Power analysis attack: an approach based on machine learning

L. Lerman

Quality and Security of Information Systems & Machine Learning Group,
Department of Computer Science,
Université Libre de Bruxelles,
Brussels, Belgium
E-mail: llerman@ulb.ac.be

G. Bontempi

Machine Learning Group,
Department of Computer Science,
Université Libre de Bruxelles,
Brussels, Belgium
E-mail: gbonte@ulb.ac.be

O. Markowitch

Quality and Security of Information Systems,
Department of Computer Science,
Université Libre de Bruxelles,
Brussels, Belgium
E-mail: olivier.markowitch@ulb.ac.be

Abstract: In cryptography, a side-channel attack is any attack based on the analysis of measurements related to the physical implementation of a cryptosystem. Nowadays, the possibility of collecting a large amount of observations paves the way to the adoption of machine learning techniques, i.e., techniques able to extract information and patterns from large datasets. The use of statistical techniques for side-channel attacks is not new. Techniques like the template attack have shown their effectiveness in recent years. However, these techniques rely on parametric assumptions and are often limited to small dimensionality settings, which limit their range of application. This paper explores the use of machine learning techniques to relax such assumption and to deal with high dimensional feature vectors.

Keywords: cryptanalysis; side-channel attack; template attack; machine learning

Biographical notes: Liran Lerman received with honours his Master degree in the Department of Computer Science at the Universit Libre de Bruxelles (ULB, Brussels, Belgium) in 2010. Currently, he is a PhD student doing research as part of a Machine Learning Group (MLG) and the Cryptography and Security Service (QUALSEC) of that same university. He is conducting its PhD studies under the supervision of Professor Gianluca Bontempi and Professor Olivier Markowitch since October 2010. His research interest includes power analysis attack, timing attack, classification and data preprocessing.

Gianluca Bontempi graduated with honours in Electronic Engineering (Politecnico of Milan, Italy) and obtained his PhD in Applied Sciences (ULB, Brussels, Belgium). He took part to research projects in academy and private companies all over Europe. His interests cover data mining, machine learning, bioinformatics, time series prediction and simulation. He is author of more than 100 scientific publications. He is also the co-author of software for data mining and prediction which was awarded in two international competitions. From January 2002, he is Professor in Computer Sciences and Head of the Machine Learning Group of ULB.

Olivier Markowitch was graduated with honours in Computer Sciences and obtained his PhD in Computer Sciences (ULB, Brussels, Belgium). His research topics are related to cryptography, computer security and the design and analysis of cryptographic protocols. He is the author of about 40 scientific publications. He is currently an Associate Professor in Computer Sciences and the Co-head of the Cryptography and Computer Security Research Group at ULB.

1 Introduction

Side-channel attacks (Kocher *et al.*, 1999) take advantage of the fact that information leakage from a cryptographic device [e.g., instantaneous power consumption, encryption time (Kocher, 1996), electromagnetic leaks (Gandolfi *et al.*, 2001; Quisquater *et al.*, 2001) and acoustic effects (Shamir *et al.*)] may depend on the processed data and the performed operations. This paper focuses on power analysis attacks, a well-known instance of side-channel attacks, which assume that the use of different encryption or decryption keys implies different power consumptions, also referred to as traces. In particular, these attacks exploit the dependence between power consumption and a set of parameters of the cryptographic algorithm, like the encryption or decryption key, the plaintext or ciphertext, and the specific implementation. The growing interest in power attacks derives also from the fact that measurement technologies make nowadays possible the collection of a large amount of traces, simply by putting a resistor in series with the power or ground input.

Side-channel attacks can be categorised in two classes according to the strategy adopted to recover the key (Bogdanov *et al.*, 2010): divide-and-conquer and analytic attacks. The first type of attack recovers the key one chunk at a time while the latter finds the entire (sub)key in a single step (e.g., by solving a system of equations). The analytic strategy is used in the algebraic (Renauld *et al.*, 2009) and the collision attacks (Bogdanov, 2007). Here we will focus on a machine learning approach to implement a divide-and-conquer attack.

The evolution of the techniques proposed for power analysis attacks along the years has been characterised by an increase in the complexity of the statistical analysis. Simple power analysis (SPA) (Kocher *et al.*, 1999) has been the first approach proposed in literature for power analysis. SPA aims to deduce information about the used key by searching patterns in the trace linked to the executed operation.

Differential power analysis (DPA) (Kocher *et al.*, 1999) uses a more advanced statistical technique than SPA by modelling the theoretic power consumption for each key. The likelihood of the observed power consumption for each model is used to predict the key. The DPA can be resumed as follow. First, it selects a target, i.e., a function of the cryptographic algorithm that handles (a part of) the guessed key and a known value like the plaintext or the ciphertext. Second, it measures the real leakage during the execution of the cryptographic algorithm. Then, it makes predictions about the information leakage based on a leakage model applied to the target (e.g., Hamming weight of the target). Eventually, the real and the predicted power consumption are compared by using metrics, also known as distinguishers, like the correlation coefficient (Coron *et al.*, 2004), the difference of means (Kocher *et al.*, 1999) or the mutual information (Gierlichs *et al.*, 2008). The rationale is that the likelihood of a key is related to the degree of similarity between the predicted and the real power consumption.

The quality of the attack is based on the quality of the collected power consumption (measured by the signal-to-noise ratio), the quality of the leakage model and others parameters. For example, predicting the output of the first round S-boxes in a block cipher leads to a better discrimination of the key than predicting its input (Prouff, 2005).

The template attack (TA) (Chari *et al.*, 2002) makes another step forward in the use of statistical modelling for side-channel attacks, by estimating the conditional probability of the trace for each key in a parametric manner. This method relies on a parametric Gaussian estimation approach which appeared to be effective in practical cases (Mangard *et al.*, 2007). If this assumption holds, it can be considered as the strongest side-channel attack in an information theoretic sense. However, though this parametric approach is simple and easy to implement, it presents some shortcomings in configurations characterised by very long traces. For instance, a parametric Gaussian approach is prone to ill-conditioning when the number of traces is smaller than the number of features used to describe the trace (Schafer *et al.*, 2005).

This paper intends to make an original contribution in the statistical analysis of power consumption data by taking advantage of machine learning techniques. For a detailed introduction to machine learning, we refer the readers to (Alpaydin, 2009).

The role of machine learning in cryptanalysis has already been discussed in (Rivest, 1993). An application of machine learning to cryptanalysis is presented in (Backes *et al.*, 2010) where a machine learning algorithm is used to find information about the printed characters of a printer by exploiting the information hidden in the acoustic noise. A recent work on the application of machine learning to power analysis problem is presented in (Hospodar *et al.*, 2011). In their paper, the authors analyse a portion of the AES algorithm based on the XOR between an 8-bit subkey and the input word, followed by the application of a SBox. Though Hospodar et al.?s work on the use of machine learning in side-channel attacks is innovative, it leaves some space for improvement. First, they attack a single (and not complete) cryptographic algorithm by using a specific machine learning model (i.e., LSSVM). Second, the results do not show any significant improvement with respect to TAs. Third, the experimental configuration is characterised by a number of traces which is large and comparable to number of time points. It is interesting to study how the machine learning approach can extend to configurations where the number of time points is much larger.

Here, we focus on two aspects in order to make machine learning effective for power consumption analysis in real settings: the issue of dimensionality reduction and the one of model selection. The first aims

to extract from the observed data a minimal number of features able to take into account the information that the trace brings about the key. The second aims to go beyond the parametric assumptions made in TA by using techniques of model assessment and selection to find in a non-parametric and data-driven way the technique which provides the best accuracy in predicting the key.

We will show that a machine learning procedure based on dimensionality reduction and model selection is able to outperform conventional TA in high dimensionality settings by implementing two attacks. The first attack targets the bytes of the secret key of a symmetric cipher while the second attack concerns the bytes of the private key of an asymmetric cipher. We show that our approach implements attacks significantly faster than TA when only few traces are available. Then we will show that in our case the difficulty to predict a bit does not depend on the cryptographic algorithm but rather on the cryptographic device. Furthermore, we will study how the number of traces influences the quality of the attack in a high dimensionality context.

This paper is organised as follows: Section 2 introduces the notation and reviews the TA approach. Section 3 presents our machine learning approach to power analysis attack. A description of the experimental system and the results of an attack based on a machine learning technique are described in Section 4. Section 5 concludes the paper and discusses future work.

2 The template attack approach

A TA (Chari *et al.*, 2002) is based on the idea that the larger the information we have about the implementation, the more precise is the model of the device and its power consumptions. This kind of attack is interesting if only few traces can be obtained from the attacked device and a clone device for the training step is available.

Let us consider a crypto device executing a decryption/encryption algorithm with the binary key $O_i, i \in [1; K]$, where $K = 2^D$ is the number of possible values of the (sub)key and D is the number of bits (excluding each parity bit). In the following $B_{(b)(i)}$ represents the i-th bit of the b-th byte of the (sub)key (see Figure 1) while $B_{(b)}$ represents the b-th byte. In the context of RSA-512, we have 64 bytes per private key (i.e. $b \in [1; 64]$) and 8 bits per byte (i.e. $i \in [1; 8]$). Note that $B_{(b)(8)}$ (respectively $B_{(b)(1)}$) represents the Most Significant Bit (respectively Least Significant Bit) of each byte.

For each (sub)key, we observe N times the power consumption of the device over a time interval of length n and denote by *trace* the series of observations. Let $T^{(i)}_{(j)} = \left\{ T^{(i)}_{(j)(t)} \in \Re \mid t \in [1; n] \right\}$ be the j-th trace associated to the i-th key where $j \in [1; N]$.

Template Attack approaches model the stochastic dependency between the key and a trace by means of a

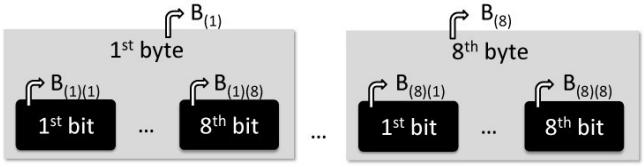

Figure 1 Representation of a cryptographic key where $B_{(j)(i)}$ represents the i-th bit of the j-th byte of the (sub)key.

multivariate normal conditional density

$$P(T^{(i)}_{(j)}|O_i; \mu_i, \Sigma_i) = \\ = \frac{1}{\sqrt{(2\pi)^n |\Sigma_i|}} e^{-\frac{1}{2}(T^{(i)}_{(j)} - \mu_i)\Sigma_i^{-1}(T^{(i)}_{(j)} - \mu_i)^t} \quad (1)$$

where $\mu_i \in \Re^n$ and $\Sigma_i \in \Re^{n \times n}$ are respectively the expected value and the covariance of the n variate traces associated to the i-th key.

In order to validate the multivariate normal hypothesis, some tests exist in literature, notably the kurtosis test (Nordhausen *et al.*, 2008) or the Mardia's test (Mardia, 1970). When considering one bit of the key the Gaussian hypothesis is rejected when at least one of the set of traces linked to a specific value of the bit presents no statistical evidence of normality. The main difference between the two tests is that the Mardia?s test is based on multivariate estimators of skewness and kurtosis measures while the kurtosis test is based essentially on the kurtosis estimation. Mathematically, the former computes two parameters (the skewness and the kurtosis measures) following two distinct distributions (resp. a chi-squared distribution and a standard normal) under the null hypothesis of multivariate normality. The latter estimates the kurtosis following the distribution of a quadratic form in p standard normal variables which is a linear combination of p chi-squared distributions with one degree of freedom.

