A Semi-Parametric Approach for Side-Channel Attacks on Protected RSA Implementations

Guilherme Perin and Łukasz Chmielewski

Riscure BV, Delftechpark 49, 2628 XJ Delft, The Netherlands,
{Perin, Chmielewski}@riscure.com

Abstract. Side-channel attacks on RSA aim at recovering the secret exponent by processing multiple power or electromagnetic traces. The exponent blinding is the main countermeasure which avoids the application of classical forms of side-channel attacks, like SPA, DPA, CPA and template attacks. Horizontal attacks overcome RSA countermeasures by attacking single traces. However, the processing of a single trace is limited by the amount of information and the leakage assessment using labeled samples is not possible due to the exponent blinding countermeasure. In order to overcome these drawbacks, we propose a side-channel attack framework based on a semi-parametric approach that combines the concepts of unsupervised learning, horizontal attacks, maximum likelihood estimation and template attacks in order to recover the exponent bits. Our method is divided in two main parts: learning and attacking phases. The learning phase consists of identifying the class parameters contained in the power traces representing the loop of the exponentiation. We propose a leakage assessment based on unsupervised learning to identify points of interest in a blinded exponentiation. The attacking phase executes a horizontal attack based on clustering algorithms to provide labeled information. Furthermore, it computes confidence probabilities for all exponent bits. These probabilities indicate how much our semi-parametric approach is able to learn about the class parameters from the side-channel information.
To demonstrate the power of our framework we attack the private exponent d_p of the 1024-bit RSA-CRT implementation protected by the SPA, 32-bit message blinding, and 64-bit exponent blinding countermeasures; the implementation runs on a 32-bit STM32F4 microcontroller.

Keywords: RSA, modular exponentiation, side-channel attacks, horizontal attacks, unsupervised learning, clustering algorithms, leakage assessment

1 Introduction

RSA[1]-based cryptosystems are frequently used in credit cards and e-commerce applications. Running on embedded systems, it is a common target of side-channel attacks. The main goal of these attacks is to recover the private key which is directly employed in the modular exponentiation phase – the main

RSA operation. Countermeasures like uniformity of modular operations execution as well as message and exponent blinding render classical side-channel attacks (SPA [5], [10], DPA [6], CPA [7], collision [8],[11], template [15]) unfeasible against RSA implementations. These attacks require a fixed exponent for all measured power (or electromagnetic) traces. The main protection relies on the full randomization of intermediate data, including input message, exponent and modulus, during the execution of an exponentiation [9], [12], [13].

Horizontal attacks [16–24] are emerging forms of side-channel attacks on exponentiation-based algorithms. Their methodology allows recovering the exponent bits through the analysis of individual traces. Therefore, horizontal attacks are efficient against exponent blinding even when combined with message and modulus blinding[1]. A basic requirement of horizontal attacks is the knowledge of the modular exponentiation algorithm. Afterwards, the attacker may choose between different common distinguishers: SPA, horizontal correlation analysis [17], or clustering [22, 23], among others.

Most forms of horizontal attacks require advanced trace preprocessings, characterization and leakage assessment before the application of distinguishers. The main problem of the horizontal attacks is: extracting the leakage from a single trace is limited by noise and unlabeled information. In particular, common leakage assessments, like [30] or t-test [35], require labeled samples and that is not possible due to exponent blinding. Therefore, we have decided to investigate semi-parametric models based on unsupervised learning [31],[33], [34].

The only horizontal solutions, to the above problem to the best of our knowledge are [22, 23]. The first paper [22] proposes to apply a clustering classification to a single trace to allow labeling specific classes operations; this method works well for low noise measurements and requires an EM station composed of multiple probes. The authors of [23] considered a heuristic approach based on difference-of-means for the points of interest selection, which can have large complexity. Furthermore, both solutions use a single trace leakage assessment, which may be affected by a large amount of noise. We discuss horizontal attacks with respect to RSA countermeasures in section 2.1.

Another approach to horizontal attacks is horizontal cross-correlation for elliptic curve scalar multiplication [32][2]. This approach exploits collisions in subsequent additions of a scalar multiplication algorithm using a single trace. Identifying points of interest is not considered in [32].

Contributions. In this paper, we propose a generic framework that aims at solving the aforementioned leakage assessment problem by combining multiple traces even if the device is protected with exponent blinding. The framework

[1] Note that message and modulus blinding affect only the exponentiation input, but not the algorithm itself. Therefore, since horizontal attacks exploit the exponentiation algorithm structure, the aforementioned countermeasures are expected to be ineffective.

