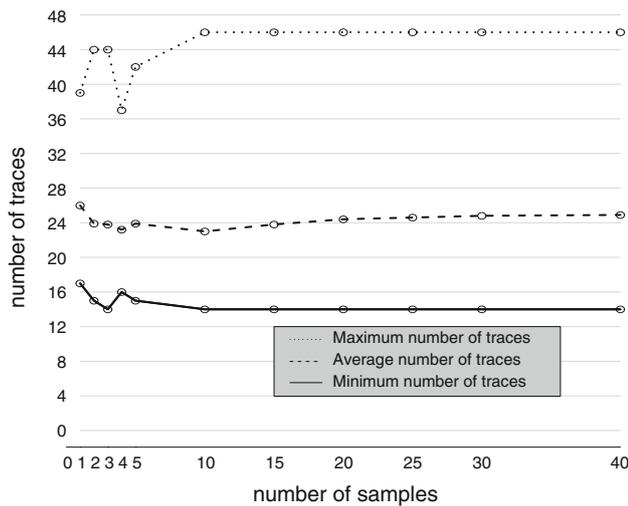


Fig. 17 SVM/multivariate CPA with the max function

The weighted mean function equals to

$$\text{COMB}(|D_0|, \dots, |D_n|) = \frac{1}{n} \sum_i^n \rho_i |D_i| \quad (31)$$

where ρ_i represents the Pearson correlation between the instant i on the traces and the i -th target value. Finally, the max function represents

$$\text{COMB}(|D_0|, \dots, |D_n|) = \max_{i \in \{1, 2, \dots, n\}} |D_i| \quad (32)$$

We varied the number of sample points between 1 and 40 to select the best value. Figures 17, 18 and 19 report the minimum, the average and the maximum number of traces used to extract the secret key in function of the number of samples combined respectively with the max, the mean and the weighted mean function. We reach the minimum of 23 traces in average with 10 samples when considering the max function. We improve this result with the mean and the weighted mean function: 20.9 traces in average with 40 samples.

The second approach combines first the points before to apply a key recovery attack. In practice, we used a support vector machine that learns the dependence between the power leakage (of different instants) and the target value that is the Hamming weight of the output of a MaskedSubBytes. Each support vector machine was build with 1,500 traces and tested with a validation set of 1,500 traces. We selected the best number of features between 2 and 40. Figure 20 shows the success rate of each support vector machine in function of the number of features used. In the best setting, the worst Support Vector Machine has a success rate of 0.85 while the best reaches a success rate of 0.95. After the selection of the best configuration, we used 3,000 traces (that were used previously) in the learning set to attack each MaskedSubBytes. Figure 16 shows the minimum, the average and the maximum number of traces when we use an /SVM (when there