A TA is made of two steps: a training phase (specific to TA and not present in DPA) and a classification phase. These phases are also known as learning (or profiling) and testing (or validation), respectively. During training, the expected value μ_i and the covariance Σ_i of the N traces (also known as training set) of the i-th key are estimated by

$$\hat{\mu}_i = \frac{1}{N} \sum_{j=1}^{N} T^{(i)}_{(j)} \quad (2)$$

and

$$\hat{\Sigma}_i = \frac{1}{N-1} \sum_{j=1}^{N} (T^{(i)}_{(j)} - \hat{\mu}_i)^t (T^{(i)}_{(j)} - \hat{\mu}_i), \quad (3)$$

respectively.

Once the training is done, the classification allows to classify traces T observed on a target device but for which no label is known. This set of traces is also known as validation set or testing set. The technique returns the

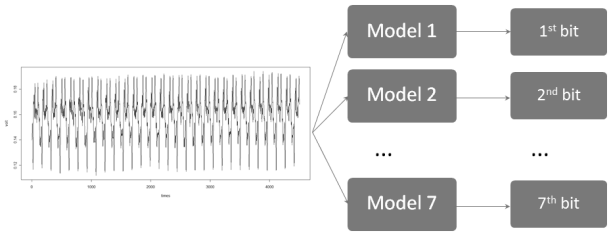

Figure 2 Decomposition of the prediction problem into a set of binary classification tasks

key which maximises the likelihood based on the Bayes theorem

$$\hat{k} = \arg\max_i \hat{P}(O_i|T) \tag{4}$$

$$\hat{k} = \arg\max_i \frac{\hat{P}(T|O_i) \times \hat{P}(O_i)}{\hat{P}(T)} \tag{5}$$

$$\hat{k} = \arg\max_i P(T|O_i; \hat{\mu}_i, \hat{\Sigma}_i) \tag{6}$$

where the *apriori* probabilities $\hat{P}(O_i)$ are estimated by the user.

Because of the Gaussian assumption the number of parameters to estimate amounts to $\frac{n^2+3n}{2}$ (i.e., $\frac{n^2+n}{2}$ for the covariance and n for the expected value). This number increases rapidly with the dimensionality and can become much larger than N for observation intervals of moderate size (e.g., $n > 20$). In order to reduce the size n, techniques of dimensionality reduction are typically adopted. Their aim is to extract a subset of p informative variables from the original set of n variables. A discussion of dimensionality reduction techniques will be provided in the following section.

3 Our approach

This paper proposes the adoption of a machine learning approach to estimate from a set of labelled traces the conditional distribution $P(O_i|T)$. In order to learn this dependency from data, we implement a procedure which relies on three steps: decomposition of the prediction task into D separate classification tasks, dimensionality reduction and model selection (Lerman *et al.*, 2011a). Although this is not the best way to perform a TA, the decomposition of the problem (Figure 2) is driven by the need of reducing the complexity of the tasks and the fact that the most common classification techniques address multi-input single-output problems. Once each single classification task is solved, the partial solutions are combined in order to have a probability distribution in the space of possible keys. This is known as the divide-and-conquer approach (Veyrat-Charvillon *et al.*, 2011) and has been proposed to reduce the average number of attacked keys.

Dimensionality reduction is necessary in order to deal with experimental settings where the number n of time steps is comparable or larger than the number N of collected traces. At the same time, model selection is used in order to avoid the parametric assumption made in TA and find in a data driven manner the model which best fits the stochastic dependency between key and power consumption.

3.1 Techniques of dimensionality reduction

Power traces can be represented as multidimensional or multivariate vectors, where each dimension (or variable) represents the power consumption of a device at a specific time during the execution of a cryptographic algorithm. Since it is possible that only a subset of variables carry relevant information about the targeted (sub)keys, dimensionality reduction has to be considered.

In what follows, after a general overview of dimensionality reduction we will detail four techniques which will be used later in our approach.

Overview on dimensionality reduction

Dimensionality reduction (also known as feature selection) aims to extract from the original n variables a subset of p informative variables. The advantages of dimensionality reduction are manifold: speed up of the learning process and more generally of the recovery of the key, enhancement of model interpretability, reduction of the amount of storage and improvement of the quality of models by mitigating the curse of dimensionality.

The curse of dimensionality is a well known problem in machine learning due to the fact that by increasing dimensionality, the sparsity of data increases at an exponential rate, too. This is a problem when considering classifiers which have to regroup traces linked to the same key. In order to address this use, feature selection is recommended.

Feature selection techniques may be regrouped into three main categories (Liu *et al.*, 2007): embedded approaches, filter approaches and wrapper approaches. In the embedded strategy, the feature selection is embedded in the classification algorithm. This means that the best subset of relevant variables and the parameters of classification models are searched simultaneously, like in classification trees. The filter approaches select features before using the classification algorithm. Finally, wrapper approaches use the classification models as black boxes to find the best subset of attributes. In this case, a classifier is learned for each feature subset in order to associate to each subset a measure of accuracy. Wrappers usually provide better results, the price being higher computational complexity.

Ranking

Ranking is the simplest filter technique which returns the p variables that have the highest variance.

This technique assumes that the degree of information of a variable is proportional to its variance.

Obviously, this assumption is extremely simplistic since no use of information about the target is made (i.e., it is an unsupervised criterion) and no notion of complementarity or redundancy of variables is taken into consideration.

Principal Component Analysis (PCA)

It is probably the most known statistical technique for dimensionality reduction (Pearson, 1901) and has been already used for side-channel analysis by (Archambeau *et al.*, 2006). PCA reduces the number of components of each trace $T_{(j)}^{(i)} \in \Re^n$ by first projecting it into a new set of n uncorrelated variables, named principal components and then selecting the p most variant ones. The trace projected in the new dimension (denoted eigen-trace) is noted $\widetilde{T}_{(j)}^{(i)} \in \Re^n$ and each of its component is a linear combination of the n components of $T_{(j)}^{(i)}$.

The rationale of PCA is to rank the new components according to their variance and to select only a subset of them, e.g., the first $p < n$ of them. This is due to the assumption that the components with the highest variance are the ones with the largest amount of information.

In algorithmic terms, the eigen-traces $\tilde{T}$ are computed by means of the n eigenvectors V_i and the n eigenvalues v_i of the covariance matrix of T. In geometric terms, the n eigenvectors denote the directions of the new space and the n eigenvalues correspond to the variance of the n components. By ordering the eigenvalues, it is then possible to order the new variables and to focus only on the p most variant.

An interesting feature of PCA is that it is possible to quantify the loss of information due to the selection of the first $p < n$ components by using the formula

$$\frac{\sum_{i=p+1}^{n} v_i}{\sum_{i=1}^{n} v_i} \tag{7}$$

Minimum redundancy maximum relevance (mRMR) filter algorithm

This filter technique was first proposed in the bioinformatics literature (Peng *et al.*, 2005) in order to deal efficiently with configurations where the number of variables is much larger than the number of samples. Minimum redundancy maximum relevance (mRMR) ranks variables by prioritising the ones which have a low mutual dependence (i.e., low redundancy) while still providing a large information about the output (i.e., large relevance).

The method starts by selecting the variable $r = \left\{ T_{(j)(t)}^{(i)} \mid i \in [1;K]\,; j \in [1;N] \right\}$ having the highest mutual information about the target variable $O = \{O_i \mid i \in [1;K]\}$. Then, given a set R of selected variables, the criterion updates R by choosing the variable $t = \left\{ T_{(j)(t)}^{(i)} \mid i \in [1;K]\,; j \in [1;N]\,; t \notin R \right\}$ that maximizes $I(t;O) - \frac{1}{|R|} \sum_{r \in R} I(t;r)$. This approach requires a reliable estimation of the mutual information quantity. In the experiments of this paper, we will make an assumption of Gaussian distribution of the variables in order to speed up the computation of the mutual information.

Self Organizing Map

Self-organising map (SOM) (Kohonen, 2001) is an artificial neural network which associates each trace with a neuron and organises the network of neurons in order to cluster together similar traces.

SOM can also be interpreted as a non-linear mapping from a high dimensional input to a low dimensional output since they provide a way to represent data in 2 or 3 dimensions while preserving the mutual distances between items of the training set.

For a given trace T, the model returns the value Y which minimises

$$Y = \arg\min_j d(T, \pi_j) \tag{8}$$

where d is a distance measurement and $\pi_j \in \Re^n$ is the vector describing the j-th neuron.

During the learning procedure, each trace T of the learning set is used in order to calibrate the vectors π_j according to the following equation:

$$\pi_j = \pi_j + \frac{\eta_t (T - \pi_j)}{\theta_{(t)(j)(Y)}} \tag{9}$$

where Y is chosen according to (8), η_t is the learning rate and $\theta_{(t)(j)(Y)}$ represents the distance between the Y-th neuron and the j-th neuron (when Y is equal to j then $\theta_{(t)(j)(Y)}$ is set to 1).

The parameter $\theta_{(t)(j)(Y)}$ in (9) increases with the time (symbolised by t, an iteration number during the training step) and is used for two main purposes. When its value is low it allows a global organisation of the map, whereas when its value increases it lets each neuron tailor those traces which are most frequently mapped onto it. In other words, each neuron plays the role of a prototype of a set of neighbouring traces.

The learning rate η_t, the second parameter of (9), is set to a high value in the beginning in order to have a rapid adaptation of all neurons. Then, it is progressively decreased to allow the neurons to diversify.

In what follows the notation SOM($x \times y$) will be used to denote a SOM with $x \times y$ neurons (i.e. each neuron has a coordinate $(i;j)$ such as $i \in [1;x]$ and $j \in [1;y]$).

Note that the power of approximation of a SOM is related to the number of neurons. This means that if on one hand, having more neurons reduces the bias of the approximation, on the other it exposes the model to a higher variance, with a consequent risk of overfitting. In other words, by increasing the number of neurons we may obtain better accuracy for the learning set but at the price of a worse generalisation, i.e., a worse prediction accuracy on the validation set.

3.2 Learning machines

In this subsection, after a general introduction to learning machines, we describe three learning algorithms, also named classifiers, which we will use in the experimental session for classifying the power traces. The aim of a classifier is to learn from observed traces the unknown (i) relationship between a trace $T_{(j)}^{(i)}$ (the input) and the key O_i (the output).

Overview on learning machines

In a conventional machine learning procedure, feature selection is followed by a *model selection* or structural identification step, which aims to select from a set of candidate models the best one. During this step, the family of classifiers (e.g., linear discriminant or neural networks) as well as the values of the hyper parameters (e.g., the degree of the polynomial or number of hidden neurons) are typically set.

This step aims to infer the most appropriate complexity of the model on the basis of a finite set of observations. This issue is also known in statistics as the *bias and variance tradeoff* where the bias is an indicator of an excessive simplicity of the model and the variance measures the instability of the model due to an excess of complexity. It is indeed well known that if on one hand too simple models are not able to capture complex non-linear dependencies (i.e., they underfit) on the other hand too complex models are sensitive to noise (i.e., they overfit the data).

