A machine learning approach against a masked AES

Liran Lerman[1,2] and Stephane Fernandes Medeiros[1] and Gianluca Bontempi[2] and Olivier Markowitch[1]

[1] Quality and Security of Information Systems, Département d'informatique, Université Libre de Bruxelles, Belgium
[2] Machine Learning Group, Département d'informatique, Université Libre de Bruxelles, Belgium

Abstract. Side-channel attacks challenge the security of cryptographic devices. One of the widespread countermeasures against these attacks is the masking approach. In 2012, Nassar *et al.* [21] presented a new lightweight (low-cost) Boolean masking countermeasure to protect the implementation of the AES block-cipher. This masking scheme represents the target algorithm of the DPAContest V4 [30]. In this article, we present the first machine learning attack against a masking countermeasure, using the dataset of the DPAContest V4. We succeeded to extract each targeted byte of the key of the masked AES with 26 traces during the attacking phase. This number of traces represents roughly twice the number of traces needed compared to an unmasked AES on the same cryptographic device. Finally, we compared our proposal to a stochastic attack and to a strategy based on template attack. We showed that an attack based on a machine learning model reduces the number of traces required during the attacking step with a factor two and four compared respectively to template attack and to stochastic attack when analyzing the same leakage information. A new strategy based on stochastic attack reduces this number to 27.8 traces (in average) during the attack but requires a larger execution time in our setting than a learning model.

Keywords: side-channel attack, masking, profiled attack, machine learning, stochastic attack, template attack.

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. For this, several cryptographic primitives can be used such as an encryption function. During the execution of an encryption algorithm, the device processes secret information. These secret information could

be retrieved with physical attacks against the physical device by analyzing unintentional leakages such as the power consumption [14], the processing time [13], the electromagnetic emanation [7] or a combination of them [28].

In recent years the cryptographic community explored new attacks based on machine learning models. These methods demonstrate that template attacks (that can be considered as the strongest leakage analysis in an information theoretic sense [4]) overestimate the security of embedded devices in several scenarios. Lerman *et al.* [15, 16] 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.* [10, 11] 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.* [9] generalized this idea by analyzing multi-class classification models in several contexts. In the same year Bartkewitz [1] applied a multi-class machine learning model allowing to improve the attack success with respect to the binary approach. Recently, Lerman *et al.* [17] proposed a machine learning approach that takes into account the temporal dependencies between power values. This method improves the success rate of an attack in a low signal-to-noise ratio with respect to classification methods. In the same year, Martinasek *et al.* [20] applied a neural network in order to extract one byte of the key of AES. Their method retrieves the secret value with probability around 0.9 using only one measured power leakage.

In parallel with attacks, the embedded systems industry implements countermeasures. They counteract side-channel attacks by inducing a leakage independent of the secret target value. It is worth to mention that so far all the attacks based on machine learning were applied on unprotected cryptographic devices. It was still unclear whether the results of the previous works are still the same in a protected environment. During the attacking phase, for a specific countermeasure and for a specific device, open questions are: (1) How many traces are required against a protected device with a machine learning model compared to a strategy based on template attack or based 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 attacking a protected device? We aim to answer these questions by proposing an original efficient combination between a machine learn-

ing model and a non-profiled attack. Our requirements are fast-execution, low-memory usage and high success rate of the attack (i.e. realistic attack scenarios).

We made a detailed assessment of the proposed approach by considering four public datasets with different number of traces during the profiling phase and ten public datasets during the attacking phase. These traces were collected on a smart card that implements the block-cipher AES protected by a masking scheme. All our datasets were extracted from the public dataset of the DPAContest V4 [30], "*an initiative towards an international benchmarking reference*" [30], thus allowing reproducing all our experiments.

This paper is organized as follows. Section 2 discusses non-profiled attacks, profiled attacks and masking 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 several datasets. Section 5 concludes this paper with several future works.