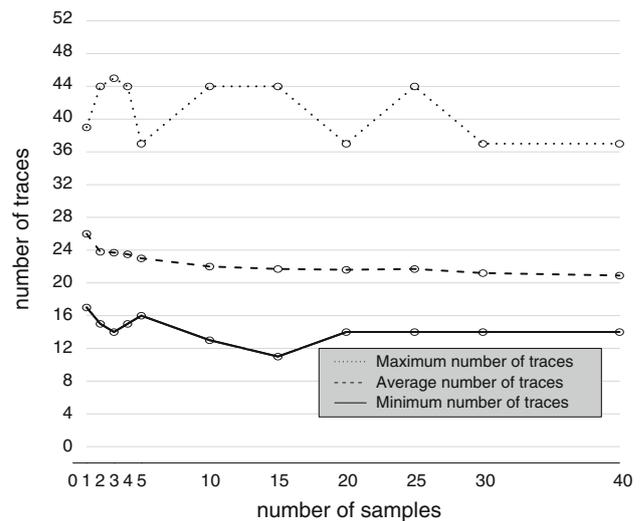


Fig. 18 SVM/multivariate CPA with the mean function

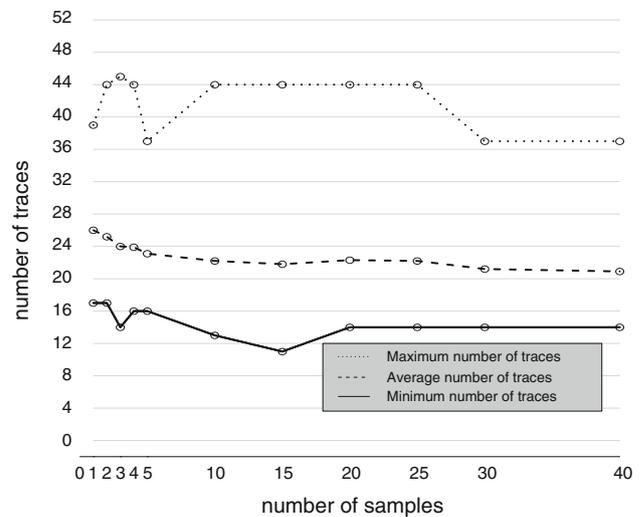


Fig. 19 SVM/multivariate CPA with the weighted mean function

are no protection) and an SVM/SVM (when there is a protection). The /SVM requires on average 6.1 traces while the SVM/SVM needs only slightly more, on average 7.8 traces. However, the SVM/SVM requires a longer execution time than SVM/CPA: less than 130 s. Table 1 resumes the results of strategies.

4.3 Discussions

The experimental results of the previous sections suggest some considerations. First, we have demonstrated that the masking scheme proposed by the DPAContest V4 can be practically attacked with a combination between profiled and non-profiled attacks. Our strategy represents a combination between a Support Vector Machine and a correlation power analysis or another set of machine learning models.

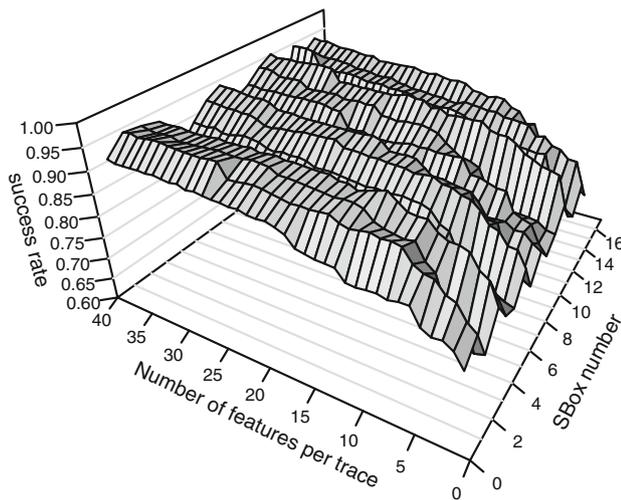


Fig. 20 SVM against the HW of the output of the MaskedSubBytes

Table 1 Summary results of strategies

Strategy	Number of traces
<i>Unmasked implementation</i>	
SVM	6.1
CPA	16.3
<i>Masked implementation</i>	
SVM/SVM	7.8
Multivariate CPA (mean function)	20.9
Multivariate CPA (weighted mean function)	20.9
Multivariate CPA (max function)	23
SVM/CPA	26
SA/CPA	27.8
TA/CPA	56.4
SA	107

The SVM/CPA requires 26 traces during the attacking step to extract the key of the implementation of a masked AES-128. In comparison, a correlation power analysis against an unmasked implementation required in average 16.3 traces. We improved our results using a support vector machine during the key recovery step (7.8 traces in average in order to find the key) but with the cost of a longer execution time. This result equals roughly to an unprotected context targeted by a support vector machine.