[2] Horizontal cross-correlation has not been yet successfully applied to RSA to the best of our knowledge.

assesses the leakage on blinded RSA implementations from multiple traces without access to any labeled information; in particular, it does not require a fully controlled device on which we can reprogram or learn the blinded exponent. Our framework builds and improves on the work from [22, 23] by employing information from multiple traces and by implementing a wide range of leakage assessment methods and statistical classifiers. A direct practical application of our framework is a side-channel ICC EMVCo smart card evaluation [38] since in such an evaluation a fully controlled device is not available.

The framework is divided in two main parts, i.e., learning and attacking phases. This parallel with template attacks comes from the fact that the first phase tries to learn the class parameters from the traces using unsupervised learning and horizontal attacks. During this learning phase, we propose a new leakage assessment based on unsupervised learning which can precisely identify the leakage location by returning the class parameters from mixture of distributions. Points of interest are identified during the leakage assessment and used as the input for a clustering-based horizontal attack [22, 23]. The latter's output is an approximate exponent for each single exponentiation trace. The attacking phase considers the approximate exponent results to re-compute the class parameters and horizontally applies them to the same set of traces using parametric and multivariate attacks. This second phase returns probabilities indicating how much the class parameters are correctly learned from the non-profiled side-channel information.

The presented framework allows the attacker to verify if a protected RSA implementations provides side-channel information even in the presence of blinding countermeasures. The basic assumption is that the device leaks some partial SPA information. We demonstrate that although in attacking phase the horizontal attack is performed on a single trace, we can use clustering algorithms in the learning phase to process multiple traces (which represent randomized exponentiations) to precisely identify the leakage location.

We present the effectiveness of our framework by attacking the private exponent d_p of 1024-bit RSA-CRT implementation protected by SPA countermeasures (like regularity between squares and multiplications), message blinding and exponent blinding. For the sake of simplicity we implement the square-and-multiply exponentiation[3]. The implementation is run on a 32-bit STM32F4 microcontroller. We apply both parts of our framework consecutively and we achieve error rate of 1.27%; it means that for a 512-bit d_p, randomized with a 64-bit value, our framework commits approximately 11 errors.

The above result (11 errors) implies that a brute-force attack is feasible to recover the correct exponent assuming that an attacker can determining the locations of possible errors. Our frameworks outputs not only the recovered exponent bits but also confidence probabilities of the correct guesses (where

[3] Observe that our framework can be also used to attack another exponentiation algorithms, square-and-multiply always [12], for instance. In this case, however, the framework needs to be applied to the whole exponentiation iteration at once and not to single modular multiplications.

0.5 denotes a random guess, for example). These probabilities can be used as a reference for the exponent bits selection with respect to a brute-force attack. In our experiment there are less than 20 bit with the probabilities between 0.45 and 0.55. A known-key analysis, for this specific case, confirmed these 20 bits contain all 11 errors. Furthermore, brute-forcing 20 bits of the RSA key is practical since it can take up to a few hours on a modern PC. The details of the error correction are presented in section 4.2.

Organization of the paper. This paper is organized as follows. We briefly describe preliminaries, in particular we introduce horizontal attacks on exponentiations and unsupervised learning in side-channel attacks, in section 2. Section 3 presents the device under test and our measurement setup. Subsequently, the attack framework is presented in section 4. Finally, section 5 summarizes our contribution, presents future work, and concludes the paper.

2 Preliminaries

Notations. let x be the realization of a random variable X. A sample from X is denoted by x_i. The term $p(X)$ defines the probability mass function when X is discrete and $p(X|Y)$ is the conditional probability of X given Y. Given a set of class parameters θ defining a univariate normal distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2. Here, $g(x|\theta)$ is the function of x defined up to the parameters θ. The term $\mathcal{L}(\theta|X)$ is the likelihood of parameters θ on the sample X.

Trace Characterization. The n-th measured side-channel trace, which represents the power consumption (or electromagnetic emanation - EM) of a cryptographic device over the time domain, is denoted by the uni-dimensional $(1 \times aL)$ vector $\mathbf{t}^n = \{O_1^n, O_2^n, ..., O_{aL}^n\}$. Here, we consider a trace $\mathbf{t}^n$ as being the side-channel information of a modular exponentiation composed by a fixed number aL of modular operations. The factor a depends on the exponentiation algorithm and L is the bit-length of the RSA private key. The trace $\mathbf{t}^n$ can be described by a set of ℓ-sized sub-vectors:

$$\mathbf{t}^n = \{O_1^n, O_2^n, ..., O_{a.L}^n\} = \left\{(t_{1,1}^n, ..., t_{1,\ell}^n), (t_{2,1}^n, ..., t_{2,\ell}^n), ..., (t_{a.L,1}^n, ..., t_{a.L,\ell}^n)\right\} \tag{1}$$

where $t_{i,j}^n$ is the j-th element of each sub-vector O_i^n. The element $t_{i,j}^n$ can also be viewed as a sample in the time domain from the side-channel trace $\mathbf{t}^n$. The set $\{t_{i,j}^n\}$, $i = 1..aL$, refers to a set of samples where each element $t_{i,j}^n$ is extracted from one modular operation O_i^n for a fixed j (e.g., $\{t_{i,10}^n\}$ contains aL samples, each element $t_{i,10}^n$ is selected from the 10^{th} sample of each sub-trace O_i).