2 Preliminaries

2.1 Side-channel attack

During the execution of an encryption algorithm, the cryptographic device processes a function f

$$f \colon \mathcal{M} \times \mathcal{O} \rightarrow \mathcal{F} \tag{1}$$

$$s = f_O(m) \tag{2}$$

called a sensitive variable [25] where $O \in \mathcal{O} = \{O_0, O_1, ..., O_{K-1}\}$ is a key-related information where $\mathcal{O} = \{0,1\}^{l_1}$ and l_1 is the size of the secret value used in f (e.g. one byte of the secret key), $m \in \mathcal{M} = \{0,1\}^{l_2}$ represents a public information where l_2 is the size of the public value used in f (e.g. one byte of the plaintext) and $\mathcal{F} = \{0,1\}^{l_3}$ is the codomain of f where l_3 is the size of the output of f. The adversary targets this function during the attack.

Let

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

be the j-th leakage information (called trace) associated to the i-th target value. We consider the leakage information ${}^{j}_{t}T_i$ of the device at time t

depending on the output of $f_{O_i}(m)$ such that

$$ {}^{j}_{t}T_i = L(f_{O_i}(m)) + \epsilon \tag{4} $$

where $\epsilon \in \mathbb{R}$ is the noise following a Gaussian distribution with zero mean and L is the leakage model

$$ L \colon \mathcal{F} \to \mathcal{Q} \tag{5} $$

$$ Q = L(f_O(m)) \tag{6} $$

where $\mathcal{Q} \subset \mathbb{R}$. Examples of models L are the Hamming weight (HW) and the Hamming distance [19].

Non-profiled attack

The non-profiled attack represents a common approach in order to attack a cryptographic device. This attack estimates the output value of $f_O(m)$ 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. This paper focuses on univariate Correlation Power Analysis (CPA) where the distinguisher represents the Pearson correlation estimator.

Profiled attack

Let $\Pr[A]$ be the probability of A and let $\Pr[A \mid B]$ be the probability of A given B. The profiled attack strategy represents a more efficient attack by a deeper leakage estimation. It estimates (with a set of traces called learning set) a template $\Pr\left[{}^{j}T_i \mid L(f_{O_i}(m)); \theta_i\right]$ (where θ_i is the parameter of the probability density function) for each target value during the profiling step. The learning set is measured on a controlled device similar to the target chip. Once a template is estimated for each target value, the adversary classifies a new trace T (measured on the target device) during the attacking step with a profiled model $A(T)$ that computes the value $\hat{O}$ which maximizes the *a posteriori* probability

$$ \hat{O} = A(T) = \arg\max_{O \in \mathcal{O}} \Pr\left[L\left(f_O\left(m\right)\right) \mid T\right] \tag{7} $$

$$ = \arg\max_{O \in \mathcal{O}} \frac{\Pr\left[T \mid L\left(f_O\left(m\right)\right)\right] \times \Pr\left[L\left(f_O\left(m\right)\right)\right]}{\Pr\left[T\right]} \tag{8} $$

$$ = \arg\max_{O \in \mathcal{O}} \hat{\Pr}\left[T \mid \hat{L}\left(f_O\left(m\right)\right)\right] \times \hat{\Pr}\left[\hat{L}\left(f_O\left(m\right)\right)\right] \tag{9} $$

where the *apriori* probabilities $\hat{\Pr}\left[\hat{L}\left(f_O\left(m\right)\right)\right]$ are estimated by the user.

Several approaches exist in order to estimate $\Pr\left[T \mid L\left(f_O\left(m\right)\right)\right]$ such as the parametric template attack [4], the stochastic attack [27] and the non-parametric machine learning models [10, 15]. The former assumes that this probability follows a Gaussian distribution for each target value.