The support vector machine succeeds to extract information on the offset because the cryptographic device chooses different operations in function of this value (e.g. the choice of the masked SBox). Furthermore, the success of the attack is related to the implementation: the device manipulates the 16 state bytes sequentially while they can be manipulated in parallel on a FPGA. Moreover, the cryptographic device selects randomly only one offset during whole of the encryp-

tion. As a result, many points in a trace relate to the chosen offset.

The attack should be improved by increasing the number of points selected in each trace. Indeed, Fig. 3 shows that the maximum value of the success rate is still not reached by the support vector machine targeting the offset value. However, Fig. 12 shows that the learning step time increases linearly with the number of points selected in each trace. As a result, there is a trade off to be made between the accuracy of the model and its learning speed.

The major consideration concerns accuracy since the experimental results show that in several settings, when targeting the offset value, machine learning improves the success of attacks with respect to a strategy based on template attack or to the state-of-the-art stochastic attack. More precisely, a machine learning model needs four times less traces than the state-of-the-art stochastic attack on masking scheme and two times less traces than a strategy based on template attack. A new strategy based on stochastic attack becomes very competitive (in term of data complexity during the attacking phase) as the machine learning model but with a longer execution time than the support vector machine.

A possible justification of the superiority of machine learning models derives from the results of the Shapiro–Wilk multivariate Gaussianity test (SW test) [13] that we carried out on the 1,500 traces used during the learning step⁵ by template attack. The SW test was applied on several dimensions and against each offset value. Figure 21 displays several box plots that summarize the p -values of SW tests by taking into account from 2 to 50 dimensions (the same as for the previous experiments) against each offset value. In agreement with our results, SW test rejected the hypothesis of Gaussianity in 90.56 % of multivariate configurations. The Mardia’s test [34], another multivariate Gaussianity test, confirms the previous results: 96.17 % of configurations are rejected by the test. This test explains why template attack has a lower success rate than the other non-parametric models: a large majority of traces follow another unknown density probability function. As a result, strategies based on template attack (e.g. templates during preprocessing) should be less efficient than a non-parametric machine learning model.

We also deem that another possible justification of the strength of machine learning models relates to the number of parameters to estimate (since a higher number of parameters require a larger learning set). An interesting future work will focus on the number of parameters of support vector machine compared to template attack (that depends on the number of features) and to stochastic attack (that depends on the number of features as well as the degree of the regression model). It is worth to note that “*how good is my profile*” is still an open question [12].

⁵ The significance level of the Gaussianity test equals 0.05.