The main goal of a model selection step is to return the model which has the lowest combination of bias and variance. In order to assess and select the best model structure, it is therefore necessary to estimate the accuracy of the model. This demands first the fitting for each alternative structure of the model parameters and then the validation of the fitted model on some independent test set. The fitting step is also known as *parametric identification* and takes different names according to the nature of the model, e.g., least-squares in linear models, convex optimisation in support vector machines (SVMs) or backgropagation in neural networks. The validation step is commonly performed in machine learning by adopting cross-validation or leave-one-out strategies (Section 3.3)..

It is well-known in literature that the final accuracy of the classifier is more sensitive to the structure selection than to the parameter fitting (Bishop, 1996). For that reason, we focus in this paper on the model selection procedure. As far as parametric identification is concerned, we limit our analysis to the standard implementations available in well-known R packages [e.g., the SVM implementation in the package `e1071` (Dimitriadou *et al.*, 2011)].

Self Organizing Map (SOM)

SOM, whose unsupervised version has been detailed previously, can also be used as a supervised model (Melssen *et al.*, 2006) when the key associated to each trace of the training set is given. In this paper, we adopt the bi-directional Kohonen (BDK) map.

A BDK builds two SOMs. The first one, named Xmap, deals with the input data and is composed of the vectors π_i^1 where each vector π_k^1 is associated to the k-th neuron in Xmap. The second one, named Ymap, has the same size of Xmap, deals with the output data and is composed of the vectors π_i^2.

Creating a BDK is done in two steps. During the first step, each trace in the training set is presented to the BDK network and the vectors π_k^1 in the Xmap are updated. The neuron in Xmap which is closest to a trace is determined by the Ymap according to the following equation:

$$K = \arg\min_k d(T, \pi_k^2) \tag{10}$$

.

In the second updating pass, only the Ymap is updated object-wise by using the winner determined by Xmap. Hence, Xmap and Ymap are updated in an alternating bi-directional way.

For a given trace T, the model returns the output of a neural Ymap located in the network at the same position than the one in Xmap which is the nearest to T.

Support Vector Machine (SVM)

SVM is one of the most successful techniques in classification (Cortes *et al.*, 1995) and has been recently used in (Hospodar *et al.*, 2011) for side-channel analysis. In a binary classification setting, if the two classes are separable, the SVM algorithm is able to compute from data the separating hyperplane with the maximal margin, where the margin is the sum of the distances from the hyperplane to the closest data points of each of the two classes. Let the input space be the space of traces $T \in \Re^n$ and the binary target values be $O_1 = 1$ and $O_2 =$?1. The SVM classification computes the parameters b and w of the separating hyperplane $[w^t T + b]$ by solving the following convex optimisation problem:

$$\min_w \frac{1}{2}(w^t w) \tag{11}$$

subject to

$$O_i(w^t T_{(j)}^{(i)} + b) \geq 1 \; \forall i \in [1;2]\,, j \in [1;N] \tag{12}$$

. In non separable setting the formulation is changed by introducing a set of slack variables $\xi_j^i \geq 0$ with $i \in [1;2]\,, j \in [1;N]$ then leading to the problem

$$\min_w \frac{1}{2}(w^t w) + C \sum_{i=1}^{2} \sum_{j=1}^{N} \xi_j^i \tag{13}$$

subject to

$$O_i(w^t T_{(j)}^{(i)} + b) \geq 1 - \xi_j^i \; \forall i \in [1;2]\,, j \in [1;N] \tag{14}$$

$$C \geq 0 \tag{15}$$

$$\xi_j^i \geq 0 \tag{16}$$

. A larger C means that a higher penalty to classification errors is assigned.

An interesting feature of SVM is that it is possible to adapt the classifier to non-linear classification tasks by performing a non-linear transformation κ of the inputs. This function is named kernel function and can have several forms (e.g., linear, polynomial, radial basis function, sigmoid). Its purpose is to find a linear separation in a higher dimension if there is no linear separation in the initial dimension.

Random Forest (RF)

The random forest (RF) (Breiman *et al.*, 2001) algorithm was introduced by Breiman in 2001 to address the problem of instability in large decision trees, where by instability we denote the sensitivity of a decision tree structure to small changes in the training set. In other words, large decision trees prone to high variance, this resulting in high prediction errors.

In order to reduce the variance, this method relies on the principle of model averaging by building a number of decision trees and returning the most consensual prediction. This means that the predicted key O of an unlabeled observation T is calculated through a majority vote of the set of trees.

RF is based on two aspects. First, each tree is constructed with a different set of traces through the boostrapping method. This method builds a bootstrap sample for each decision tree by resampling (with replacement) the original dataset. Observations in the original dataset that do not occur in a bootstrap sample are called out-of-bag observations and are used as a validation set. Secondly, each tree is built by adopting a random partitioning criterion. This idea allows to obtain decorrelated trees, thus improving the accuracy of the resulting RF model.

In conventional decision trees each node is split using the best split among all variables. In the case of a RF, each node is split using the best among a subset of variables randomly chosen at that node. Also, unlike conventional decision trees, the trees of the RF are fully grown and are not pruned. In other words, each node contains traces linked to a value of the key. This implies null training error but large variance and consequently a large test error for each single tree. The averaging of the single trees represents a remedy to the variance issue without increasing the bias, and allows the design of an overall accurate predictor.

3.3 Validation technique

In order to assess the predictive power of our models and to select the best one, we adopt a leave-one-out validation strategy. This strategy demands a number N of rounds. Each round uses $N-1$ traces to learn a model and the remaining trace to assess the generalisation accuracy that is the accuracy in predicting keys associated to traces not belonging to the training set. This is repeated until all traces have been used for testing purposes. The best model configuration (in terms of features and learning machine) is the one which minimises the error computed by leave-one-out.

Note that the aim of the validation is not to perform an attack but rather to assess robustly the rate of success of an attack in a statistically equivalent context. When the attacker wishes to proceed with the attack, she will take advantage of the results of the validation by choosing the best model, retraining it on the whole set of labelled traces and then applying it to classify unlabeled traces.

4 Experiments and discussion

We carried out two experiments on real power consumption data. The first one concerns a 3DES algorithm (Section 4.1) while the second deals with an RSA-512 algorithm (Section 4.2). Both algorithms run on the same cryptographic device, an FPGA Xilinx Spartan XC3s5000 with frequency around 33 MHz. Section 4.3 discusses the main considerations resulting from the two experiments. The whole data analysis procedure is implemented in the R language by means of the package `sideChannelAttack` (Lerman *et al.*, 2011b) available on CRAN.

4.1 Experiments on 3DES

Device under attack

This attack concerns a 3DES algorithm that encrypts a constant message of 64 bits chosen at random. In our experiment, triple DES uses three different keys of 56 bits (excluding parity bits) in encrypt-decrypt-encrypt (EDE) mode.

For the sake of simplicity, we restrict to consider attacks of a single byte (e.g., 7 non-parity bits) of the key bundle at the time. This means that we consider a target value O_i where $i \in [1;128]$.

Note that in the following we will use synthetically the term key to denote the target of our attack, though, in fact, we address one byte at the time.

Measurement Setup

For practical reasons, we measured traces with two oscilloscopes: an Agilent infiniium DSO80204B (2 Ghz 40 GSa/s) and an Agilent infiniium DSO8104A (1 GHz 4 GSa/s) oscilloscope. The first one collects traces of $20,000$ points containing n $= 9,399$ values associated to encryption. The second oscilloscope collects traces of length $5,999$. Except the last part of experiments on 3DES (i.e., the '*generalisation to the other DES bytes*' part) where the second oscilloscope was used, the other parts refer to the first oscilloscope.

3D visualization

Before proceeding with the quantitative analysis, we reports here a preliminary visualisation of the distribution of the traces associated to the byte $B_{(8)}$ of 3DES. Since for each value of the key we have $N = 400$ power consumption traces, we first filter out the noise by computing for each O_i the average trace value $\hat{\mu}_i \in \Re^{9399}$:

$$\hat{\mu}_i = \frac{1}{400} \sum_{j=1}^{400} T_{(j)}^{(i)} \tag{17}$$

Then, a preliminary visualisation of the dependence between $\hat{\mu}_i$ and O_i is obtained by representing a projection of the n dimensional traces in a tridimensional space. In order to visualise the trace distribution, we use the first three PCA components (V_1, V_2 and V_3 in Figure 3) causing only 25.77% (see (7)) of loss of information. The seven subfigures of Figure 3 correspond to seven bits of the $B_{(8)}$ byte ($B_{(8)(1)}$, $B_{(8)(2)}$, $B_{(8)(3)}$, $B_{(8)(4)}$, $B_{(8)(5)}$, $B_{(8)(6)}$, $B_{(8)(7)}$). Points with equal greyscale denote traces associated to keys having the same bit value. The visualisation suggests that traces, linked to different values of their lower bits, are less separable. As a consequence, we should expect that those bits will be more difficult to predict in this high dimensionality context.

Model selection

This section assesses and compares several classifier configurations by using a leave-one-out approach. Note that, for the sake of coinciseness, we limit here to report results concerning the byte $B_{(8)}$ of 3DES.

As discussed in Section 3, we build a different classifier for each non-parity bit of the byte $B_{(8)}$. We considered three different types of models and four types of feature selection. In the following, the notation A/B is used to denote the classifier configuration with the learner A and feature selection algorithm B.

The assessed configurations in this paper are listed below:

- SOM(8×5) / Nosel
- SOM(9×5) / Nosel
- SOM(8×6) / Nosel
- SOM(9×6) / Nosel
- SVM (kernel radial and $C = 1$) / Rank
- SVM (kernel radial and $C = 1$) / Nosel
- RF (500 trees) / Rank
- RF (500 trees) / Nosel
- RF (500 trees) / SOM
- RF (500 trees) / PCA

where Nosel means that no dimensionality reduction is carried out (i.e., 9,399 dimensions were considered) while the number of dimensions tested for SVM/rank, RF/rank, RF/SOM and RF/PCA ranges between 1 and 120. It is worthy to remark here that, again for the sake of space, we do not report the results of all combinations of dimensionality reduction and learning techniques. We prefer to show a reasonable sample of alternatives by giving priority to the techniques which appeared to be more accurate, like RF and SVM.

The leave-one-out accuracy percentage for different learning configurations and the different bits are reported in Table 1. Note that the accuracy of some bits amounts to 50%, meaning that for these bits the classifier accuracy is not better than random.

Table 1 highlights that the most accurate learning configuration is the one made by a PCA algorithm and a RF learner in this high dimensionality context. Indeed, by performing the product of probabilities of bits for each model (column ?entire byte? in the table), we can see that RF/PCA obtains the highest score. Therefore, in the following the RF/PCA learning configuration is used to attack each key byte of 3DES.

Sensitivity to the number of traces

In the previous sections, we applied an average of 400 traces for each key in order to reduce the noise. In an attack perspective, it is however important to determine how much the resulting accuracy is sensitive to the amount of traces. In order to address this issue, we attack $B_{(8)}$ of 3DES by means of RF/PCA and restrict the number of traces per key to 50, 150, 250, 400, respectively. The success rate (between 0% and 100%) is returned by the product of the success rates of each attacked bit and is shown in Figure 4 as a function of the number of features.