Statistical Model. let us consider a set of side-channel traces $\mathbf{T}$. It represents the power consumption of N modular exponentiation executions. We assume a modular exponentiation trace $\mathbf{t}^n$ as a data set composed by k classes or sets of parameters θ_k. Each class is statistically defined by a univariate normal distribution $\mathcal{N}(\mu_k, \sigma_k^2)$ and the set of aL samples $\left\{t^n_{1:aL,j}\right\}$ (for a fixed j) is drawn from a multivariate normal distribution, or a mixture of Gaussians, $\mathcal{N}_d(\mu, \mathbf{\Sigma})$. Because the statistical model is built from RSA implementations, the classes can be understood as squares and multiplications or even exponent bits zeros or ones.

2.1 Horizontal Side-Channel Attacks on Exponentiations

Exponent blinding [6], [12] is the main RSA countermeasure against side-channel attacks. During decryption or singing, this countermeasure changes the sequence of exponent bits for every execution of the algorithm. Therefore, power (or electromagnetic emanation) traces cannot be aligned and processed together to reduce the noise from a set of measurements. Horizontal attacks [16–24] were proposed as an alternative methodology to extract the private key bits from a single trace.

By applying side-channel attacks in a horizontal setting, the n-th trace $\mathbf{t}^n$ must be split in sub-traces, $O^n_i = \{t_{i,1:\ell}\}$, each one representing a particular iteration from the ring of the exponentiation. The sub-trace O^n_i might be the set of samples representing one modular operation or the processing of one exponent bit. This splitting procedure is illustrated in Fig. 1.

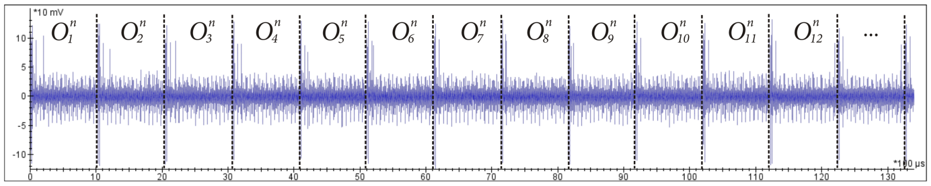

Fig. 1. Modular exponentiation trace.

A basic assumption for the horizontal attacks is the knowledge of the modular exponentiation algorithm (for instance, square-and-multiply always [12], sliding window exponentiation, Montgomery ladder [14], etc.). After, the attacker may choose between different distinguishers: SPA, horizontal correlation analysis [17], Euclidean distance [16], horizontal collision-correlation [18–21] or clustering [22, 23]. The encryption or verification with a public-key can be performed with the same set of instructions used for decryption and signing. In this case, the classes parameters could be learned using the public key and applied as templates on single traces using the private key [24]. However, in this work we assume that this scenario is not possible. Therefore, we decided to investigate semi-parametric models based on unsupervised learning [31], [33], [34] for the horizontal attacks.

2.2 Unsupervised Learning in Side-Channel Attacks

Before entering in the context of unsupervised learning methods for side-channel attacks, it is worth to establish a parallel between supervised and unsupervised learning when attacking exponentiations. Profiled template attacks [15] are a classical example of supervised learning approaches. In this case, a cryptographic device having a fixed or known key is used to learn the statistics (for instance, mean and variance) related to the processing of known exponent bits. Other attacks (e.g., SPA, CPA) that are efficient on non-randomized exponent devices do not consider a learning phase and they are able of breaking the device by statistically processing a set of measurements. Thus, noise contained in the traces can be sufficiently eliminated, enabling the key recovering with the application of specific distinguishers (difference-of-means [6], correlation [7], mutual information [26, 27]).

In a situation when the secret exponent is a random value for each measured trace, and of course, in the absence of a known-key device, multiple traces cannot be conventionally processed together. Thus, as we present in the section 4, unsupervised learning is the methodology that can still provide leakage assessment if the appropriate leakage or statistical model is defined for the set of traces. Clustering methods allow learning the mixture parameters from unlabeled data.