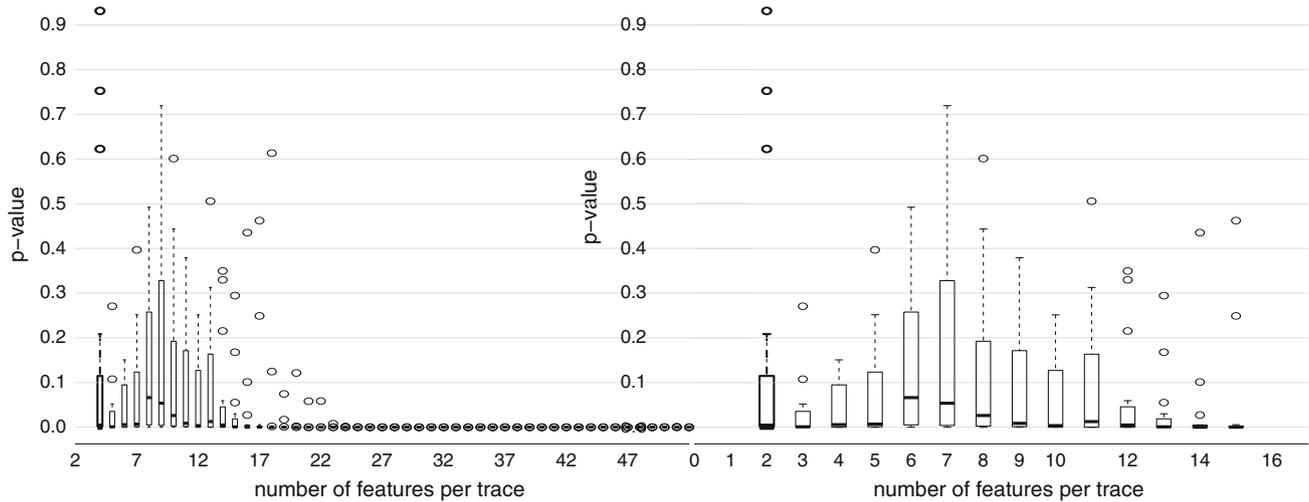


Fig. 21 Boxplots of Shapiro–Wilk multivariate Gaussianity test. In each box plot, the *central bar* corresponds to the median, the hinges to the first and third quartiles, and the whisker represents the great-

est/lowest value excluding outliers. A *p*-value is considered as an outlier when its value is more than $\frac{3}{2}$ times of upper/lower quartile

5 Conclusion and perspectives

In this paper, we introduced an efficient machine learning approach to evaluate the security level of a masked implementation of AES. Specifically, we extended the results of previous related works to protected devices [2,21–23,27–29]. The machine learning approach against a masked cryptographic algorithm consists in attacking first the mask with a machine learning model (i.e. a profiled attack) before targeting the secret key with a non-profiled attack or profiled attack.

We showed that the multivariate regression attack, the stochastic attack or a strategy based on template attack overestimates the security level of protected device while the machine learning approach improves significantly this estimation. The main reason of the superiority of machine learning with respect to template attack relates to the rejection of the multivariate Gaussianity tests that reject the hypothesis that the traces follow a Gaussian distribution in a high number of configurations. Therefore, a machine learning model extracts more information on the secret information (than template attack) by analyzing the same leakage information.

The complexity of the key recovery step mainly depends on the quality of the profiled model. The higher the success to retrieve the mask, the lower is the number of traces during the attacking phase. As a result, compared to a template attack, a learning model improves the probability to find the true mask value from 0.66 to 0.88 with a consequent reduction of the average number of traces during the attacking phase from 56.4 to 26 when considering a correlation power analysis during the key recovery step. Regarding the state-of-the-art stochastic attack, the learning model divides the number of

traces during the attacking phase by four. However, a new strategy based on stochastic attack reduces this number to 27.8 traces (in average) when considering a correlation power analysis during the key recovery step. In our context, the main advantage of a machine learning approach represents its speed: 80 s of execution time for a strategy based on stochastic attack while the machine learning model requires four times less. In comparison, a non-profiled attack against an unmasked implementation needs 17 traces with 5 s of execution time on the same cryptographic device. Therefore, the masked implementation increases the data complexity of the attack by two and the time complexity by four. We reduced significantly the data complexity when considering a machine learning model during the key recovery step: 7.8 traces in average to retrieve the secret key. This small number is roughly the same when the cryptographic device has no protection and targeted by a profiled attack.

The quality of the profiled attacks mainly depends on the number of points selected on the traces. A robust feature selection method allowed to reach a high success rate to find the mask value and the output of each sensitive information by the profiled model.

Interesting and as expected, the number of traces in the learning set of the machine learning model influences the result of the learning model targeting the mask values (the higher the number of traces in the learning set, the better). This is due to a reduction of the variance of the model.

The results of the DPAContest v4 validate the performance of our attacks. According to the DPAContest, the SVM/CPA approach requires 22 traces to find the correct key while the SVM/SVM approach needs 7 traces. As a result, in both cases, the hold-out validation method underestimates the

strength of the attacks in the (private) dataset of the DPA-Contest.

From an applied perspective, the machine learning approach represents an effective, efficiency and automatic black-box testing methodology capable of assessing the vulnerabilities of cryptographic hardware in the worst case scenario. The rationale is that data acquisition from a cryptographic device is expensive in terms of time and storage while the penetration tests have to be performed in a very short time (e.g. few weeks). As a result, strategies that reduce these constraints are preferable.