The stochastic attack modelizes the leakage model L at time t with a regression model ${}_th$, i.e.

$$\begin{aligned} {}^j_tT_i &= L\left(f_{O_i}\left(m\right)\right) + \epsilon && (10)\\ &= {}_th\left(f_{O_i}\left(m\right)\right) + {}_tR && (11)\\ &= {}_tc + \sum_{u=1}^{U} {}_t\alpha_u \; g_u\left(f_{O_i}\left(m\right)\right) + {}_tR && (12) \end{aligned}$$

where ${}_tR$ is a residual Gaussian noise at time t, $\{{}_tc, {}_t\alpha_1, {}_t\alpha_2, ..., {}_t\alpha_U\}$ is the parameter of the regression model ${}_th$ and $\{g_1, g_2, ..., g_U\}$ is the basis used in the regression. Usually each function g_j equals to

$$g_j\left(f_{O_i}\left(m\right)\right) = \mathrm{Bit}_j\left(f_{O_i}\left(m\right)\right) \tag{13}$$

where $\mathrm{Bit}_j\left(x\right)$ returns the j-th bit of x. Then, the attacker assumes that $\Pr\left[T \mid L\left(f_{O_i}\left(m\right)\right)\right]$ follows the Gaussian distribution $\mathcal{N}\left(h\left(f_{O_i}\left(m\right)\right), \Sigma\right)$ where $h(x)$ equals to $\{{}_1h(x), {}_2h(x), ..., {}_nh(x)\}$ and $\Sigma \in \mathbb{R}^{n \times n}$ is the covariance matrix of the residual term.

The non-parametric machine learning models make no assumption on the density distribution functions. For example Random Forest model (RF) [2] builds a set of decision trees that classifies a trace based on a voting system. Support Vector Machine (SVM) [5] discriminates traces associated to different target values with hyperplanes. We refer to [1, 9–11, 15–17] for deeper explanations on the parametric template attack and on 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 [3]. For this, the method masks a public information m with d uniformly distributed random values $v = \{v_0, v_1, ..., v_{d-1}\} \in \mathcal{V}^d$ 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. In a theoretical point of view, the security level of a masked implementation against side-channel attacks

increases exponentially with d [3] when the amount of noise in the traces is sufficient [29].

Potentially, an adversary can retrieve the secret information by 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. points correlated to $\mathrm{HW}\left(f_O\left(m \oplus v\right)\right)$ and to $\mathrm{HW}\left(v_i\right)$). Then, after this combination, a classical non-profiled attack is performed. However, it turns out that the mask values still influence the result of this combination (in a CPA context) and, as a result, an attack against a masked implementation needs more traces than against an unmasked implementation [22].

3 Machine learning approach against masking countermeasure

In a secure implementation context, it is necessary that the mask values remain secret. It is quite natural to wonder whether an adversary can retrieve information on these secret values by analyzing the leakage information. Indeed, once the mask values is revealed or removed, the attacker is able to execute an efficient non-profiled or profiled attack.

In 2008, Werner Schindler [26] extended the stochastic attack to a masking context by taking into account the mask value v in the deterministic part. The main advantage of this approach is that we need a smaller set of measurements during the profiling step compared to TA applied to masking [26].

Oswald *et al.* [22] evaluated several approaches to attack a masked implementation with a combination between TA and CPA. In the same year, Gierlichs *et al.* [8] 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. $\mathrm{HW}((f_O\left(m \oplus v\right))$ and $\mathrm{HW}\left(v_i\right)$) before combining them and to apply a CPA. 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 CPA 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. $\mathrm{HW}\left(f_O\left(m\right)\right)$) and performs a CPA on the extracted un-

masked sensitive value. The last approach (called Template based DPA) performs a template attack against the masking implementation by replacing $\Pr\left[T \mid L\left(f_O\left(m\right)\right)\right]$ in equation 9 with

$$\hat{\Pr}\left[T \mid L\left(f_O\left(m\right)\right)\right] = \sum_{v \in \mathcal{V}^d} \hat{\Pr}\left[T \mid L\left(f_O\left(m \oplus v\right)\right) \wedge v\right] \times \hat{\Pr}\left[v\right] \qquad (14)$$

As a result, we need $\text{card}(\mathcal{Q}) \times \text{card}(\mathcal{V}^d)$ templates (where $\text{card}(x)$ represents the cardinality of the set x), one for each possible combination of $L\left(f_O\left(m \oplus v\right)\right)$ and v.

