Side channel attack: an approach based on machine learning

Liran Lerman, Gianluca Bontempi, and Olivier Markowitch

Département d'informatique,
Université Libre de Bruxelles,
Boulevard du Triomphe,
1050 Brussels, Belgium.

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 Template Based DPA have shown their effectiveness in recent years. However these techniques rely on parametric assumptions and are often limited to small dimensionality setting, which limits 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.
For this purpose, we first formalize the problem of studying the relation between power consumption and encryption key as a supervised learning task. Then we compare and assess several classifiers and dimensionality reduction techniques in a real experimental setting. Our promising results regarding the 3DES encryption scheme confirms the importance of adopting machine learning approaches in cryptanalysis.

Keywords: Cryptanalysis, side channel attack, template Based DPA, machine learning.

1 Introduction

Side channel attacks [9] take advantage of the fact that instantaneous power consumption, encryption time [8] or/and electromagnetic leaks [6] of a cryptographic device depend on the processed data and the performed operations. Power analysis attacks are an instance of side-channel attacks which assume that different encryption keys imply different power consumptions. The evolution of the techniques proposed for power analysis attacks along the years has been characterized by an increase of the complexity of the statistical analysis.

Simple Power Analysis (SPA) [9] is the first approach proposed for power attack and it relies on an interpretation of the power consumption in order to deduce information about the used key. In other words, it tries to detect a pattern

in a trace linked to an information about the operation executed. For example, Kocher [8] in 1996 showed that RSA implemented with a square and multiply algorithm allows recovering the key.

Differential Power Analysis (DPA) [9] uses a more advanced statistical analysis than SPA by modeling theoretic power consumption for each key. The likelihood of the observed power consumption for each model is then used to predict the key.

Template Based DPA [4] makes an additional step by estimating the conditional probability of the trace for each key. It extracts all possible informations available in each trace and is hence the strongest form of side channel attack possible in an information theoretic sense. This method relies on a parametric Gaussian estimation approach which, though effective in some cases [11], has several limitations. The method is ill-conditioned if the number of observed traces is smaller than the number of features used to describe the trace. This is the case if the entire sequence of trace measures is taken into consideration.

This paper intends to make one further step in the statistical analysis of power consumption data by taking advantage of machine learning techniques [7]. The role of machine learning in cryptanalysis has already been discussed in [14]. An application of machine learning to cryptanalysis is presented in [2] where a machine learning algorithm is used to find information about the printed characters of a printer by exploiting the information hidden into the acoustic noise.

Here we focus on two aspects of power consumption analysis which has been neglected so far in the literature: 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 TDPA by using techniques of model assessment and selection to find in a nonparametric 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 TDPA by implementing an attack of a byte of the 3DES encryption key which is two times faster.

This paper is organized as follows: Section 2 introduces the notation and reviews the TDPA 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 TDPA approach

Let us consider a crypto device executing a decryption/encryption algorithm with the binary key $O_k, k = 1, \ldots, K$, where $K = 2^B$ is the number of possible values of the key and B is the number of bits (excluding parity bits). Let us observe N times over a time interval of length n the power consumption of

such device and let us denote by *trace* the series of observations $T_i^{(k)} \in \Re^n$, $i = 1, \ldots, N$ associated to the kth key. The state-of-the-art Template Based DPA approaches models the stochastic dependency between the key and a trace by means of a multivariate normal conditional density

$$P(T^{(k)}|O_k; \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_k|}} e^{-\frac{1}{2}(T^{(k)}-\mu_k)\Sigma_k^{-1}(T^{(k)}-\mu_k)^T}$$

where $\mu_k \in \Re^n$ and Σ_k, $k = 1, \ldots, K$, are respectively the expected value and the covariance of the n variate trace associated to the kth key.