About the interpretability issue, machine learning provides a methodology to obtain black-box tools to evaluate the security level of a cryptographic device. Given its black-box nature it is not easy to deduce why an attack works better than another with a specific dataset. However, if the understandability of the attack is considered as more valuable than the accuracy of the results then other machine learning models and more white-box approaches should be pursued (such as decision trees). Furthermore, it would be interesting to compare the resulting accuracy with black-box approaches.

In light of our analysis, we believe that our work opens up new avenues for interesting further research works. Among them, we will consider other multivariate (profiled and non-profiled) attacks. Furthermore, experiments must be performed on different public datasets of masking or hiding implementations that will be available in the DPAContest V4. More precisely, the DPAContest V4.2 will provide traces collected on a software implementation of AES protected with an improved masking scheme to thwart collision attack as well as second-order CPA. On the other hand, the DPA-Contest V4.3 will provide traces collected on a hardware implementation of a masked AES where all the 16 Sboxes are processed in parallel. Another interesting possible future work concerns the scenario: “What would have been the results in a not worst-case scenario where, for example, the machine learning approach targets an information related partially to the mask values?” If such experiments confirm the above results, then there are important implications. Strategies based on template attack or stochastic attack against countermeasures scheme may be shown to be less suitable for security level estimation in the worst case scenario compared to a machine learning approach.

References

- Akkar, M.-L., Giraud, C.: An implementation of DES and AES, secure against some attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES. LNCS, vol. 2162, pp. 309–318. Springer, Berlin (2001)
- Bartkewitz, T., Lemke-Rust, K.: Efficient template attacks based on probabilistic multi-class support vector machines. In: Mangard, S. (ed.) CARDIS. LNCS, vol. 7771, pp. 263–276. Springer, Berlin (2012)
- Breiman, L.: Random forests. *Mach. Learn.* **45**, 5–32 (2001)
- Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards sound approaches to counteract power-analysis attacks. In: Wiener, M.J. (ed.) CRYPTO. LNCS, vol. 1666, pp. 398–412. Springer, Berlin (1999)
- Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES. LNCS, vol. 2523, pp. 13–28. Springer, Berlin (2002)
- Coron, J.-S., Naccache, D., Kocher, P.: Statistics and secret leakage. *ACM Trans. Embed. Comput. Syst.* **3**, 492–508 (2004)
- Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995)
- Dash, M., Liu, H.: Feature selection for classification. *Intell. Data Anal.* **1**(1–4), 131–156 (1997)
- Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., Weingessel, A.: e1071: Misc functions of the Department of Statistics (e1071), TU Wien. R package version 1.6 (2011)
- Doget, J., Prouff, E., Rivain, M., Standaert, F.-X.: Univariate side channel attacks and leakage modeling. *J. Cryptogr. Eng.* **1**(2), 123–144 (2011)
- DPAContest V4. <http://www.dpacontest.org/home/> (2014). Accessed 1 Feb 2014
- Durvaux, F., Standaert, F.-X., Veyrat-Charvillon, N.: How to certify the leakage of a chip? In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT. LNCS, vol. 8441, pp. 459–476. Springer, Berlin (2014)
- Gonzalez Estrada, E., Villasenor Alva, J.A.: mvShapiroTest: generalized Shapiro–Wilk test for multivariate normality. R package version 0.0.1 (2009)
- Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES. LNCS, vol. 2162, pp. 251–261. Springer, Berlin (2001)
- Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual information analysis—a generic side-channel distinguisher. In: CHES. LNCS, vol. 5154, pp. 426–442. Springer, Berlin (2008)
- Gierlichs, B., Janussen, K.: Template attacks on masking: an interpretation. In: Lucks, S., Sadeghi, A.-R., Wolf, C. (eds.) WEWoRC (2007)
- Gierlichs, B., Lemke-Rust, K., Paar, C.: Templates vs. stochastic methods. In: Proceedings of the 8th International Conference on Cryptographic Hardware and Embedded Systems. LNCS, vol. 4249, pp. 15–29. Springer, Berlin (2006)
- Golic, J.Dj., Tymen, C.: Multiplicative masking and power analysis of AES. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES. LNCS, vol. 2523, pp. 198–212. Springer, Berlin (2002)
- Hajra, S., Mukhopadhyay, D.: SNR to success rate: reaching the limit of non-profiling DPA. *Cryptology ePrint Archive*, Report 2013/865 (2013). <http://eprint.iacr.org/>
- Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction, 2nd edn. Springer, Berlin (2009)
- Heuser, A., Zohner, M.: Intelligent machine homicide—breaking cryptographic devices using support vector machines. In: Proceedings of the Third International Conference on Constructive Side-Channel Analysis and Secure Design. LNCS, vol. 7275, pp. 249–264. Springer, Berlin (2012)
- Hospodar, G., Gierlichs, B., Mulder, E.D., Verbauwhede, I., Vandewalle, J.: Machine learning in side-channel analysis: a first study. *J. Cryptogr. Eng.* **1**(4), 293–302 (2011)
- Hospodar, G., Mulder, E.D., Gierlichs, B., Vandewalle, J., Verbauwhede, I.: Least squares support vector machines for side-channel analysis. In: Second International Workshop on Constructive SideChannel Analysis and Secure Design, pp. 99–104. Center for Advanced Security Research, Darmstadt (2011)