Two considerations can be made on the basis of this analysis. First, as expected, the higher the number of traces per key, the higher the signal-to-noise ratio and the associated accuracy. Second, by reducing the number of traces, the dimensonality reduction procedure avoids the risk of overfitting by reducing accordingly the number of selected features. For such number of features, in spite of a drastic reduction of the number of traces (from 400 down to 50) the RF/PCA returns a reasonably accurate performance in this high dimensionality setting.

Comparison between Machine Learning and Template Attack

In this section, we compare the accuracy of the RF/PCA model to the one of TA. For that reason, we carry out a set of attacks against a byte of the key under the same conditions. This means that the following parameters are identical for both attack strategies:

1. the oscilloscope
2. the device

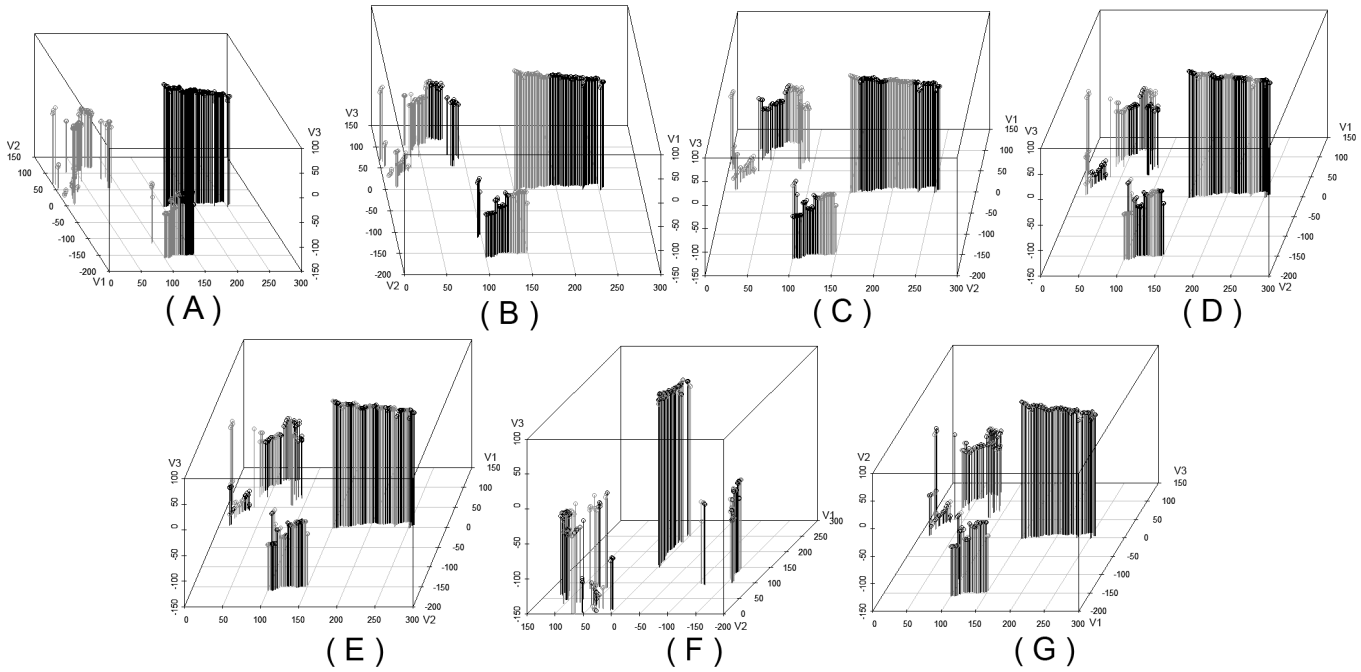

Figure 3 This figure shows the 128 traces, from the 8-th byte of the first key ($B_{(8)}$) of 3DES, projected in 3D. The black dots represent a bit value 1 and the others symbolize a bit value 0. Points of the same grayscale indicate the same value of the 7-th bit ($B_{(8)(7)}$) in A, of the 6-th bit ($B_{(8)(6)}$) in B, of the 5-th bit ($B_{(8)(5)}$) in C, of the 4-th bit ($B_{(8)(4)}$) in D, of the 3-rd bit ($B_{(8)(3)}$) in E, of the 2-nd bit ($B_{(8)(2)}$) in F, and of the 1-st bit ($B_{(8)(1)}$) in G.

	7-th bit	6-th bit	5-th bit	4-th bit	3-rd bit	2-nd bit	1-st bit	Dim	entire byte
SOM(8×5)	*96.09*	*90.23*	*87.50*	*74.22*	*53.52*	*50.00*	*50.00*	9399	7.53
SOM(9×5)	*97.27*	*92.19*	*83.98*	*69.53*	*57.81*	*51.17*	*50.00*	9399	7.74
SOM(8×6)	*96.48*	*89.45*	*83.59*	*73.44*	*57.42*	*50.00*	*50.00*	9399	7.61
SOM(9×6)	*95.70*	*92.97*	*85.94*	*78.52*	*58.20*	*51.56*	*50.00*	9399	9.01
SVM / Rank	*94.53*	*80.47*	*72.66*	*62.5*	*50.00*	*50.78*	*50.00*	20	4.39
SVM / Nosel	*96.48*	*90.23*	*82.81*	*73.05*	*64.06*	*53.52*	*50.00*	9399	9.03
RF / Rank	*97.66*	*83.98*	*81.64*	*77.34*	*61.33*	*57.42*	*50.00*	20	9.12
RF / Nosel	*96.09*	*92.58*	*89.06*	*83.98*	*59.77*	*55.47*	*50.00*	9399	11.03
RF / SOM	*96.48*	*89.06*	*82.81*	*76.17*	*60.94*	*50.00*	*50.00*	20	8.26
RF / PCA	*96.09*	*92.58*	*90.63*	*85.55*	*75.39*	*58.98*	*50.00*	14	15.33

Table 1 The leave-one-out accuracy percentage for different learning configurations and the different bits. "Dim" denotes the number of selected variables while the "entire byte" denotes the probability to predict the entire byte correctly.

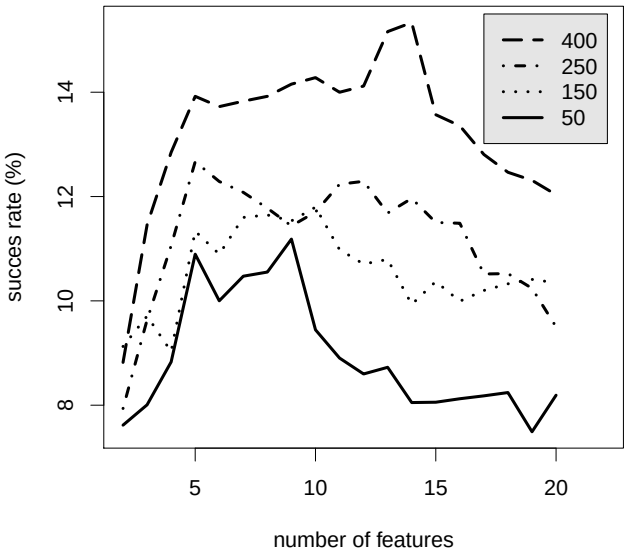

Figure 4 Leave-one-out success rate for 3DES obtained with RF/PCA with different number of traces $N \in \{50, 150, 250, 400\}$.

3. the implemented encryption scheme
4. the probes
5. the number of traces (400)
6. the measured traces
7. the attacked byte (the byte $B_{(8)}$ of 3DES)
8. the validation technique (leave-one-out)

Note that we limited our analysis to consider the byte $B_{(8)}$ since for that specific byte we have traces measured with the most accurate oscilloscope.

The comparison is done in terms of success rate (the higher the better).

We reduced the number of points for each trace through a feature selection method. The large dimensionality of the traces requires the adoption of a dimensionality reduction technique during the TA?s training step. For the sake of comparison we considered here PCA, the mRMR filter and the sum of squared pairwise T-differences (SOST) filter (Gierlichs *et al.*, 2006).

The accuracy of the TA/mRMR attack as a function of the number of features is reported in Figure 5, the accuracy of the TA/PCA attack is shown in Figure 6 and the accuracy of the TA/SOST attack is summarised in Figure 7.

It is interesting to remark that the TA is not reliable at all when the number of features goes beyond a certain size (see Figure 5). This is presumably due to the ill-conditioning of the covariance matrix when the number of features is too large. The adoption of a regularised

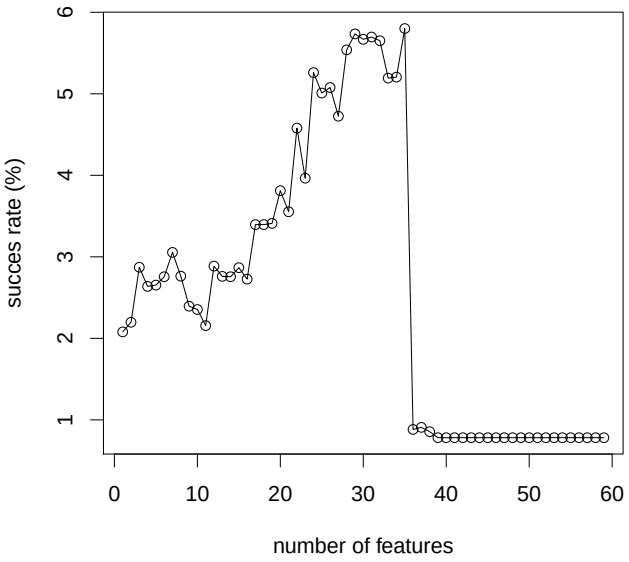

Figure 5 3DES: rate of correct classification vs. number of variables with TA/mRMR.

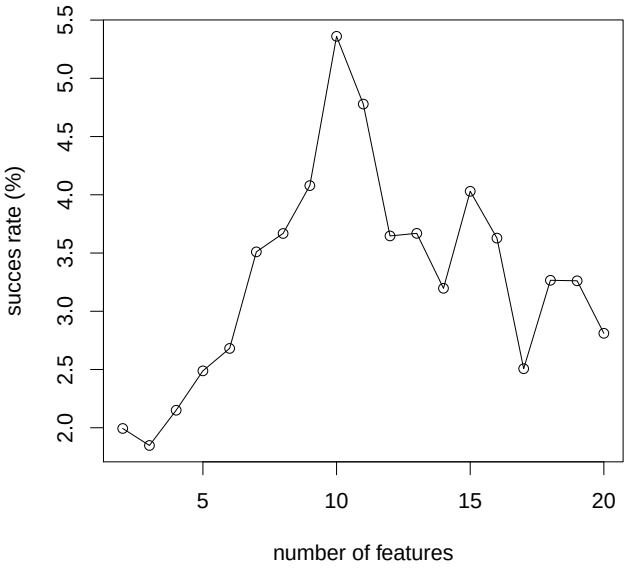

Figure 6 3DES: rate of correct classification vs. number of variables with shrinked TA / PCA.