Considering that a set of samples $\{t_{i,j}^n\}$, where $i = 1..aL$ and j is fixed, is characterized by a mixture density:

$$p(t_{i,j}^n) = \sum_{k=1}^{K} p(t_{i,j}^n|\mu_{k,j}^n, \sigma_{k,j}^{2(n)})\mathcal{N}(\mu_{k,j}^n, \sigma_{k,j}^{2(n)}), \tag{2}$$

where $p(t_{i,j}^n|\mu_{k,j}^n, \sigma_{k,j}^{2(n)})$ is the probability density of $t_{i,j}^n$ with respect to the class parameters $(\mu_{k,j}^n, \sigma_{k,j}^{2(n)})$. The number of parameters K should be defined beforehand. Given a set of samples $\left\{t_{i,j}^n\right\}$ extracted from a single exponentiation trace $\mathbf{t}^n$, learning corresponds to estimating the component densities and proportions. We assume the samples obey a parametric model and that we need to estimate their parameters: the mean $\mu_{k,j}^n$ and variance $\sigma_{k,j}^{2(n)}$ for each class k on each sample j.

Let us first define $r_{i,j}^n$ as a parameter defining whether a sample belongs to a specific group or distribution $\mathcal{N}(\theta_{k,j}^n)$. Therefore, $r_{i,j}^n = 1$ if the sample $t_{i,j}^n$ belongs to class k and $r_{i,j}^n = 0$ otherwise. In a supervised setting, $r_{i,j}^n$ is known and we compute the class parameters according to:

$$\mu_{k,j}^n = \frac{\sum_{i=1}^{aL} r_{i,j}^n t_{i,j}^n}{\sum_{i=1}^{a.L} r_{i,j}^n} \qquad \sigma_{k,j}^{2(n)} = \frac{\sum_{i=1}^{aL} r_{i,j}^n (t_{i,j}^n - \mu_{k,j}^n)(t_{i,j}^n - \mu_{k,j}^n)^T}{\sum_{i=1}^{a.L} r_{i,j}^n} \tag{3}$$

In this work, the analysis is unsupervised and the values of $r_{i,j}^n$ are unknown. The **k-means** algorithm [31] can label the data sample by iteratively computing the means for k classes using the distance between samples and means. First, the means $\mu^n_{k,j}$ are randomly initialized by receiving, at random, one element of $\{t^n_{i,j}\}$. The values of $r^n_{i,j} \in \{0,1\}$ are them defined as being:

$$r^n_{i,j} = \begin{cases} 1 & \text{if} \quad |t^n_{i,j} - \mu^n_{k,j}| = \operatorname{argmin}_c |t^n_{i,j} - \mu^n_{c,j}| \\ 0 & \text{otherwise} \end{cases}$$

Following, new means $\mu^n_{k,j}$ are computed using equation 3. The process continues iteratively with the re-definition of $r^n_{i,j}$ and the respective re-computation of means $\mu^n_{k,j}$ till convergence. Usually, the convergence is verified when the current and the previous means are equal.

Fuzzy k-means [31], [33] is an extension of k-means and compute the probability of a sample $t^n_{i,j}$ to belong to a cluster k. Now, the cluster parameter, or probabilities, are $p(t^n_{i,j}|\mu^n_{k,j}) \in [0,1]$. They are initialized at random, with random probabilities inside the interval $[0,1]$ such that $\sum_k p(t^n_{i,j}|\mu^n_{k,j}) = 1$, $\forall i$, and initial means $\mu^n_{k,j}$ are computed as following:

$$\mu^n_{k,j} = \frac{\sum t^n_{i,j}.(p(t^n_{i,j}|\mu^n_{k,j}))^{\frac{2}{\eta-1}}}{\sum (p(t^n_{i,j}|\mu^n_{k,j}))^{\frac{2}{\eta-1}}} \tag{4}$$

where η is free parameter and usually set to 2. Different choices for η depend on the sample set features (e.g., size, noise, etc). The algorithm computes new probabilities with the following equation:

$$p(t^n_{i,j}|\mu^n_{k,j}) = \frac{1/(|t^n_{i,j} - \mu^n_{k,j}|^{\frac{2}{\eta-1}})}{\sum_{c=1}^{K} 1/(|t^n_{i,j} - \mu^n_{c,j}|^{\frac{2}{\eta-1}})} \tag{5}$$

and, respectively new means $\mu^n_{k,j}$. It continues iteratively till convergence, i.e., when the previous and current means remain unchanged. The fuzzy k-means has the cluster centers, or means, and the probabilities as class parameters.