Once a set of N traces $T_i^{(k)}, i = 1, \ldots, N$, is observed for each key the TDPA estimates the expected value μ_k and the covariance Σ_k by the sampled mean

$$\hat{\mu}_k = \frac{1}{N} \sum_{i=1}^{N} T_i^{(k)}$$

and the sample covariance

$$\hat{\Sigma}_k = \frac{1}{N} \sum_{i=1}^{N} (T_i^{(k)} - \hat{\mu}_k)^T (T_i^{(k)} - \hat{\mu}_k)$$

An unlabeled trace T is observed when we do not know the key linked to it.

Once such an unlabeled trace T is observed, the techniques returns the key which maximizes the likelihood

$$\hat{k} = \arg\max_k \hat{P}(T|O_k) = P(T|O_k; \hat{\mu}_k, \hat{\Sigma}_k)$$

This approach makes implicitly the assumption that the distribution of the traces for a given key follows a parametric Gaussian distribution, whose number of parameters is equal to $(n^2 + 3n)/2$. Note that the number of parameters can rapidly become very large, and much larger than N, for observation intervals of moderate size (e.g. $n > 20$).

3 Our approach

This paper proposed the adoption of a machine learning approach to estimate from a set of annotated traces the conditional distribution $P(O_k|T)$ where $O_k \in \{0, 1\}^B$. In order to learn this dependency from data we implement a procedure which relies on three steps: decomposition of the prediction task into B separate classification tasks, dimensionality reduction and model selection.

The decomposition of the problem (Figure 1) is dictated by the fact that most of the classification techniques commonly used in machine learning address multi-input single-output problems.

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. We consider four techniques of dimensionality reduction:

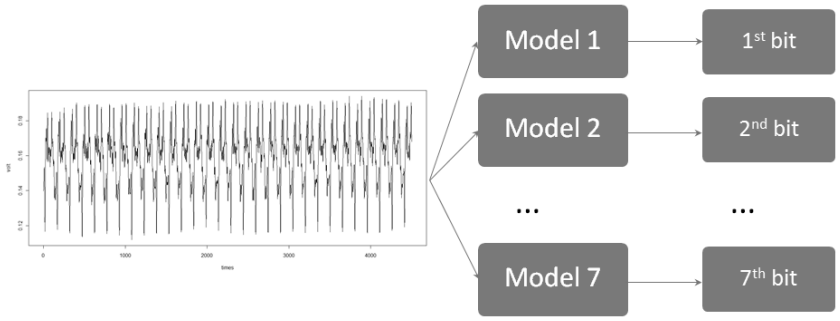


Fig. 1. Decomposition of the prediction problem into a set of classification tasks

- Principal Component Analysis (PCA) [12]: this well-known technique reduces the number n of variables by projecting each trace into a new set of uncorrelated variables.
- Ranking: it is a simple selection technique which returns the highest variant variables
- the minimum Redundancy maximum Relevance (mRMR) filter algorithm [13]: this technique was proposed in bioinformatics to deal efficiently with configurations where the number of variables is larger than the number of samples. It ranks variables by prioritizing the ones which have a low mutual dependence while still providing a large information about the output.
- Self Organizing Map (SOM) [10]: it is a kind of artificial neural network. It associates each trace with a neuron and organizes the network in order to group similar traces.

Model selection is used in order to avoid the parametric assumption made in TDPA and find in a data driven perspective the model which best fits the stochastic dependency between key and power consumption. We considered three different learning machines:

- Self Organizing Map (SOM) [10]: SOM has also been used as a predictive model.
- Support Vector Machine (SVM) [5]: it is one of the most used techniques in classification since it builds efficiently hyperplanes in high dimensional spaces.
- Random Forest (RF) [3]: it relies on the principle of model averaging by building a number decision trees and returning the most consensual prediction.

In order to assess the predictive power of our models, we use the method of leave-one-out. It is executed in N rounds. Each round uses $N-1$ traces to learn a model and the remaining trace to assess the generalization accuracy of the model. This is repeated until every trace has been used for testing purposes. The best one is which maximizes the value returned by leave-one-out.