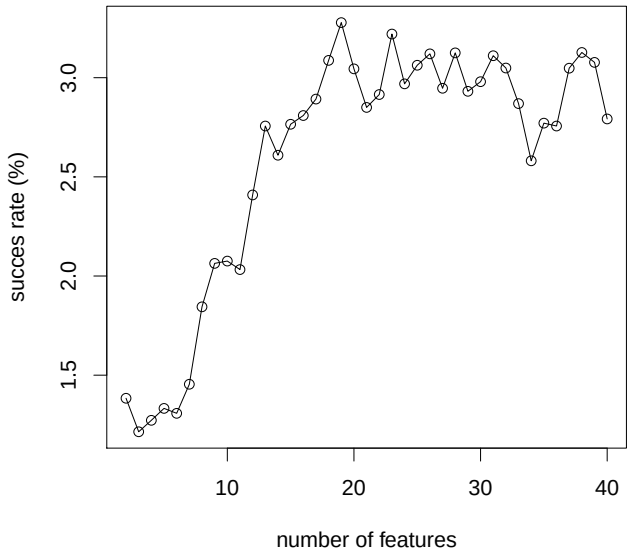

Figure 7 3DES: rate of correct classification of the byte vs. number of variables with shrinked TA / SOST.

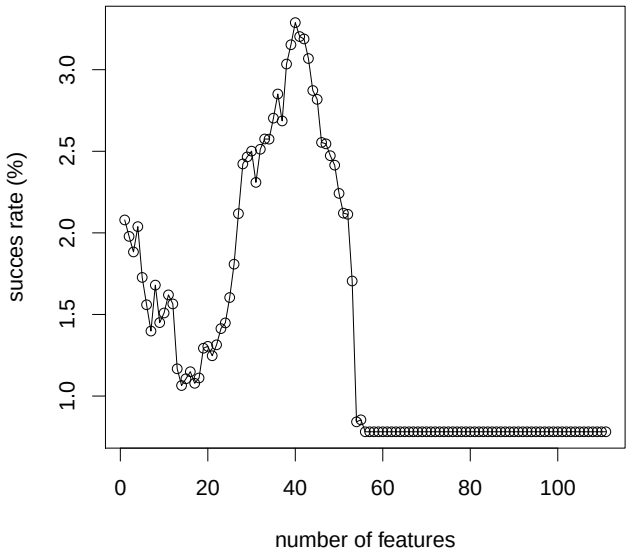

Figure 8 3DES: rate of correct classification of the byte vs. number of variables with shrinked TA/mRMR.

approach [shrinkage estimation (Schafer *et al.*, 2005))] for computing the covariance makes possible the use of a larger number of variables though this has no remarkable effects in terms of accuracy (see Figure 6, Figure 7 and Figure 8).

The rate of correct predictions is indeed below the rate of RF/PCA as indicated by the Table 2 showing the percentage of correct classification in the case of 35 variables with TA/mRMR.

The empirical comparison between TA and machine learning models supports the idea that the normal hypothesis is not necessary. In order to validate these empirical comparisons we performed two classical multivariate normality tests with a significance level of 5%: the Kurtosis (Nordhausen *et al.*, 2008) and the Mardia's test (Mardia, 1970).

In agreement with our results, Mardia?s test and the multivariate normality based on kurtosis rejected the hypothesis of Gaussianity in all multivariate configurations with a number of dimensions ranging between 2 and 40 (selected by mRMR, SOST and PCA). Box plots are available in Appendix A for the Mardia?s test and in Appendix B for the multivariate normality based on kurtosis. Each box plot visualises for all the bits the distribution of p-values for the different dimensions.

Generalization to the other DES bytes

In this section, we generalise the attack discussed so far to all the 24 bytes of the DES key bundle. As this section focuses on other bytes, the results below should not be compared with previous ones. For each byte of the key bundle, $N = 400$ traces are collected for each of the 128 possible instances. After the preprocessing step, the RF/PCA is used to predict the bits. The prediction accuracy results computed by leave-one-out are summarised in Table 3 for the first key (i.e., $B_{(1)}, B_{(2)}, ..., B_{(8)}$), in Table 4 for the second key (i.e., $B_{(9)}, B_{(10)}, ..., B_{(16)}$) and in Table 5 for the last key of 3DES (i.e., $B_{(17)}, B_{(18)}, ..., B_{(24)}$).

As previously mentioned, the dimensionality reduction procedure for RF/PCA selects the optimal number (between 1 and 120) of dimensions on the basis of the product of probabilities of a correct classification.

These results confirm the output of the visualisation phase since on average the last bits of the byte appear to be the most predictable in our high dimensionality context. For instance, the prediction error for $B_{(1)(7)}$ is lower than the one for $B_{(1)(1)}$. Moreover, on average, the number of variables to consider is about 31 with a standard deviation of 17.38.

From prediction to the attack

The prediction results obtained in the previous section encourage the definition of an attack strategy that we will denote as *key search strategy*. The rationale of the strategy is the following: we start by running the RF/PCA model to predict the encryption key. In the case the key is not correctly predicted, we invert the value of the most difficult bit to predict. If the key is still incorrect we proceed by flipping the value of the second most difficult bit and so on.

Let us consider the following example. Suppose we need to predict a key of 8 bits and that our model predicted the value 0011 1101. Suppose that the least

7-th bit	6-th bit	5-th bit	4-th bit	3-rd bit	2-nd bit	1-st bit	entire byte
94.53	*78.13*	*78.13*	*67.19*	*53.13*	*55.47*	*50.78*	5.80

Table 2 Rate of correct classification results in the case of 35 variables with TA/mRMR computed by leave-one-out against a byte of 3DES. The entire byte denotes the probability to predict the entire byte correctly.

	7-th bit	6-th bit	5-th bit	4-th bit	3-rd bit	2-nd bit	1-st bit	Dim	entire byte
1-st byte	*78.13*	*65.63*	*77.34*	*60.16*	*60.16*	*53.13*	*50.00*	61	3.81
2-nd byte	*85.16*	*75.00*	*67.97*	*50.00*	*57.03*	*50.00*	*50.00*	17	3.09
3-rd byte	*78.91*	*67.97*	*70.31*	*69.53*	*67.97*	*50.00*	*51.56*	44	4.59
4-th byte	*85.16*	*73.44*	*60.94*	*57.81*	*50.00*	*50.00*	*54.69*	25	3.01
5-th byte	*89.84*	*78.91*	*65.63*	*60.16*	*64.84*	*52.34*	*50.00*	28	4.75
6-th byte	*82.03*	*73.44*	*60.16*	*59.38*	*50.78*	*54.69*	*60.94*	40	3.64
7-th byte	*69.53*	*67.19*	*61.72*	*50.78*	*54.69*	*50.00*	*50.00*	24	2
8-th byte	*78.91*	*72.66*	*56.25*	*50.00*	*53.91*	*50.00*	*50.00*	39	2.17

Table 3 Rate of correct classification results of RF/PCA computed by leave-one-out for the first key. Dim denotes the number of selected variables while the entire byte denotes the probability to predict the entire byte correctly.

	7-th bit	6-th bit	5-th bit	4-th bit	3-rd bit	2-nd bit	1-st bit	Dim	entire byte
1-st byte	*95.31*	*67.19*	*70.31*	*59.38*	*53.91*	*55.47*	*50.00*	18	4
2-nd byte	*78.13*	*75.00*	*67.19*	*59.38*	*50.00*	*50.00*	*57.03*	51	3.33
3-rd byte	*97.66*	*85.94*	*65.63*	*57.81*	*50.00*	*50.00*	*50.00*	28	3.98
4-th byte	*93.75*	*84.38*	*63.28*	*52.34*	*57.03*	*52.34*	*50.00*	41	3.91
5-th byte	*92.19*	*82.81*	*67.97*	*63.28*	*50.00*	*62.50*	*50.00*	43	5.13
6-th byte	*75.00*	*71.88*	*64.06*	*65.63*	*50.00*	*50.00*	*54.69*	68	3.10
7-th byte	*90.63*	*69.53*	*70.31*	*61.72*	*56.25*	*51.56*	*50.00*	2	3.97
8-th byte	*91.41*	*83.59*	*82.81*	*67.19*	*64.84*	*50.00*	*50.00*	32	6.89

Table 4 Rate of correct classification results of RF/PCA computed by leave-one-out for the second key. Dim denotes the number of selected variables while the entire byte denotes the probability to predict the entire byte correctly.

	7-th bit	6-th bit	5-th bit	4-th bit	3-rd bit	2-nd bit	1-st bit	Dim	entire byte
1-st byte	*89.84*	*74.22*	*62.50*	*54.69*	*60.94*	*50.00*	*54.69*	23	3.80
2-nd byte	*96.09*	*82.81*	*64.06*	*60.16*	*65.63*	*50.00*	*50.00*	31	5.03
3-rd byte	*95.31*	*84.38*	*76.56*	*54.69*	*60.94*	*50.00*	*50.00*	17	5.13
4-th byte	*84.38*	*74.22*	*68.75*	*64.06*	*57.03*	*50.00*	*50.00*	6	3.93
5-th byte	*93.75*	*81.25*	*60.94*	*54.69*	*57.81*	*50.00*	*50.00*	16	3.67
6-th byte	*90.63*	*89.84*	*72.66*	*68.75*	*60.16*	*50.00*	*50.00*	56	6.12
7-th byte	*96.88*	*87.50*	*64.06*	*61.72*	*61.72*	*50.00*	*50.00*	30	5.17
8-th byte	*71.09*	*66.41*	*64.06*	*65.63*	*50.00*	*60.94*	*50.00*	5	3.02

Table 5 Rate of correct classification results of RF/PCA computed by leave-one-out for the third key. Dim denotes the number of selected variables while the entire byte denotes the probability to predict the entire byte correctly.

significant bits are less predictable than the remaining ones. If the model did not return the correct key, we complement the least significant bit. Then we proceed by testing the following keys: 0011 1101, then 0011 110**0**, then 0011 11**1**1, then 0011 11**10**, then 0011 1**0**01, ...

4.2 Experiments on RSA

The aim of this section is to assess the robustness of the machine learning approach by applying it to another encryption scheme: the RSA-512 asymmetric algorithm.

Device under attack

We consider an instance of the RSA-512 algorithm that decrypts a constant message of 256 bits chosen at random and encrypted beforehand. Note that RSA-512 is used here as a decryption algorithm with a private key of 512 bits (64 bytes) though it is also known as a signature algorithm. Note that 512-bit RSA keys are not longer considered secure (Cavallar *et al.*, 2000). Nevertheless, the attack can be generalised to a larger RSA key.

For our purposes, we consider the RSA implementation based on the left-to-right m-ary exponentiation algorithm (Knuth, 1981) where $m = 4$.

As for 3DES, our target value is not the whole 512 bit private key but O_i where $i \in [1; 256]$ (i.e., a byte of the key).

Measurement Setup

Trace measures are performed with the Agilent infiniium DSO8104A 1GHz 4GSa/s oscilloscope. This device allowed to collect traces $T^{(i)}_{(j)}$ of length $n = 5,999$ corresponding to the decryption phase of RSA.

Model selection

This new context led us to perform a new model selection. As for 3DES, we collected a set of 400 traces per key and we used them to select the best model to attack two bytes of the private key. In this case, the model selection step returned the RF/mRMR configuration whose results are shown in Table 6.

Analogously to what was observed during the 3DES experiment, we remark that the initial RSA bits are more difficult to predict than the other ones. This result suggests that the lack of predictability at the bit level could depend on the cryptographic device rather than on the algorithm.