The **Expectation-Maximization (EM) algorithm** [34] is a probabilistic approach which includes the covariance in the set of parameters. Given a Gaussian mixture, the EM algorithm can model the class parameters and maximize the likelihood function with respect to these parameters. Firstly, the means $\mu^n_{k,j}$, covariances $\Sigma^n_{k,j}$ and mixing coefficients $\pi^n_{k,j}$ are initialized. The mixing coefficients must satisfy $0 \leq \pi^n_{k,j} \leq 1$ together with $\sum_{k=1}^{K} \pi^n_{k,j} = 1$. Following, the algorithm alternates between E-step and M-step:

1. E-step: evaluate the responsibilities using the current class parameter values:

$$\lambda^n_{i,k} = \frac{\pi^n_{k,j} exp(-\frac{1}{2}(t^n_{i,j} - \mu^n_{k,j})\Sigma^{-1(n)}_{k,j}(t^n_{i,j} - \mu^n_{k,j})^T)}{\sum_{c=1}^{K} \pi^n_{c,j} exp(-\frac{1}{2}(t^n_{i,j} - \mu^n_{c,j})\Sigma^{-1(n)}_{c,j}(t^n_{i,j} - \mu^n_{c,j})^T)}$$

2. M-step: re-estimate the class parameters using the responsibilities computed in the E-step:

$$\mu^{new(n)}_{k,j} = \frac{1}{L^n_{k,j}} \sum_{i=1}^{a.L} \lambda^n_{i,k} t^n_{i,j} \tag{6}$$

$$\Sigma_{k,j}^{new(n)} = \frac{1}{L_{k,j}} \sum_{i=1}^{aL} \lambda_{i,k}^{n} (t_{i,j}^{n} - \mu_{k,j}^{new(n)})(t_{i,j}^{n} - \mu_{k,j}^{new(n)})^{T} \quad (7)$$

$$\pi_{k,j}^{new} = \frac{L_{k,j}^{n}}{aL} \quad L_{k,j}^{n} = \sum_{i=1}^{aL} \lambda_{i,k}^{n} \quad (8)$$

3. Verify the convergence by verifying the log-likelihood:

$$\mathcal{L}(\theta|\mathbf{t}^{n}) =$$
$$\sum_{i=1}^{aL} \log \left\{ \sum_{c=1}^{K} \pi_{c,j}^{new(n)} exp(-\frac{1}{2}(t_{i,j}^{n} - \mu_{c,j}^{new(n)}) \Sigma_{c,j}^{-1(new)(n)} (t_{i,j}^{n} - \mu_{c,j}^{new(n)})^{T}) \right\} \quad (9)$$

The E and M steps are recursively repeated till a convergence criteria is satisfied.

The three aforementioned clustering methods can be used to model the class parameters from a measured trace $\mathbf{t}^{n}$ and the contained set of sampled $\{t_{i,j}^{n}\}$. Section 4 demonstrates how the unsupervised learning can be used to learn the class parameters (μ, σ, Σ) and identify points of interest for horizontal side-channel attacks on protected exponentiations. First, section 3 gives details about the target implementation and measurement setup.

3 Device Under Test and Measurement Setup

The target under evaluation is a software 1024-bit RSA-CRT implementation. The design runs on a training target for side channel analysis and fault injection testing – piñata board[4]. The board is based on a 32-bit STM32F4 microcontroller with an ARM-based architecture.

Given the RSA private key d and the respective modulus $M = pq$. RSA-CRT algorithm solves the modular exponentiation by using a set private parameters d, p, q, $d_p = d \bmod (p-1)$, $d_q = d \bmod (q-1)$ and $i_q = q^{-1} \bmod p$. This protocol provides a decrypted or signed value $s = CRT(s_p, s_q) = s_q + q(i_q(s_p - s_q) \bmod p)$, where: $s_p = y^{d_p} \bmod p \quad s_q = y^{d_q} \bmod q$.

The modular exponentiation is computed with the left-to-right square-and-multiply algorithm. The implementation features SPA and DPA countermeasures like regularity between squares and multiplications and exponent/message randomization. Algorithm 1 shows the exponentiation method and countermeasures regarding the CRT exponentiation related to prime p. It returns the result s_p. The computation of s_q can be done with the same method.