4 Experiments and discussion

In our experiments we consider the attack of a FPGA device[1], encrypting with a block cipher 3DES algorithm a randomly chosen constant message of 64 bits. Triple DES uses a "key bundle" which includes three DES keys each of 56 bits (excluding parity bits). It represents an encryption DES, followed by a decryption DES followed by another encryption DES. Each one uses a different key. Triple DES takes in input a message of 64 bits and returns ciphertext of the same length through a series of operations. To decrypt, it executes the same operations in reverse. For the sake of simplicity, we restrict to consider attacks of a single byte (e.g. 7 non-parity bits) of the bundle at the time. This means that we will consider a target value O_k which can take $K = 2^7 = 128$ different values. 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.

The experimental procedure is composed of four parts. The first one is a preliminary analysis of the trace distribution which relies on a visualization in a three dimensional space of principal components. The second part is devoted to model selection and aims to select the combination of a learner and feature selector which returns the highest classification accuracy. Such selection is done by considering only the last byte of the first key (B_{18} in Figure 2)[2]. Each time when a byte is attacked, the others are randomly chosen constants. Once the optimal configuration is chosen this one is used for attacking the remaining bytes of the 3DES bundle. The last part of the experimental session considers again the byte B_{18} and assesses the machine learning approach versus the TDPA approach.

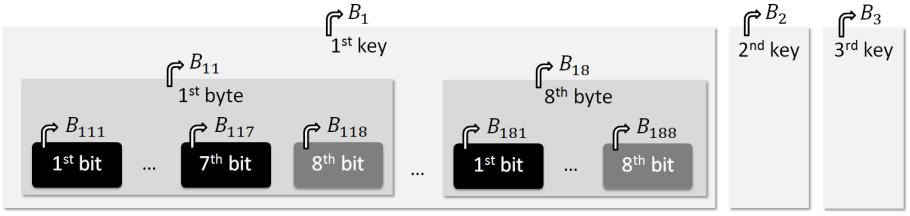


Fig. 2. Representation of the three DES keys.

In the 3D visualization and model selection, for each possible value of the last byte of the first key (B_{18}), $N = 400$ power consumption traces $T_i^{(k)}$ over

[1] Our target device is a Xilinx Spartan XC3s5000 running at a frequency around 33 MHz.

[2] Instead to gather the bits of the key positioned in the same byte, we could group the bits entering in the same S-Box in order to improve our attack. Nevertheless, in order to have a generic attack, we choose not to do so.

a time interval of length 20000 are collected by an oscilloscope Agile infiniium DSO80204B 2Ghz 40GSa/s. Note that the final dimensionality of the trace is set to $n = 9399$ since only the encryption portion of the trace (between 8000th and 17399th times) is considered.

When we attack all the 24 bytes of the DES bundle, for each possible value of the other target bytes, $N = 400$ power consumption traces $T_i^{(k)}$ over a time interval of length 5999 are collected by an oscilloscope Agile infiniium 1GHz 4GSa/s.

4.1 3D visualization

This section focuses on a preliminary visualization of the distribution of the traces associated to the byte B_{18}. Since for each value of the key $N = 400$ power consumption traces are collected, we first filter out the noise by computing for each value O_k of the key the average trace value $\hat{\mu}_k \in \Re^{20000}$ for each key :

$$\hat{\mu}_k = \frac{1}{400} \sum_{i=1}^{400} T_i^{(k)}$$

Then, a preliminary idea of the dependence between $\hat{\mu}_k$ and O_k can be obtained by representing a projection of the n dimensional traces in a tridimensional space. To project the traces, we used the first three PCA components [12]. The seven subfigures of Figure 3 correspond to the seven bits of the B_{18} byte (B_{181}, B_{182}, ..., B_{187}). Points of the same color are linked to traces connected to keys that share the same value for a bit. These visualizations suggest that traces, linked to different values of their lower bits, are less separable and as a consequence, those bits should be more difficult to predict.