Comparison between Machine Learning and Template Attack

The last part of the RSA experiment compares the accuracy of the RF/mRMR models to the TA. As for the 3DES case, the two types of attacks used the same dataset of 400 traces per key collected by varying the last byte of the key. The comparison was performed on the basis of the success rate estimated by leave-one-out.

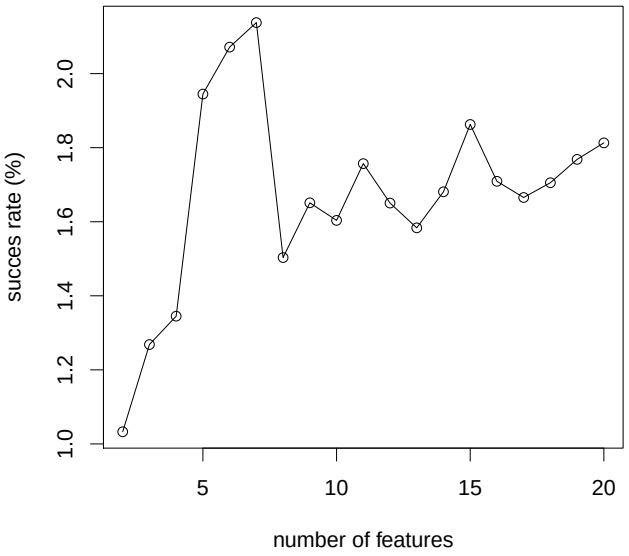

Figure 9 RSA: rate of correct classification vs. number of variables with shrinked TA/mRMR.

As for 3DES, we reduced the number of points for each trace through three feature selection techniques during the training step (i.e., PCA, the mRMR filter and the SOST filter).

The accuracy of the attack as a function of the number of features is reported for TA/mRMR in Figure 9, for TA/PCA in Figure 10 and for TA/SOST in Figure 11. Note that in all case the rates of TA correct predictions are lower than the RF/mRMR rate in this high dimensionality context.

A possible justification of the superiority of the machine learning approach derives from the results of the two normality tests (kurtosis and Mardia) that we carried out on the last byte of RSA. Except one single case, the Mardia?s and the kurtosis tests rejected (pval = 0.05) the parametric hypothesis of normality for all dimensions from 2 to 40. The related box plots of p-value distributions are available in Appendix C for the Mardia?s test and in Appendix D for the kurtosis test.

4.3 Discussion

The experimental results of the previous sections suggest some considerations. The major one concerns accuracy since the experimental results show that for both 3DES and RSA-512, machine learning improves the accuracy of the power analysis attack with respect to conventional TA in high dimensionality settings. In quantitative terms, the use of machine learning increases the probability of recovering a byte of the key from 5.80% to 15.33% in the case of 3DES and from 2.14% to 2.79% in the case of RSA-512. The most probable

	8-th bit	7-th bit	6-th bit	5-th bit	4-th bit	3-rd bit	2-nd bit	1-st bit	Dim	entire byte
last byte	78.13	78.91	83.98	84.77	50	50	50.78	50	16	2.79
32-th byte	71.09	68.75	76.17	76.56	54.30	63.28	56.25	58.59	5	2.23

Table 6 Rate of correct classification results of RF/mRMR computed by leave-one-out for the first key of RSA. Dim denotes the number of selected variables while the entire byte denotes the probability to predict the entire byte correctly.

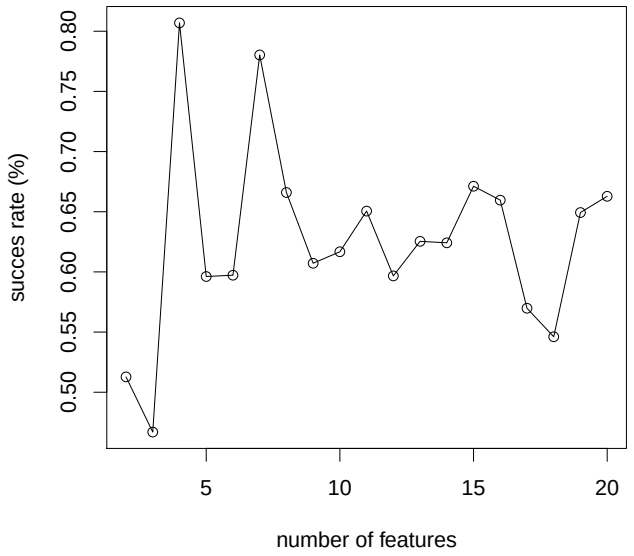

Figure 10 RSA: rate of correct classification vs. number of variables with shrinked TA/PCA.

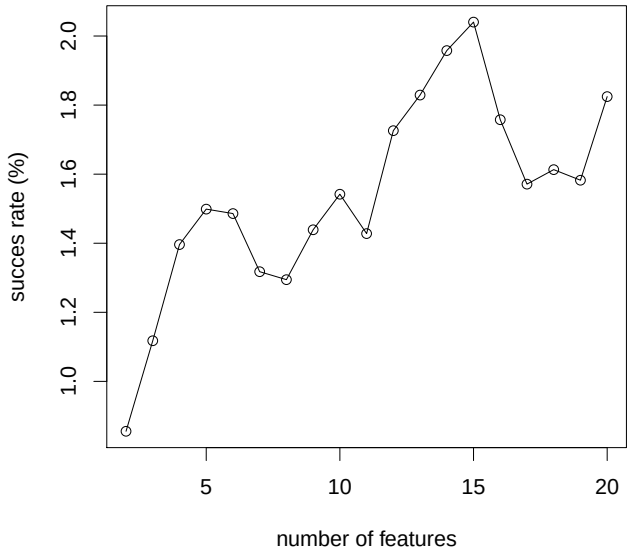

Figure 11 RSA: rate of correct classification vs. number of variables with shrinked TA/SOST.

justification is related to the TA parametric assumption. As our multivariate normality tests showed, the Normal hypothesis is in the majority of cases not supported by the empirical data.

The added value of a machine learning approach can be quantified by the adoption of an alternative accuracy measure: the *guessing entropy* measure. According to (Kopf *et al.*, 2007), "*the guessing entropy of a random variable X is the average number of questions of the kind "does X = x hold" that must be asked to guess X's value correctly*". Mathematically, they define the guessing entropy as:

$$G = \sum_{1 \leq i \leq |\chi|} (iP(x_i)) \tag{18}$$

where χ is the state space of X and $P(x_i) \geq P(x_j)\ \forall i \leq j$. In other words, this term quantifies the difficulty of guessing the value of a key by returning the number of guesses needed on average before finding the right key with the key search strategy.

Suppose that the possible values of the key are sorted with decreasing probability as shown in Section 4.1 ($O_{[1]}$, $O_{[2]}$, ..., $O_{[K]}$) where $O_{[1]}$ denotes the most predictable key. The guessing entropy is defined as:

$$G = \sum_{k=1}^{K} (kP(O_{[k]})) \tag{19}$$

where $P(O_{[k]})$ is the probability the k-th value of the key is the correct one.

If we measure the accuracy of the strategy in terms of guessing entropy, we obtain that in the case of 3DES (resp. RSA) on average the key search strategy needs 11 (resp. 49) tests to recover a byte of the key while the TA requires 21 (resp. 78) of them.

A second interesting conclusion concerns the heterogenous performance in predicting the bits of the key. Our results suggest that this is not caused by the algorithm but rather by the cryptographic device. At this stage, though additional study should be conducted, we could guess that the reason is related to the fact that the device manages each bit of each byte of the key in a similar way.

A third consideration concerns the adoption of feature selection and in particular the fact that not only feature selection improves the accuracy (as in (Archambeau *et al.*, 2006)) but also helps the interpretation. In particular, their use helps understanding which part of the trace is the most informative about the key. For instance, we could be interested in testing whether there is any important information outside the period of encryption or

decryption. The mRMR results suggest that there is a certain amount of information also outside the interval. A possible explanation could be that the private and the secret key are sent unencrypted to the FPGA before each encryption and decryption.

The last remark concerns the adoption of a Gaussian estimator of mutual information in mRMR in site of the fact that data do not follow a Gaussian distribution. Our approach relies on feature selection to reduce the excessive variance due to the huge large number of variables. Now, according to the bias/variance decomposition, the use of a biased estimator (like in regularisation) is recommended in feature selection to reduce variance. This is the reason why the adoption of a Gaussian assumption in mRMR leads anyway to an improvement of the resulting performance.

5 Conclusion

We presented and assessed a machine learning approach, based on non-parametric techniques, able to infer from power consumption observations a model which predicts the bits of a 3DES secret key and the bits of an RSA-512 private key. The availability of an increasing amount of observations about the physical behaviour of a cryptosystem makes machine learning algorithms an important component of an attack strategy.

This paper relies on a large number of experimental comparisons to support the use of a machine learning approach. Some questions remain however unanswered, e.g., why some model configurations perform better than others and if these results may be generalised to other attacks.

About the interpretability issue, it is important to remark that machine learning provides a methodology to train black-box tools in order to predict accurately the keys of the algorithm. Given its black-box nature it is not easy to deduce why an algorithm works better than another. In any case if the interpretation of the model is considered as more valuable than the accuracy of the results then other models and more white-box approaches should be pursued.

About the generalisation ability, we deem that our validation procedure provides an honest estimation of how the prediction algorithms could behave with new data coming from similar problems (in terms of attack algorithms, number of traces and nature of the hardware). At this stage, it is not possible to extrapolate to contexts characterised by different types of signal, dimensionality and noise. We do not claim as a consequence that the proposed learning architecture is the universally best one for SCA tasks since, as formalised by the no-free-lunch theorem, no statistical modelling algorithms can be entitled to be the universally best one. At the same time, we think that our results, based on a large amount of real data, support the idea that non-parametric and dimensionality reduction techniques can be competitive and sometimes better than state-of-the-art approaches when simplistic assumptions do not hold and a high dimensionality context is taken into consideration. It is also worthy to add a word of caution about the generalisation of our results to very large datasets. We considered on purpose a practical side channel attack setting where only a few traces are available. As a consequence, all the results have to be considered in a high dimensionality context and all techniques analysed would perform differently if more traces in the training set were available.

Future work will focus on the generalisation of these preliminary results to other datasets and other classification tasks: first by considering larger portions of the key, second by assessing the impact of the coded message on the prediction accuracy and by varying the cryptographic device. Furthermore, a modified cryptographic algorithm implementation that indexes bits in reverse order will be analysed in order to validate the results of bit leaking order.

Interesting future research perspectives concern the research of alternative values for the parameters of machine learning algorithms, the adoption of multiclass classification (Fürnkranz, 2002) to extend the results of the binary models in side-channel attacks, the adaptation of specific learning techniques for the classification of time series (Caiado, 2010) and the fusion of different measurements as discussed in Agrawal *et al.* (2003).

Acknowledgements

The authors would like to thank Filip Demaertelaere and Cédric Meuter for their support. Furthermore Gianluca Bontempi acknowledges the support of the Communauté Francaise de Belgique (ARC project).

References

E. Alpaydin, (2009), "*Introduction to Machine Learning, second edition edition*", The MIT Press.

D. Agrawal and J.R. Rao and P. Rohatgi, (2003), "*Multi-Channel Attacks*", in the proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2003, LNCS, volume 2779, pages 2-16, Springer.