The random numbers r_1 and r_2 are 32-bit and 64-bit values, respectively. The target of the present evaluation is the exponentiation $y^{d_{p_r} = d_p + r_2(p-1)} \bmod p$. Here, the size of RSA prime p is 512 bits and the random exponent d_{p_r} is, in average, 576 bits. Therefore, we consider L as being 576 and the parameter a as being 1.5. Therefore, the average number of modular operations aL in an exponentiation trace is $1.5 \times 576 = 864$. From this amount, we expect to have, in average, $l_0 = 576$ squares and $l_1 = 288$ multiplications for every execution of

[4] https://www.riscure.com/security-tools/hardware/pinata

Algorithm 1: Protected left-to-right square-and-multiply

```
Data: $y$, $p$, $d_p = (d_{L-1}...d_1 d_0)_2$.
Result: $s_p = y^{d_p} \bmod p$
$y_r \leftarrow y + r_1 p$
$d_{p_r} \leftarrow d_p + r_2(p-1)$
$A \leftarrow 1$
for $i = L-1$ to $0$ do
    $A \leftarrow A^2 \bmod p$
    if $d_{p_r}(i) = 1$ then
        $A \leftarrow A \times y_r \bmod p$
    end
end
Return $A$
```

the exponentiation. Once the randomized exponent d_{p_r} is recovered, the method proposed in [28] can be adopted to recover p.

The measurement setup is composed by the STM32F4 evaluation board, a current probe, an oscilloscope and a power computer to communicate with the equipment and store the acquired traces. The power traces were measured at a sampling frequency of 500MS/sec.

4 Attack Framework

This section describes a new methodology for the application of side-channel attacks on protected RSA implementations. The attack framework is presented in Figure 2. All stages of the framework are described in the remaining part of this section.

The framework combines the concepts of unsupervised learning, horizontal attacks, maximum likelihood estimation and template attacks. The first phase aims at learning the class parameters which are supposed to be protected due to the embedded countermeasures. The second phase follows the principles of maximum likelihood estimation and template attacks. The output is given in terms of probabilities and provides an indicative of how much the class parameters were learned from the protected device.

4.1 Learning Phase

The first phase of our methodology concentrates its efforts in the learning of the class parameters contained in a set of N measured power traces. The analysis starts by assessing the contained leakage in order to identify points of interest. In this work, a point of interest is defined as the sample index $j \in \{1, \ell\}$ where the corresponding sample set $\{t_{i,j}^n\}$, for $i = 1 : aL$, has an observable difference between its class parameters μ_k, σ_k, Σ_k, $k \in \{0, 1\}$ (since the goal is to identify

1. **Learning Phase** executes:
    - (a) **Unsupervised Learning for Leakage Assessment** takes as input a set of traces $\{\mathbf{t}^n\}$, and returns points of interest;
    - (b) **Horizontal Attack** takes as input the points of interest, the set of traces, and for each trace $\mathbf{t}^n$ returns an approximate exponent;
    - (c) **Optimizing the Points of Interest Selection** takes as input the set of traces, the approximate exponents, and executes:
        - i. Using the approximate exponents, determines refined points of interest using T-test;
        - ii. Repeats the **Horizontal Attack** using the refined point of interest and outputs improved approximate exponents.
2. **Attacking Phase** to recover the correct exponent executes:
    - (a) **Computing Final Probabilities** takes as input the set $\{\mathbf{t}^n\}$, the improved approximate exponents, and returns final probabilities; for a modular operation, its final probability is the probability of that operation being a square;
    - (b) **Error Detection and Correction**, for a single trace, it takes as input an improved approximate exponent and the final probabilities; it returns a correct exponent or reports a failure.

Fig. 2. Attack Framework

squares and multiplications, we consider only two possible classes). Therefore, in a single trace $\mathbf{t}^n$, the maximum amount of points of interest is ℓ, which is exactly the amount of samples in a modular operation interval. The question here is: which are the points that leaks confidential information according to a pre-defined leakage model? To identify the points of interest, the leakage model is defined as the samples containing observable difference between squares and multiplications (conditional branches, address-bit). These points of interest are the entry for the horizontal attack that provides an approximate exponent for each trace $\mathbf{t}^n$. Finally, the approximate exponents are used to refine the points of interest selection. The horizontal attack might then be repeated to reduce the error rate in the approximate exponents.

Unsupervised Learning for Leakage Assessment. leakage assessment techniques determine if a cryptographic device is leaking side-channel information according to a specific algorithm and leakage model. Welch t-test is a statistical method which can assess the presence of leakage in cryptographic devices. In the context of public-key algorithms, the document presented in [30] shows a case study of leakage assessment from RSA implementations.

In [29], the authors demonstrate how t-test and mutual information analysis identify the leakage location in time domain. The authors of [27] considered mutual information as a tool to identify the leakage location in the frequency domain and, consequently, the frequency bands in RSA EM traces which contain larger differences between squares and multiplications. Once the leakage is identified, points of interest can be selected for the application of different forms of side-channel attacks with appropriate distinguishers.