4.2 Model selection

This section assesses and compares several classifier configurations by using a leave-one-out approach. As discussed in Section 3 we build a different classifier for each non parity bit of the byte B_{18}. We considered 3 different types of models and 4 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 different types are used:

- SOM(8×5) /: a self organizing map with size 8×5
- SOM(9×5) /: a self organizing map with size 9×5
- SOM(8×6) /: a self organizing map with size 8×6
- SOM(9×6) /: a self organizing map with size 9×6
- SVM / Rank: a support vector machine with feature selection rank
- SVM / Nosel: a support vector machine with feature selection nosel
- RF / Rank: a random forest with feature selection rank
- RF / Nosel: a random forest with feature selection nosel

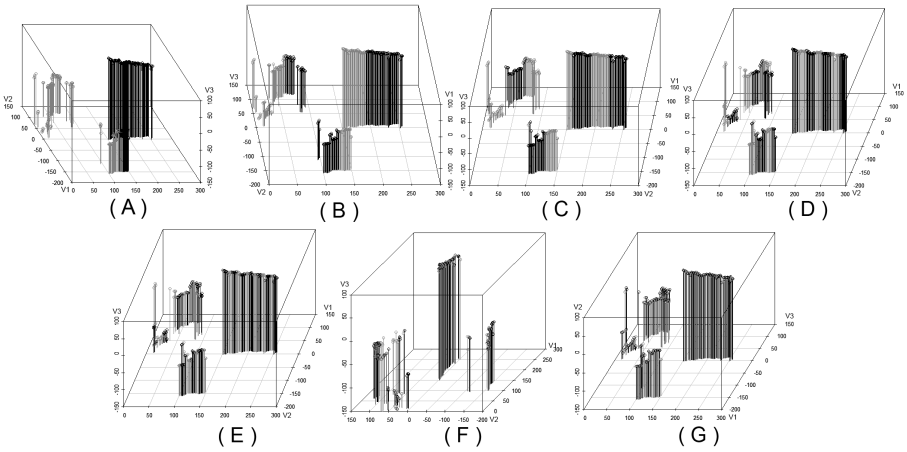


Fig. 3. This figure shows the 128 traces, from the 8th byte of the first key (B_{18}), projected in 3D. The black dots represent a bit value 1 and the others symbolize a bit value 0. The different colors indicate a different value of: the 7th bit (B_{187}) in A, of the 6th bit (B_{186}) in B, of the 5th bit (B_{185}) in C, of the 4th bit (B_{184}) in D, of the 3rd bit (B_{183}) in E, of the 2nd bit (B_{182}) in F and of the 1st bit (B_{181}) in G.

- RF / SOM: a random forest with feature selection self organizing map
- RF / PCA: a random forest with feature selection principal component analysis

The leave-one-out accuracy percentage for different learning configurations and the different bits are reported in Table 1. Note that some bits are predicted with 50%. Those percentages are a rounding to 50% for each percentage lower than 50% estimated by the leave-one-out. The reason of this rounding comes from the fact that the worst estimation of the percentage of good answers is 50%.[3]

It is possible to remark that the most accurate learning configuration is the one made by a PCA algorithm and a Random Forest learner. In the following this learning configuration will be the one used to attack the other bytes of the DES bundle.

4.3 Generalization to the other DES bytes

This section generalizes the attack to all the 24 bytes of the DES bundle. For each byte of the 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 summarized in Table 2.

[3] The idea to invert the value of predicted bit by the model, in order to switch the percentage of good answers, is not relevant. The reason is that a model of machine learning cannot learn the opposite of what is showed to it.

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

Table 1. The leave-one-out accuracy percentage for different learning configurations and the different bits.