C. Archambeau and E. Peeters and F.-X. Standaert and J.-J. Quisquater, (2006), "*Template Attacks in Principal Subspaces*", in the proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2006, LNCS, volume 4249, pages 1-14, Springer.

M. Backes and M. Dürmuth and S. Gerling and M. Pinkal and C. Sporleder, (August 11-13, 2010), "*Acoustic side-channel attacks on printers*", in the proceedings of the 19th USENIX Security Symposium, USENIX Association, pages 20-20.

L. Batina and J. Hogenboom and N. Mentens and J. Moelans and J. Vliegen, (2010), "*Side-channel evaluation of FPGA implementations of binary Edwards curves*",

in the International Conference on Electronics, Circuits, and Systems 2010, IEEE, pages 1255-1258.

C.-M. Bishop, (1996), "*Neural Networks for Pattern Recognition*", Oxford University Press, USA, 1 edition.

A. Bogdanov, (2007), "*Improved side-channel collision attacks on AES*", in the proceedings of the 14th international conference on Selected areas in cryptography (SAC) 2007, volume 4876, pages 84-95, Springer-Verlag.

A. Bogdanov and I. Kizhvatov, (2010), "*Beyond the Limits of DPA: Combined Side-Channel Collision Attacks*", IACR Cryptology ePrint Archive 2010: 590.

L. Bohy and M. Neve and D. Samyde and J.-J. Quisquater, (2003), "*Principal and Independent Component Analysis for Crypto-systems with Hardware Unmasked Units*", in the proceedings of e-Smart 2003.

L. Breiman, (2001), "*Random Forests*", Machine Learning, volume 45 n.1, pages 5-32.

J. Caiado, (2010), "*Classification and Clustering of Time Series*", LAP Lambert Academic Publishing.

S. Cavallar and B. Dodson and A. Lenstra and W. Lioen and P. Montgomery and B. Murphy and H. Riele and K. Aardal and J. Gilchrist and G. Guillerm and P. Leyland and J. Marchand and F. Morain and A. Muffett and C. Putnam and C. Putnam and P. Zimmermann, (2000), "*Factorization of a 512-bit RSA modulus*", in the proceedings of the 19th international conference on Theory and application of cryptographic techniques (EUROCRYPT) 2000, LNCS, volume 1807, pages 1-18, Springer.

S. Chari and J.R. Rao and P. Rohatgi, (2002), "*Template Attacks*", in the proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2002, LNCS, volume 2523, pages 51-62. Springer.

J.-S. Coron and D. Naccache and P.C. Kocher, (2004), "*Statistics and Secret Leakage''*, ACM Transactions on Embedded Computing Systems (TECS) 2004, ACM, volume 3, pages 492-508.

C. Cortes and V. Vapnik, (1995), "*Support-Vector Networks*", Machine Learning, volume 20, pages 273-297.

E. Dimitriadou and K. Hornik and F. Leisch and D. Meyer and A. Weingessel, (2011), "*e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*", R package version 1.6, `http://CRAN.R-project.org/package=e1071`.

J. Fürnkranz, (2002), "*Round robin classification*", Journal of Machine Learning Research, volume 2, pages 721-747, JMLR.org.

K. Gandolfi and C. Mourtel and F. Olivier, (2001), "*Electromagnetic analysis: Concrete results*", in the proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2001, volume 2162, pages 251-261, Springer-Verlag.

B. Gierlichs and K. Lemke-Rust and C. Paar, (2006), "*Templates vs. Stochastic Methods*", in the proceedings of the 8th International Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2006, volume 4249, pages 15-29, Springer-Verlag.

B. Gierlichs and L. Batina and P. Tuyls and B. Preneel, (2008), "*Mutual information analysis - a generic side-channel distinguisher*", in the proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2008, volume 5154, pages 426-442. Springer-Verlag.

G. Hospodar and E. De Mulder and B. Gierlichs and I. Verbauwhede and J. Vandewalle, (2011), "*Least Squares Support Vector Machines for Side-Channel Analysis*", in the proceedings of the 2nd Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE), Darmstadt, Germany.

G. Hospodar and B. Gierlichs and E. De Mulder and I. Verbauwhede and J. Vandewalle, (2011), "*Machine learning in side-channel analysis: a first study*", Journal of Cryptographic Engineering, volume 1, issue 4, pages 293-302.

D. E. Knuth, (1981), "*The Art of Computer Programming*", Seminumerical Algorithms, 2nd Edition, Addison-Wesley, volume 2, pages 441-466.

P.C. Kocher, (1996), "*Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*", in the proceedings of the 16th Annual International Cryptology Conference on Advances in Crypto (CRYPTO) 1996, volume 1109, pages 104-113, Springer-Verlag.

P.C. Kocher and J. Jaffe and B. Jun, (1999), "*Differential Power Analysis: Leaking Secrets*", in the proceedings of the 19th Annual International Cryptology Conference on Advances in Crypto (CRYPTO) 1999, LNCS, volume 1666, pages 388-397, Springer-Verlag.

T. Kohonen, (2001), "*Self-Organizing Maps*", Third extended edition, Springer.

B. Kopf and D. Basin, (2007), "*An information-theoretic model for adaptive side-channel attacks*", in the proceedings of the 14th ACM conference on Computer and communications security (CCS), pages 286-296, ACM.

L. Lerman and G. Bontempi and O. Markowitch, (2011a), "*Side Channel Attack: an Approach based on Machine Learning*", in the proceedings of the 2nd Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE), Darmstadt, Germany.

L. Lerman and G. Bontempi and O. Markowitch, (2011b), "*sideChannelAttack: Side Channel Attack.*", R package version 1.1, `http://homepages.ulb.ac.be/~llerman/`.

H. Liu and H. Motoda, (2007), "*Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*", Chapman & Hall/CRC.

S. Mangard and E. Oswald and T. Popp, (2007), "*Power Analysis Attacks: Revealing the Secrets of Smart Cards*", Springer.

K. V. Mardia, (1970), "*Measures of multivariate skewness and kurtosis with applications*", Biometrika, volume 57, issue 3, pages 519-530, Biometrika Trust.

W. Melssen and R. Wehrens and L. Buydens, (2006), "*Supervised Kohonen networks for classification problems*", Chemometrics and Intelligent Laboratory Systems, volume 83, pages 99-113.

K. Nordhausen and H. Oja and D. E. Tyler, (2008), "*Tools for Exploring Multivariate Data: The Package ICS*", Journal of Statistical Software, volume 28, issue 6, pages 1-31.

K. Pearson, (1901), "*On Lines and Planes of Closest Fit to Systems of Points in Space*", Philosophical Magazine, volume 2, issue 6, pages 559-572.

H. Peng and F. Long and C. Ding, (2005), "*Feature Selection based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy*", IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 27, issue 8, pages 1226-1238.

E. Prouff, (2005), "*DPA Attacks and S-Boxes*", in the proceedings of the 12th International Workshop on Fast Software Encryption (FSE) 2005, volume 3557, pages 424-441, Springer-Verlag.

J.-J. Quisquater and D. Samyde, (2001), "*ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards*", in the proceedings of the 2nd International Conference on Research in Smart Cards: Smart Card Programming and Security (E-SMART) 2001, LNCS, volume 2140, pages 200-210, Springer.

M. Renauld and F.-X. Standaert and N. Veyrat-Charvillon, (2009), "*Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA*", in the proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2009, LNCS, volume 5747, pages 97-111, Springer.

R. L. Rivest, (1993), "*Cryptography and Machine learning*", in the proceedings of the Advances in Cryptology (ASIACRYPT) 1991, volume 793, pages 427-439, Springer.

J. Schafer and K. Strimmer, (2005), "*A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics*", Statistical Applications in Genetics and Molecular Biology, volume 4, issue 1, pages 1175.

A. Shamir and E. Tromer, "*Acoustic cryptanalysis: On nosy people and noisy machines*", http://people.csail.mit.edu/tromer/acoustic.

K. Smith and M. Lukowiak, (2010), "*Methodology for Simulated Power Analysis Attacks on AES*", in the proceedings of the Military Communications Conference (MILCOM) 2010, pages 1292-1297, San Jose, CA, USA.

F.-X. Standaert and C. Archambeau, (2008), "*Using Subspace-Based Template Attacks to Compare and Combine Power and Electromagnetic Information Leakages.*", in the proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2008, LNCS, volume 5154, pages 411-425, Springer.

N. Veyrat-Charvillon and B. Gérard and M. Renauld and F.-X. Standaert, (2011), "*An optimal Key Enumeration Algorithm and its Application to Side-Channel Attacks*", Cryptology ePrint Archive, Report 2011/610.

D. H. Wolpert, (1996), "*The Lack of A Priori Distinctions Between Learning Algorithms.*", Neural Computation, volume 8, issue 7, pages 1341-1390.

Appendix A: Mardia's test for 3DES

Each box plot summarises the p-values of Mardia's tests for a specific bit of 3DES by taking into account from 2 to 40 dimensions selected by a feature selection.

In each box plot, the central bar corresponds to the median, the hinges to the first and third quartiles, and the whisker represents the greatest/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.

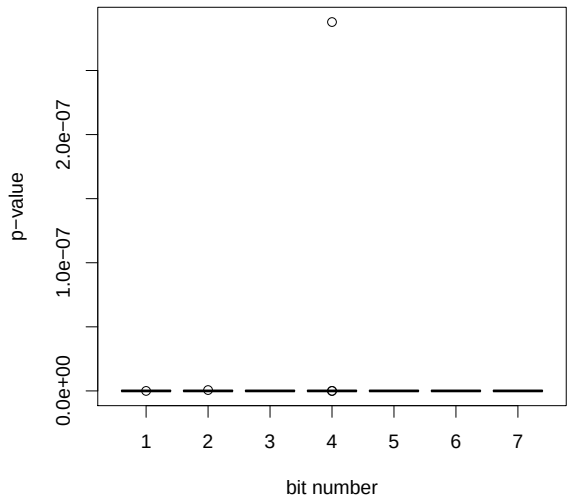

Figure 12 Mardia's test for 3DES/mRMR: p-values vs bit numbers.

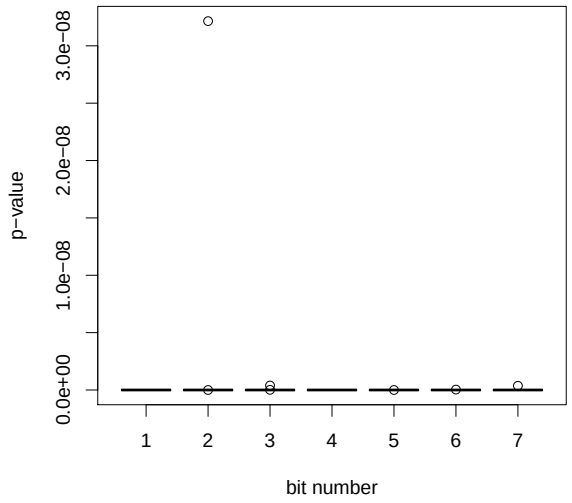

Figure 13 Mardia's test for 3DES/PCA: p-values vs bit numbers.