We propose a new approach that uses a machine learning approach in order: (1) to bypass the problem of combining masks-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) to keep all traces in the attacking step (unlike the Templates During Preprocessing); (3) to 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. In a theoretical point of view: (i) the number of required data increases with the number of templates (c.f. we need one learning set per template that leads to a gigantic workload in the profiling step [26]) and (ii) the imbalanced class problem [12] arises in the Template based DPA according to the density distribution of $L\left(f_O\left(m\right)\right)$ (unlike our proposal). In a practical point of view, in the case of the DPA-Contest V4, the adversary has no control on the attacked device and, 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 - in order 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.

We suggest to apply a profiled attack to extract the mask values before a non-profiled attack that retrieves the secret key. 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 [1, 9–11, 15, 16]. 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 CPA that does not require any estimation probability density function. Note that our method can be extended to other (nonlinear) distinguishers.

4 Experiments and discussion

4.1 Target implementation

The experiments were carried out on electromagnetic emission leakages that are freely available on the DPAContest V4 website [30] in order 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*" [21]; 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 are the same, in the following we focus on AES-128.

Briefly, the masking scheme generates several mask values based on one 4-bit random value (called offset value) for each encryption. We refer to [21, 30] for additional information on the 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 CPA to find the key value. 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.

Finding the offset value on traces
Before proceeding with the quantitative analysis, we reports 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, we computed the Pearson correlation between each instant of 1500 traces and the offset values (see Figure 1). It is worth emphasizing that several instants are (significantly) correlated with the target value. Except in the middle of traces, the visualization suggests that there is a high amount of information on 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.

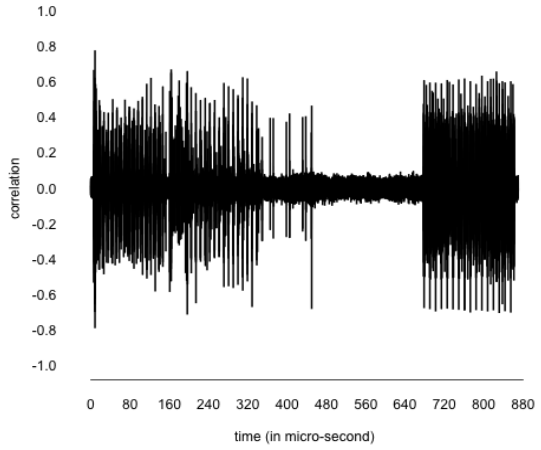

Fig. 1: Correlation between offset and power values at each time in the first round of the masked AES.

Model selection

This section assesses and compares several classifiers that extract the secret offset value. We considered four different types of multiclass classification models: Support Vector Machine (SVM), Random Forest (RF), Template Attack (TA) and Stochastic Attack (SA). We used two disjoint sets: a learning set of 1500 traces to estimate the parameters of each model and a validation set of 1500 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 highest linearly correlated with

the offset value[3]. We did not considered other feature selection methods (such as "*Principal Component Analysis*" [23] or "*minimum Redundancy Maximum Relevance*" [24]) due to their massive memory requirements or time consuming while our dataset contained $1500 \times 435,002$ bytes $> 2^{29}$ bytes[4]. In spite of the low feature selection complexity, we observed a high success rate of the models.

Figures 2, 3, 4 and 5 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 SVM, RF, TA and SA. We can extract the following observations. First, as expected, the higher the number of traces in the learning set (from 25% to 100% of 1500 traces), the higher the accuracy. Secondly, the number of selected points in each trace influences the success rate: the higher the number of features, the higher the success rates for SVM, RF and SA. It is interesting to remark that in a small learning set context (i.e. less than 75% of the entire learning set) the TA reduces its success rate when the number of features goes beyond a certain size. This is presumably due to the ill-conditioning of the covariance matrix when the number of features is too large. In the other hand, the success rate of the new proposal based on SA varies only slightly in function of the size of the learning set.