	7th bit	6th bit	5th bit	4th bit	3rd bit	2nd bit	1st bit	Dim
1st key								
1st byte	*78.13*	*65.63*	*77.34*	*60.16*	*60.16*	*53.13*	*50.00*	*61*
2nd byte	*85.16*	*75.00*	*67.97*	*50.00*	*57.03*	*50.00*	*50.00*	*17*
3rd byte	*78.91*	*67.97*	*70.31*	*69.53*	*67.97*	*50.00*	*51.56*	*44*
4th byte	*85.16*	*73.44*	*60.94*	*57.81*	*50.00*	*50.00*	*54.69*	*25*
5th byte	*89.84*	*78.91*	*65.63*	*60.16*	*64.84*	*52.34*	*50.00*	*28*
6th byte	*82.03*	*73.44*	*60.16*	*59.38*	*50.78*	*54.69*	*60.94*	*40*
7th byte	*69.53*	*67.19*	*61.72*	*50.78*	*54.69*	*50.00*	*50.00*	*24*
8th byte	*78.91*	*72.66*	*56.25*	*50.00*	*53.91*	*50.00*	*50.00*	*39*
2nd key								
1st byte	*95.31*	*67.19*	*70.31*	*59.38*	*53.91*	*55.47*	*50.00*	*18*
2nd byte	*78.13*	*75.00*	*67.19*	*59.38*	*50.00*	*50.00*	*57.03*	*51*
3rd byte	*97.66*	*85.94*	*65.63*	*57.81*	*50.00*	*50.00*	*50.00*	*28*
4th byte	*93.75*	*84.38*	*63.28*	*52.34*	*57.03*	*52.34*	*50.00*	*41*
5th byte	*92.19*	*82.81*	*67.97*	*63.28*	*50.00*	*62.50*	*50.00*	*43*
6th byte	*75.00*	*71.88*	*64.06*	*65.63*	*50.00*	*50.00*	*54.69*	*68*
7th byte	*90.63*	*69.53*	*70.31*	*61.72*	*56.25*	*51.56*	*50.00*	*2*
8th byte	*91.41*	*83.59*	*82.81*	*67.19*	*64.84*	*50.00*	*50.00*	*32*
3rd key								
1st byte	*89.84*	*74.22*	*62.50*	*54.69*	*60.94*	*50.00*	*54.69*	*23*
2nd byte	*96.09*	*82.81*	*64.06*	*60.16*	*65.63*	*50.00*	*50.00*	*31*
3rd byte	*95.31*	*84.38*	*76.56*	*54.69*	*60.94*	*50.00*	*50.00*	*17*
4th byte	*84.38*	*74.22*	*68.75*	*64.06*	*57.03*	*50.00*	*50.00*	*6*
5th byte	*93.75*	*81.25*	*60.94*	*54.69*	*57.81*	*50.00*	*50.00*	*16*
6th byte	*90.63*	*89.84*	*72.66*	*68.75*	*60.16*	*50.00*	*50.00*	*56*
7th byte	*96.88*	*87.50*	*64.06*	*61.72*	*61.72*	*50.00*	*50.00*	*30*
8th byte	*71.09*	*66.41*	*64.06*	*65.63*	*50.00*	*60.94*	*50.00*	*5*
average	*86.66*	*76.47*	*66.89*	*59.54*	*56.90*	*51.79*	*51.40*	*31.04*
s.d.[4]	*8.44*	*7.39*	*6.09*	*5.71*	*5.71*	*3.45*	*2.88*	*17.38*

Table 2. The prediction acuracy results of RF/PCA computed by leave-one-out. The value of "Dim" is the number of variables to consider.

These results confirm the output of the visualization phase since on average the last bits of the byte appear to be the most predictable. For instance, the prediction error for B_{117} is lower than the one for B_{111}. 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 suggest an attack strategy that we will denote as *enhanced brute force*. 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 predicted correctly, we invert the value of the most difficult bit to predict. If the key is correct the attack is successful otherwise we proceed by flipping the value of the second most difficult bit and so on. As a result we obtain a brute force strategy enhanced by the fact that we take into account the rate of correct predictions for each bit of the key.

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 first bits are less predictable than the remaining ones. Let us assume that the probabilities of predicting correctly each bit are as follows:

8th bit	7th bit	6th bit	5th bit	4th bit	3rd bit	2nd bit	1st bit
90.42	*86.66*	*76.47*	*66.89*	*59.54*	*56.90*	*51.79*	*51.40*

If the model did not return the correct key, we complement the first 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, ...