24. Japkowicz, N., Stephen, S.: The class imbalance problem: a systematic study. *Intell. Data Anal. J.* **6**(5), 429–449 (2002)
25. Kocher, P.C.: Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) *CRYPTO*. LNCS, vol. 1109, pp. 104–113. Springer, Berlin (1996)
26. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: *CRYPTO*. LNCS, pp. 388–397. Springer, Berlin (1999)
27. Lerman, L., Bontempi, G., Markowitch, O.: Side channel attack: an approach based on machine learning. In: *Second International Workshop on Constructive SideChannel Analysis and Secure Design*, pp. 29–41. Center for Advanced Security Research, Darmstadt (2011)
28. Lerman, L., Bontempi, G., Markowitch, O.: Power analysis attack: an approach based on machine learning. *Int. J. Appl. Cryptogr.* **3**(2), 97–115 (2014)
29. Lerman, L., Bontempi, G., Ben Taieb, S., Markowitch, O.: A time series approach for profiling attack. In: Gierlichs, B., Guilley, S., Mukhopadhyay, D. (eds.) *SPACE*. LNCS, vol. 8204, pp. 75–94. Springer, Berlin (2013)
30. Lerman, L., Fernandes Medeiros, S., Bontempi, G., Markowitch, O.: A machine learning approach against a masked AES. In: Francillon, A., Rohatgi, P. (eds.) *International Conference on Smart Card Research and Advanced Applications (CARDIS)*. LNCS. Springer, Berlin (2013)
31. Liaw, A., Wiener, M.: Classification and regression by randomforest. *R News* **2**(3), 18–22 (2002)
32. Lomné, V., Prouff, E., Roche, T.: Behind the scene of side channel attacks. In: Sako, K., Sarkar, P. (eds.) *ASIACRYPT*. LNCS, vol. 8269, pp. 506–525. Springer, Berlin (2013)
33. Mangard, S., Oswald, E., Popp, T.: *Power Analysis Attacks—Revealing the Secrets of Smart Cards*. Springer, Berlin (2007)
34. Mardia, K.V.: Measures of multivariate skewness and kurtosis with applications. *Biometrika* **57**(3), 519–530 (1970)
35. Martinasek, Z., Zeman, V.: Innovative method of the power analysis. *Radioengineering* **22**(2), 586–594 (2013)
36. Messerges, T.S.: Securing the AES finalists against power analysis attacks. In: Goos, G., Hartmanis, J., Leeuwen, J., Schneier, B. (eds.) *FSE*. LNCS, vol. 1978, pp. 150–164. Springer, Berlin (2001)
37. Montminy, D.P., Baldwin, R.O., Temple, M.A., Laspe, E.D.: Improving cross-device attacks using zero-mean unit-variance normalization. *J. Cryptogr. Eng.* **3**(2), 99–110 (2013)
38. Moradi, A., Guilley, S., Heuser, A.: Detecting hidden leakages. *Cryptology ePrint Archive*, Report 2013/842 (2013). <http://eprint.iacr.org/>