Figure 6 combines the three previous plots by choosing the best size for the learning set (i.e. 100% of 1500 traces). The success rates of SVM, RF and SA are similar and greater than the success rate of TA. Note that we did not select the best meta-parameter values for SVM and RF (such as the number of trees in the RF) but only the best number of features (from 2 to 50) to predict the target value. The default values of the implementation of SVM [6] and RF [18] were used[5]. As a consequence, we do not claim that the SVM configurations and the RF configurations are necessarily the best one for profiled attack neither for our experiments. However, our experiments show that a profiled attack based on a machine learning model extracts more information on the offset value than a strategy based on TA for the presented task.

Based on the above considerations, and in order 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

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

[4] 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.

[5] SVM had a radial kernel with a gamma equals to the inverse of the data dimension and a cost of 1. RF had 500 trees.

points (see Figures 8 and Figures 9)[6]. TA has the lowest learning time while its prediction time increases exponentially in the number of selected features. SVM has a lower learning time and a reasonable prediction time compared to RF. As a result, in the attacking step, we use only SVM as the machine learning model. We do not report the results for SA as we used an unoptimized and nonpublic implementation. According to the previous results, we selected 50 features for SVM, TA and SA leading to a success rate of respectively 0.88, 0.66 and 0.90.

Attacking step

During the attacking step we considered four settings targeting the Hamming weight of the MaskedSubBytes. In the first setting, CPA 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 SVM extracts the mask value and where a CPA searches the secret key. In the third and fourth experiments, the SVM is changed by respectively the TA and the SA. We repeated ten times each setting with a different set of traces during the attacking phase while the learning set remains the same.

Figure 10 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 seconds of execution time) for the unmasked implementation. For the masked implementation, we extracted the key with 26 traces (with less than 20 seconds of execution time) by using the SVM, with 27.8 traces (with less than 80 seconds of execution time) by using the SA and with 56.4 traces (with less than 45 seconds of execution time) by using the TA. Figure 11 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 SVM (combined with the CPA) 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 Werner Schindler [26]. Figure 7 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 Figure 10, SA needs 107 traces in average in order

[6] The experiments were executed on a MacBook Pro with 2.66 GHz Intel Core 2 Duo, 8 GB 1067 MHz DDR3.

to extract the 16 key bytes on a testing set (with less than 180 seconds of execution time).

4.3 Discussions

The experimental results of the previous sections suggest some considerations. First, 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 SVM and a CPA that required 26 traces during the attacking step to extract the key of the implementation of a masked AES-128. In comparison, in the same device, a CPA against an unmasked implementation required in average 16.3 traces. SVM 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 sixteen 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 encryption. 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, Figure 2 shows that the maximum value of the success rate is still not reached. However, Figure 8 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, machine learning improves the success of attacks with respect to a strategy based on TA or to the state-of-the-art SA. More precisely, a machine learning model needs four times less traces than the state-of-the-art SA on masking scheme and two times less traces than a strategy based on TA. However, a new strategy based on SA 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 machine learning model.

We submitted our best attack to the DPAContest V4 in order to have a validation of our results by a third party. According to this contest, the combination of SVM with CPA needs 22 traces with 0.528 seconds to retrieve a new AES-128 key with their computational power.

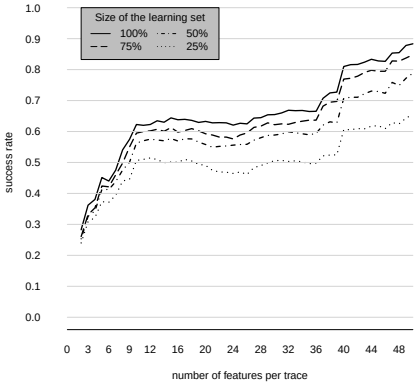

Fig. 2: Support Vector Machine.