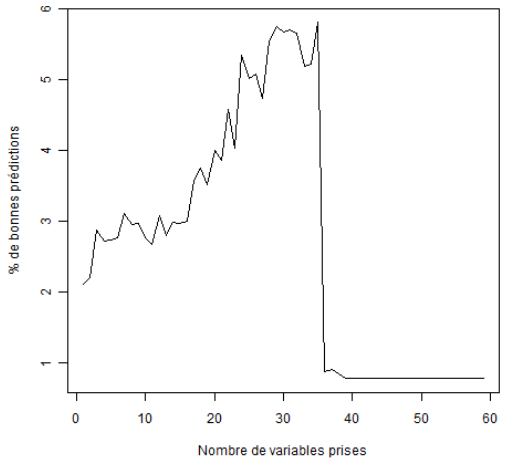
4.4 Comparison between RF / PCA and template Based DPA

The last part of our experimental setting aims to compare the accuracy of the RF / PCA model to the template based DPA. For that reason, we carry out a set of attacks of 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 implementing the 3DES encryption
3. the probes
4. the number of traces
5. the measured traces
6. the computer that performs the calculations of effectiveness of each attack.

We reduce 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 step before implementing the TDPA. Here we used the mRMR filter [13].

The accuracy of the template Based DPA / mRMR attack as a function of the number of features is summarized in Figure 4.



.

Fig. 4. Percentage of correct classification of the byte vs. number of variables.

It is interesting to remark that the technique is not reliable when the number of features goes beyond a certain value. This is due to the ill-conditioning of the covariance matrix when the number of features is too large. The adoption of a regularized approach (shrinkage estimation [15]) for computing the covariance makes possible the use of a larger number of variable though this has no remarkable effects in terms of accuracy (Figure 5).

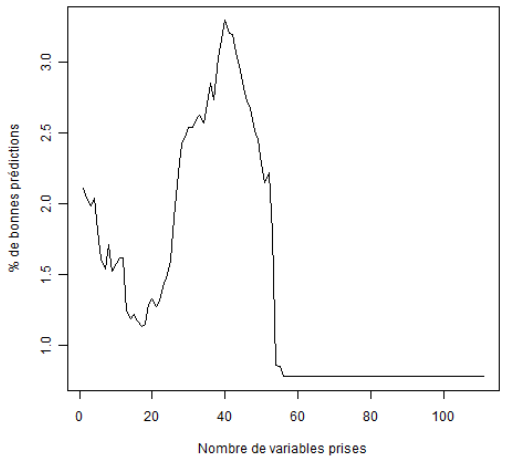
The rate of correct predictions is indeed below the rate of model RF / PCA as shown by the table below showing the percentage of correct classification in the case of 35 variables.

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

4.5 Discussion

The experimental results show that in the case of a FPGA encryption device implementing 3DES, the use of machine learning techniques improves the effectiveness of the power analysis attack with respect to conventional TDPA. In quantitative terms the use of machine learning increases the probability of recovering a byte of the key form 5.80% to 15.33%.

An alternative way of assessing the effectiveness of an attack strategy is to consider the *guessing entropy* measure [16] which quantifies the difficulty of guessing the value of a key. This quantity measures the number of guesses to make on average before finding the right key with the enhanced brute force



.

Fig. 5. Percentage of shrinked TDPA correct classification of the byte vs. number of variables.

strategy. Suppose that the possible values of the key are sorted with decreasing probability as shown in Section 4.3 ($O_{[1]}, O_{[2]}, \ldots, O_{[K]}$) where $O_{[1]}$ denotes the most predictable key. The guessing entropy is defined as:

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

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

If we measure the accuracy of the strategy in terms of guessing entropy, we obtain that on average the enhanced brute force strategy needs 11 tests to recover the key while the template Based DPA / mRMR needs 21 of them.

Another interesting side effect of the adoption of feature selection strategy in our approach is that this helps understanding which part of the trace is the most informative about the key. For instance it could be interesting to know whether there is any important information outside the period of encryption.

The mRMR results suggest that there is a certain amount of information also outside the encryption interval. A possible explanation could be that the encryption key is sent unencrypted to the FPGA before encryption by a co-processor Intel 386.