An RSA implementation protected with exponent blinding provides random sequences of exponent bits at every execution of the algorithm. The aforementioned application of t-test and mutual information becomes unfeasible since the operations cannot be initially grouped in different and labeled classes. Recent publications addressed the use of unsupervised learning method to exploit the leakage from symmetric and asymmetric cryptographic algorithms [22], [23], [25]. In [25], the authors considered a device with a fixed key, which is not our case. The approaches presented in [22, 23] do not demonstrate how to precisely identify points of interest. The authors of [23] considered that a heuristic approach based on difference-of-means for the points of interest selection, which can be very complex in some cases. [22] proposes to apply a clustering classification to a single trace which allows labeling the operations in specific classes. The method works better for low noise measurements and requires a EM station composed by multiple probes.

To suppress this gap in terms of points of interest location from randomized exponentiations, we demonstrate that we can combine multiple traces for the leakage assessment even if the device is protected with exponent blinding. Additional and optional software countermeasures like message and modulus randomization has no influence against this unsupervised leakage assessment.

As specified in section 3, the terms l_0 and l_1 refer to the amount of squares and multiplications, respectively, in a single trace $\mathbf{t}^n$. The proportion l_0/l_1 is approximately constant for every exponentiation. Due to the exponent blinding countermeasure, the order of squares and multiplications varies from trace to trace. However, the clustering algorithm classifies samples in k groups taking into account the amplitude of these samples. The samples are discrete values represented by an amplitude value, given in volts, and come from the oscilloscope quantization features (resolution, sampling rate). The position of squares and multiplications inside the exponentiation loop is irrelevant for the clustering mechanism. We make the following assumptions:

Assumption 1: At the trace $\mathbf{t}^n$, the mean value for the set of samples $\{t^n_{i,j}\}$ (at sample j) for squares and multiplications are, respectively, $\mu^n_{0,j} + \gamma^n_{0,j}$ and $\mu^n_{1,j} + \gamma^n_{1,j}$, where $\gamma^n_{k,j}$ is the random noise having Gaussian distribution for the class k.

Assumption 2: The means $\mu^n_{k,j}$ are constant values for all traces $\mathbf{t}^n$.

Let us consider the n-th trace represented as a set of samples according to Equation 1. We apply a clustering algorithm to each set of samples $\{t^n_{i:l_0+l_1,j}\}$ at a fixed sampling time j. The clustering returns two centers $c_{0,j}$ and $c_{1,j}$ and two groups of clustered samples $\{g_{0,j}\}$ and $\{g_{1,j}\}$ containing $p_{0,j}$ and $p_{1,j}$ elements, respectively, where $p_{0,j} + p_{1,j} \approx l_0 + l_1$.

Now, for every trace $\mathbf{t}^n$, we have a set of parameters $c^n_{0,j}$, $c^n_{1,j}$, $\{g^n_{0,j}\}$, $\{g^n_{1,j}\}$, $p^n_{0,j}$ and $p^n_{1,j}$ ($j \in \{1,..,\ell\}$) that can be used for the leakage assessment. We provide results for four different techniques: difference-of-means (DoM), sum-of-squared differences (SOSD), sum-of-squared t-values (SOST), and mutual information analysis (MIA). Because, we expect $p^n_{0,j} \approx l_0$ and $p^n_{1,j} \approx l_1$, we define the following ratio parameters:

$$r^n_{k,j} = \begin{cases} p^n_{k,j}/l_k & \text{if} \quad l_k \geq p^n_{k,j} \\ l_k/p^n_{k,j} & \text{if} \quad l_k \leq p^n_{k,j} \end{cases}$$

The ratio parameters $r^n_{k,j}$ are considered as a multiplication factor in the equations for difference-of-means, SOSD and SOST which are given by:

$$\delta_j = \left(\prod_{k=0}^{1} r^n_{k,j}\right)\left(\frac{1}{N}\sum_{n=1}^{N} |c^n_{0,j} - c^n_{1,j}|\right) \quad \text{(DoM)} \tag{10}$$

$$\varsigma_j = \left(\prod_{k=0}^{1} r^n_{k,j}\right)\left(\frac{1}{N}\sum_{n=1}^{N} |c^n_{0,j} - c^n_{1,j}|^2\right) \quad \text{(SOSD)} \tag{11}$$

$$\tau_j = \left(\prod_{k=0}^{1} r^n_{k,j}\right)\left(\frac{1}{N}\sum_{n=1}^{N} \left(\frac{|c^n_{0,j} - c^n_{1,j}|}{\sqrt{\frac{\sigma^{2(n)}_{0,j}}{p^n_{0,j}} + \frac{\sigma^{2(n)}_{1,j}}{p^n_{1,j}}}}\right)^2\right) \quad \text{(SOST)} \tag{12}$$