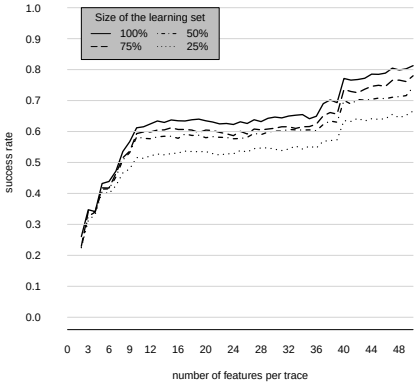

Fig. 3: Random Forest.

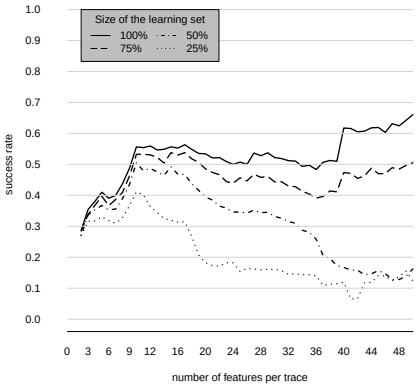

Fig. 4: Template Attack.

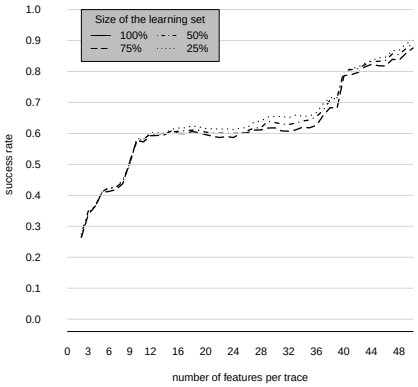

Fig. 5: Stochastic Attack.

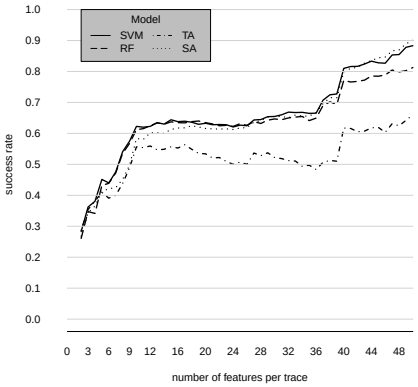

Fig. 6: SVM vs RF vs TA vs SA.

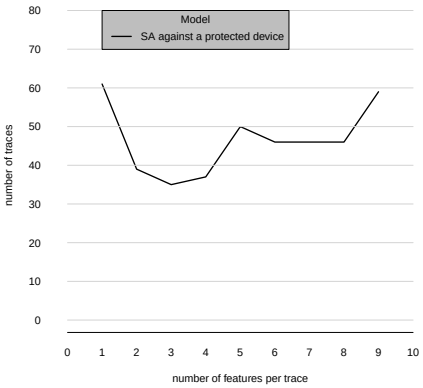

Fig. 7: Number of traces during the attacking phase in function of the number of features.

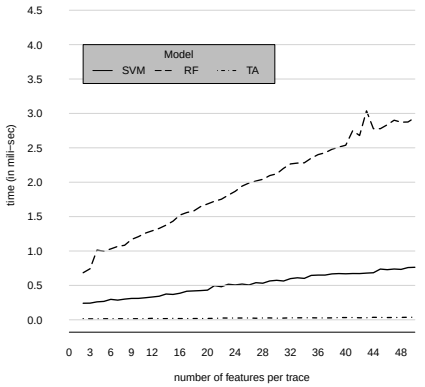

Fig. 8: Time to process a trace in the learning step (in mili-sec) per number of feature selected.