39. Nassar, M., Souissi, Y., Guilley, S., Danger, J.-L.: RSM: a small and fast countermeasure for AES, secure against 1st and 2nd-order zero-offset SCAs. In: Rosenstiel, W., Thiele, L. (eds.) *DATE*, pp. 1173–1178. IEEE (2012)
40. Oswald, E., Mangard, S.: Template attacks on masking-resistance is futile. In: Abe, M. (ed.) *Topics in Cryptology—CT-RSA 2007*. LNCS, vol. 4377, pp. 243–256. Springer, Berlin (2006)
41. Pearson, K.: On lines and planes of closest fit to systems of points in space. *Philos. Mag.* **2**(6), 559–572 (1901)
42. Peng, H., Long, F., Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(8), 1226–1238 (2005)
43. Prouff, E.: DPA attacks and S-boxes. In: Gilbert, H., Handschuh, H. (eds.) *Fast Software Encryption*. LNCS, vol. 3557, pp. 424–441. Springer, Berlin (2005)
44. Rivain, M., Dottax, E., Prouff, E.: Block ciphers implementations provably secure against second order side channel analysis. In: Nyberg, K. (ed.) *FSE*. LNCS, vol. 5086, pp. 127–143. Springer, Berlin (2008)
45. Schindler, W.: Advanced stochastic methods in side channel analysis on block ciphers in the presence of masking. *J. Math. Cryptol.* **2**(3), 291–310 (2008)
46. Schindler, W., Lemke, K., Paar, C.: A stochastic model for differential side channel cryptanalysis. In: Rao, J.R., Sunar, B. (eds.) *CHES*. LNCS, vol. 3659, pp. 30–46. Springer, Berlin (2005)
47. Standaert, F.-X., Archambeau, C.: Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In: Oswald, E., Rohatgi, P. (eds.) *CHES*. LNCS, vol. 5154, pp. 411–425. Springer, Berlin (2008)
48. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The world is not enough: another look on second-order DPA. In: Abe, M. (ed.) *ASIACRYPT*. LNCS, vol. 6477, pp. 112–129. Springer, Berlin (2010)
49. Sugawara, T., Homma, N., Aoki, T., Satoh, A.: Profiling attack using multivariate regression analysis. *IEICE Electron. Express* **7**(15), 1139–1144 (2010)
50. von Willich, M.: A technique with an information-theoretic basis for protecting secret data from differential power attacks. In: Honary, B. (ed.) *IMA International Conference*. LNCS, vol. 2260, pp. 44–62. Springer, Berlin (2001)
51. Whitnall, C., Oswald, E.: Profiling DPA: efficacy and efficiency trade-offs. In: Bertoni, G., Coron, J.-S. (eds.) *CHES*. LNCS, vol. 8086, pp. 37–54. Springer, Berlin (2013)
52. Whitnall, C., Oswald, E., Mather, L.: An exploration of the Kolmogorov–Smirnov test as a competitor to mutual information analysis. In: Prouff, E. (ed.) *CARDIS*. LNCS, vol. 7079, pp. 234–251. Springer, Berlin (2011)