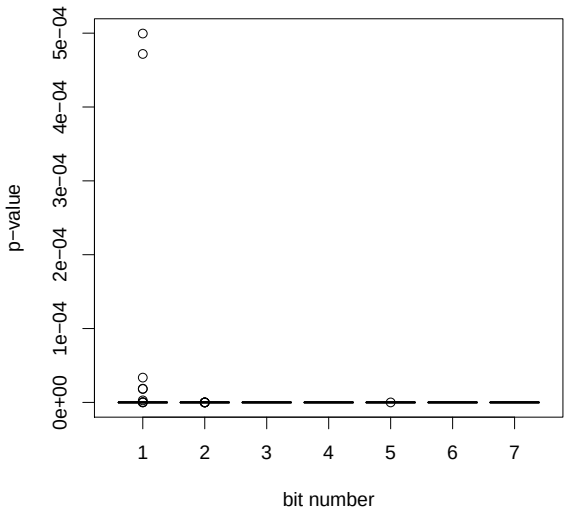

Figure 14 Mardia's test for 3DES/SOST: p-values vs bit numbers.

Appendix B: Multivariate normality test based on Kurtosis for 3DES

Each box plot summarises the p-values of multivariate normality tests based on Kurtosis for a specific bit of 3DES by taking into account from 2 to 40 dimensions selected by a feature selection.

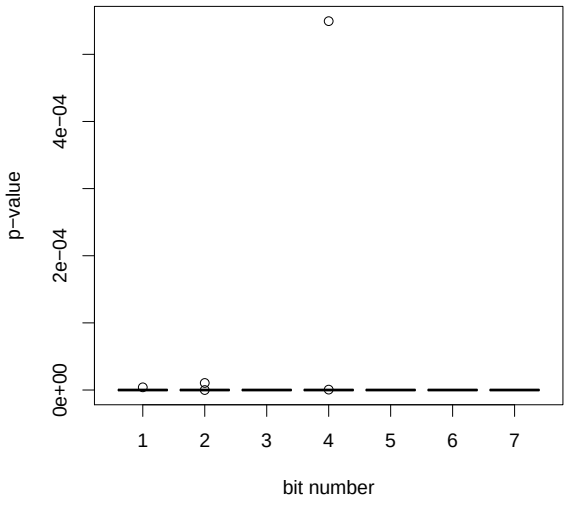

Figure 15 Multivariate normality test based on Kurtosis for 3DES/mRMR: p-values vs bit numbers.

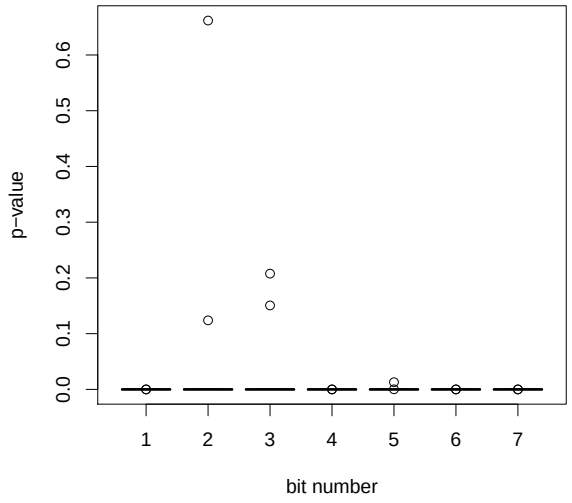

Figure 16 Multivariate normality test based on Kurtosis for 3DES/PCA: p-values vs bit numbers.

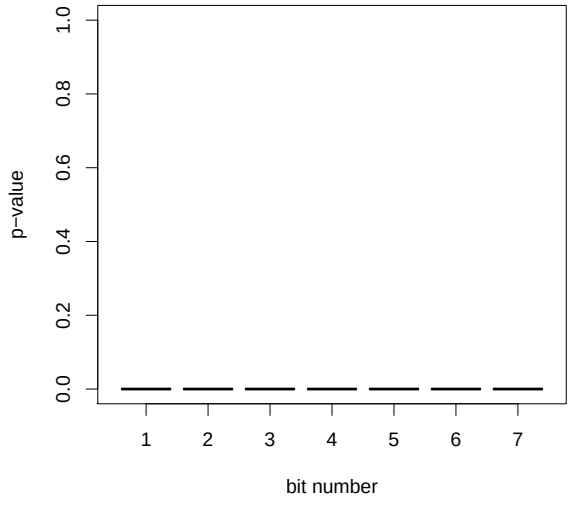

Figure 17 Multivariate normality test based on Kurtosis for 3DES/SOST: p-values vs bit numbers.

Appendix C: Mardia's test for RSA-512

Each box plot summarises the p-values of Mardia's tests for a specific bit of RSA-512 by taking into account from 2 to 40 dimensions selected by a feature selection.

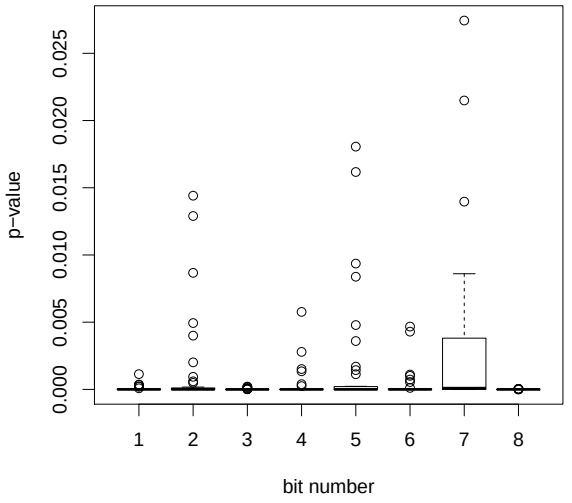

Figure 18 Mardia's test for RSA/mRMR: p-values vs bit numbers.

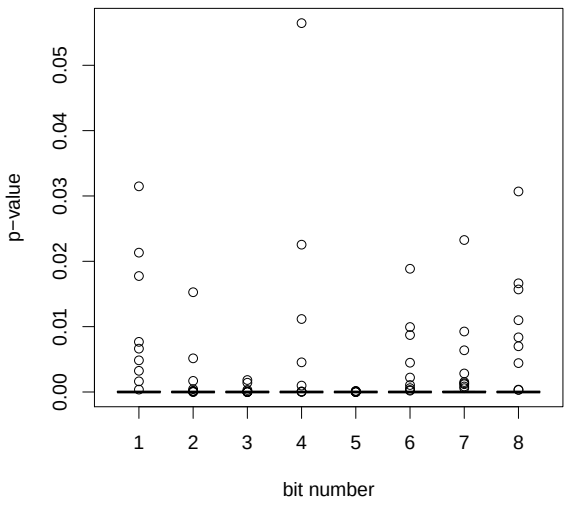

Figure 19 Mardia's test for RSA/PCA: p-values vs bit numbers.

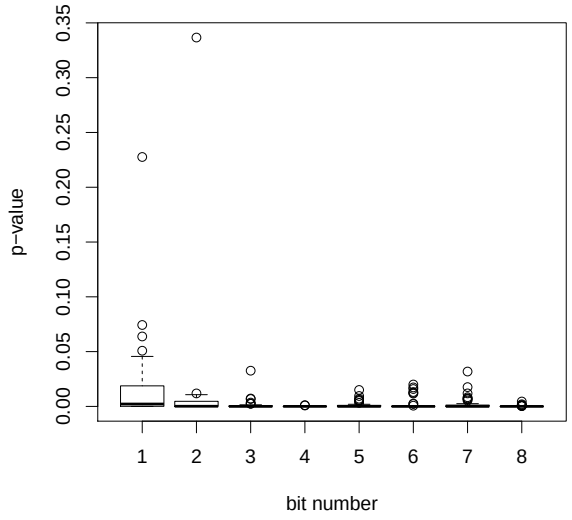

Figure 20 Mardia's test for RSA/SOST: p-values vs bit numbers.

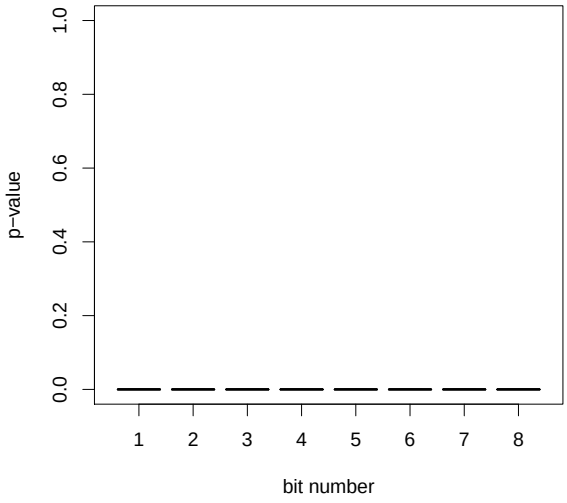

Figure 22 Multivariate normality test based on Kurtosis for RSA/PCA: p-values vs bit numbers.

Appendix D: Multivariate normality test based on Kurtosis for RSA-512

Each box plot summarises the p-values of multivariate normality tests based on Kurtosis for a specific bit of RSA-512 by taking into account from 2 to 40 dimensions selected by a feature selection.

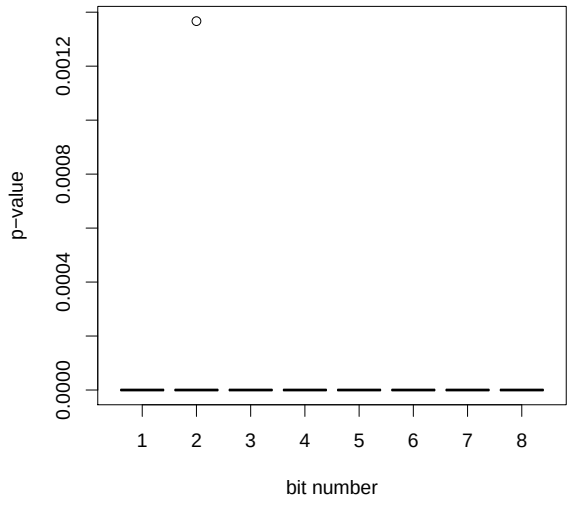

Figure 21 Multivariate normality test based on Kurtosis for RSA/mRMR: p-values vs bit numbers.

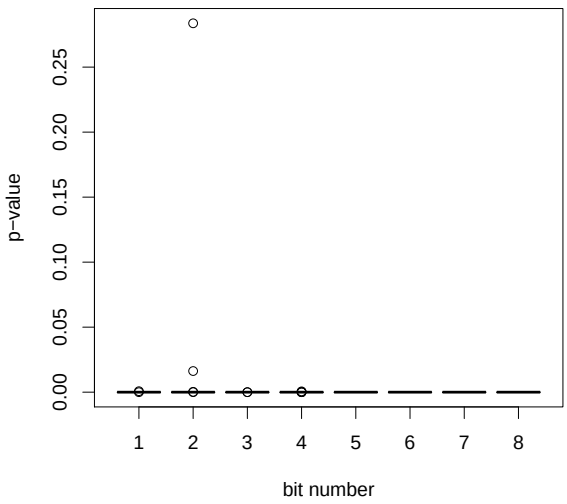

Figure 23 Multivariate normality test based on Kurtosis for RSA/SOST: p-values vs bit numbers.