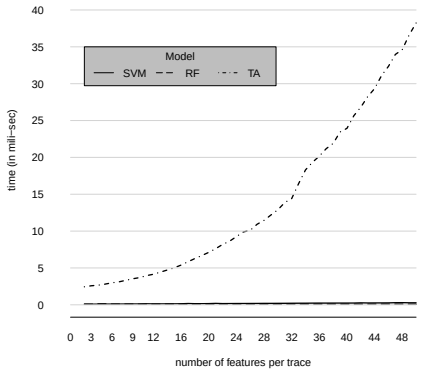

Fig. 9: Time to process a trace in the attacking step (in mili-sec) per number of feature selected.

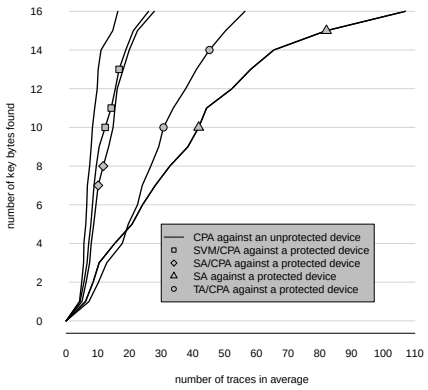

Fig. 10: Comparison of attacks against unprotected and protected AES.

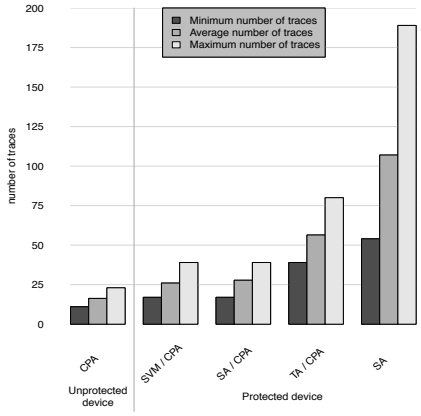

Fig. 11: Min., max. and average number of traces used by each attack to find the key.

5 Conclusion and perspectives

In this paper we have introduced an efficient machine learning approach in order to evaluate the security level of a masked implementation of AES. Specifically, we have extended the results of previous related works to protected devices [1, 9–11, 15–17]. 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 for example a non-profiled attack.

We showed that 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 arises with the result of the multivariate Gaussianity tests that we carried out and 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 non-profiling step mainly depends on the quality of the profiled model. The higher the success to retrieve the mask, the lower 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 that implies a reduction of the number of traces in average during the attacking phase from 56.4 to 26. 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) during the attack. In our context, the main advantage of a machine learning approach represents its speed: 80 seconds of execution time for a strategy based on SA while the machine learning model requires four times less. In comparison, a non-profiled attack against an unmasked implementation needs 17 traces with 5 seconds 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.

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 by the profiled model. Interesting and as expected, the number of traces in the learning set of the machine learning model influences the result in a masking context

(the higher the better). This is due to a reduction of the variance of the model.

We believe that our work opens up new avenues for interesting further research works. Among them, experiments must be performed on different public datasets of masking or hiding implementations which should be available in the DPAContest V4. 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

1. T. Bartkewitz and K. Lemke-Rust. Efficient template attacks based on probabilistic multi-class support vector machines. In S. Mangard, editor, *CARDIS*, volume 7771 of *LNCS*, pages 263–276. Springer, 2012.
2. L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
3. S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In M.J. Wiener, editor, *CRYPTO*, volume 1666 of *LNCS*, pages 398–412. Springer, 1999.
4. S. Chari, J.R. Rao, and P. Rohatgi. Template attacks. In B.S. Kaliski Jr., Ç.K. Koç, and C. Paar, editors, *CHES*, volume 2523 of *LNCS*, pages 13–28. Springer, 2002.
5. C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.
6. E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2011. R package version 1.6.
7. K. Gandolfi, C. Mourtel, and F. Olivier. Electromagnetic Analysis: Concrete Results. In Ç.K. Koç, D. Naccache, and C. Paar, editors, *CHES*, volume 2162 of *LNCS*, pages 251–261. Springer, 2001.
8. B. Gierlichs and K. Janussen. Template attacks on masking: An interpretation. In S. Lucks, A.-R. Sadeghi, and C. Wolf, editors, *WEWoRC*, 2007.
9. A. Heuser and M. Zohner. 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*, volume 7275 of *LNCS*, pages 249–264, Berlin, Heidelberg, 2012. Springer-Verlag.
10. G. Hospodar, B. Gierlichs, E.D. Mulder, I. Verbauwhede, and J. Vandewalle. Machine learning in side-channel analysis: a first study. *J. Cryptographic Engineering*, 1(4):293–302, 2011.
11. G. Hospodar, E.D. Mulder, B. Gierlichs, J. Vandewalle, and I. Verbauwhede. Least Squares Support Vector Machines for Side-Channel Analysis. In *Second International Workshop on Constructive SideChannel Analysis and Secure Design*, pages 99–104. Center for Advanced Security Research Darmstadt, 2011.
12. N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis Journal*, 6(5):429–449, October 2002.

13. Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In N. Koblitz, editor, *CRYPTO*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.
14. P.C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *CRYPTO*, LNCS, pages 388–397. Springer, 1999.
15. L. Lerman, G. Bontempi, and O. Markowitch. Side Channel Attack: an Approach Based on Machine Learning. In *Second International Workshop on Constructive SideChannel Analysis and Secure Design*, pages 29–41. Center for Advanced Security Research Darmstadt, 2011.
16. L. Lerman, G. Bontempi, and O. Markowitch. Power analysis attack: an approach based on machine learning. *International Journal of Applied Cryptography*, 2013. To appear.
17. L. Lerman, G. Bontempi, S. Ben Taieb, and O. Markowitch. A time series approach for profiling attack. In B. Gierlichs, S. Guilley, and D. Mukhopadhyay, editors, *SPACE*, volume 8204 of *LNCS*, pages 75–94. Springer, 2013.
18. A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.
19. S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.
20. Zdenek Martinasek and Vaclav Zeman. Innovative method of the power analysis. *Radioengineering*, 22(2):586–594, jun 2013.
21. M. Nassar, Y. Souissi, S. Guilley, and J-L. Danger. Rsm: A small and fast countermeasure for aes, secure against 1st and 2nd-order zero-offset scas. In W. Rosenstiel and L. Thiele, editors, *DATE*, pages 1173–1178. IEEE, 2012.
22. E. Oswald and S. Mangard. Template Attacks on Masking-Resistance Is Futile. In Masayuki Abe, editor, *Topics in Cryptology - CT-RSA 2007*, volume 4377 of *LNCS*, pages 243–256. Springer Berlin / Heidelberg, 2006.
23. K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.
24. H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, aug. 2005.
25. M. Rivain, E. Dottax, and E. Prouff. Block ciphers implementations provably secure against second order side channel analysis. In K. Nyberg, editor, *FSE*, volume 5086 of *LNCS*, pages 127–143. Springer, 2008.
26. W. Schindler. Advanced stochastic methods in side channel analysis on block ciphers in the presence of masking. *J. Mathematical Cryptology*, 2(3):291–310, 2008.
27. W. Schindler, K. Lemke, and C. Paar. A stochastic model for differential side channel cryptanalysis. In J. R. Rao and B. Sunar, editors, *CHES*, volume 3659 of *LNCS*, pages 30–46. Springer, 2005.
28. F.-X. Standaert and C. Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In E. Oswald and P. Rohatgi, editors, *CHES*, volume 5154 of *LNCS*, pages 411–425. Springer, 2008.
29. F.-X. Standaert, N. Veyrat-Charvillon, E. Oswald, B. Gierlichs, M. Medwed, M. Kasper, and S. Mangard. The world is not enough: Another look on second-order dpa. In M. Abe, editor, *ASIACRYPT*, volume 6477 of *LNCS*, pages 112–129. Springer, 2010.
30. DPAContest V4. http://www.dpacontest.org/home/, July 2013.