5 Conclusion

We presented and assessed a machine learning approach able to infer from power consumption observations a model which predicts the bits of a 3DES encryption

key. The availability of an increasing amount of observations about the physical behavior of a cryptosystem makes of machine learning algorithms an important component of an attack strategy.

We can notice that we made empirically comparisons between each model. This is the only kind of comparisons that we were allowed to do because of two facts. First, each model of machine learning used in this paper is as a blackbox who did not allow to know why one of them is the best. Second, template Based DPA was used in a context with many variables. In this context, it does not allowed us to verify if the hypothesis done by the model is correct. In other words, we do not have an algorithm to verify if a set of traces comes from a multivariate Gaussian.

Future work will focus on the generalization of these preliminary results: first by considering larger portions of the key, second by assessing the impact of the coded message on the prediction accuracy, then by varying the cryptographic device and finally by varying the number of measures during learning and validation process.

Interesting future research perspectives are provided also by the adaption of specific learning techniques for the classification of time series and the fusion of different measurements as in [1].

Acknowledgements

The authors would like to thank Atos Worldline to put at the disposal hardware as well as anonymous reviewers for their very useful and valuable advices.

References

[1] D. Agrawal & J.R. Rao & P. Rohatgi, (2003), "*Multi-Channel Attacks*", in the proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2003, LNCS, vol 2779, pp 2-16, Cologne, Germany.

[2] M. Backes & M. Dürmuth & S. Gerling & M. Pinkal & C. Sporleder, (August 11-13, 2010), "*Acoustic side-channel attacks on printers*", to appear in; Proceedings of the 19th USENIX Security Symposium, Washington, DC, USA

[3] L. Breiman, (2001), "*Random Forests*", Machine Learning, v.45 n.1, p.5-32.

[4] S. Chari & J. R. Rao & P. Rohatgi, (2002), "*Template Attacks* ", in CHES, volume 2523 of LNCS, pages 13–28. Springer.

[5] C. Cortes & V. Vapnik, (1995), "*Support-Vector Networks*", Machine Learning, 20.

[6] K. Gandolfi & C. Mourtel & F. Olivier, (2001), "*Electromagnetic analysis: Concrete results*", CHES 2001, C¸ . K. Koc¸, D. Naccache, and C. Paar, Eds., vol. 2162 of LNCS, pp. 255–265, Springer-Verlag.

[7] T. Hastie & R. Tibshirani & J. Friedman, (2009), "*The Elements of Statistical Learning: Data Mining, Inference, and Prediction*", Second Edition, New York: Springer.

[8] P. C. Kocher, (1996), "*Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*", Neal Koblitz, Advances in Cryptology – CRYPTO'96, volume 1109 de Lecture Notes in Computer Science, pages 104–113. Springer-Verlag.

[9] P. C. Kocher & J. Jaffe & B. Jun, (1999), "*Differential Power Analysis: Leaking Secrets*", In Proc. Crypto '99, Springer-Verlag, LNCS 1666, pages 388–397.

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

[11] S. Mangard & E. Oswald & T. Popp, (2007), "*Power Analysis Attacks: Revealing the Secrets of Smart Cards*", Springer, Cambridge, Massachusetts.

[12] K. Pearson, (1901), "*On Lines and Planes of Closest Fit to Systems of Points in Space*", Philosophical Magazine 2 (6), 559–572.

[13] H. Peng & F. Long & 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, Vol 27, No 8, pp 1226-1238.

[14] R. L. Rivest, (1993), "*Cryptography and Machine learning*", Laboratory for Computer Science, Massachusetts Insitute of Technology, Cambridge.

[15] J. Schäfer & K. Strimmer, (2005), "*A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics*", Statistical Applications in Genetics and Molecular Biology, Vol. 4, No. 1, Article 32.

[16] F.-X. Standaert & T. G. Malkin & M. Yung, (2009), "*A Unified Framework for the Analysis of Side-Channel Key Recovery Attacks*", in International Conference on the Theory and Applications of Cryptographic Techniques - Eurocrypt 2009, ser. LNCS, vol. 5479. Springer, pp. 443–461.