Considering $\sigma^{2(n)}_j$ as the variance for the sample set $\{g_{0,j}(n)\} \cup \{g_{1,j}(n)\}$, the mutual information value υ_j at each sample index j is computed by deriving the differential entropy [27] for each set of samples according to:

$$\upsilon_j = \sum_{n=1}^{N} \log \sqrt{\frac{1}{p^n_{0,j} + p^n_{1,j} - 1}\sigma^{2(n)}_j} + \sum_{k=0}^{1} (-1)\varrho^k (1-\varrho)^{1-k} \log \sqrt{\frac{1}{(p^n_{1-k,j}) - 1}\sigma^{2(n)}_{1-k,j}} \tag{13}$$

where $\varrho = \frac{p^n_{0,j}}{p^n_{0,j} + p^n_{1,j}}$. The values of υ_j can also be multiplied by the factor $\prod_{k=0}^{1} r^n_{k,j}$ in order to improve the results. Note that the four leakage assessment methods are a sum of individual leakage assessments for each individual trace. The application of clustering algorithms provide an estimation for the mean values $\mu^n_{k,j}$. Due to the aforementioned assumptions on the mean values and due to the summations from equations for SOSD, SOST, DoM and MIA, the noise $\gamma^n_{k,j}$ is eliminated if the number of processed traces is sufficiently large. Figure 3 shows the leakage assessment results for the four aforementioned methods. The time interval represented in this figure is a modular operation interval. The larger peaks in the leakage assessment results indicate the presence of a mixture of distributions with observable difference between its class parameters. A flat line means that the clustered set of samples are drawn from a normal distribution.

Horizontal Attack. The horizontal attack methodology we adopted is based on unsupervised learning approaches [22, 23]. The main goal is to create one approximate exponent for each trace $\mathbf{t}^n$. We particularly adopted the method presented in [23], which is divided in four main steps:

1. Trace preprocessings: an exponentiation trace is organized as a set of samples, following Equation 1.

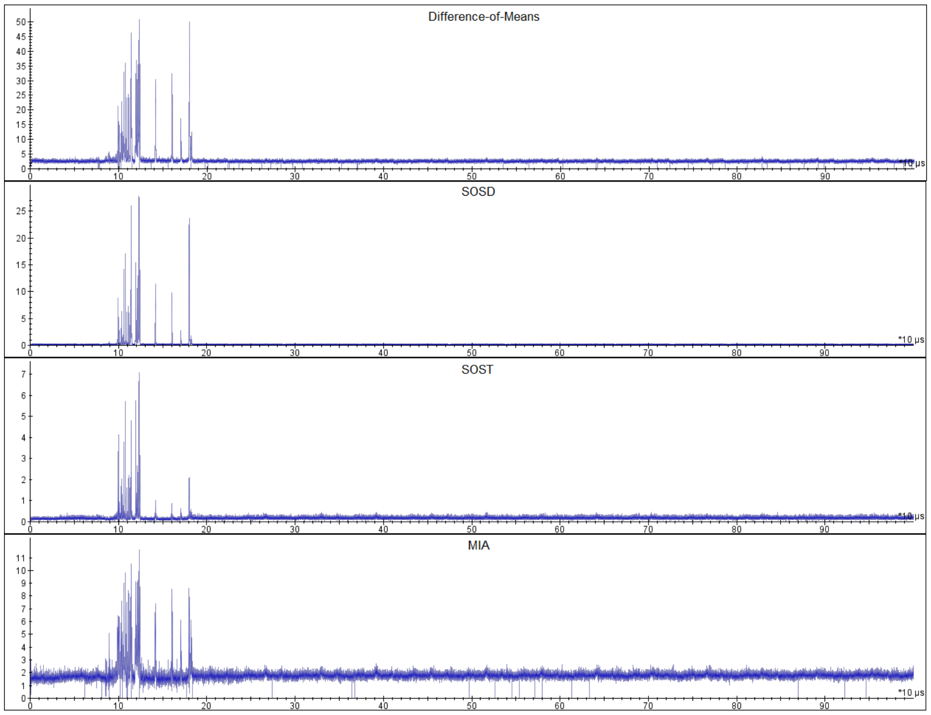

Fig. 3. SOSD, SOST, DoM and MIA leakage assessments.

2. Points of interest selection: for this, we considered the proposed unsupervised learning for leakage assessment, as detailed in the last subsection.
3. Cluster classification of set of samples: the basic idea is to consider a fixed amount of points of interest J_{poi} and apply a clustering algorithm (k-means, fuzzy k-means or EM algorithm) over each set of samples $\{t_{i,j}^n\}$, $i = 1 : aL$ having a fixed $j \in \{1, ..., J_{poi}\}$. A reference point of interest is necessary to associate clusters with classes. The output of this step are exactly J_{poi} clustered sets, grouped into squares and multiplications.
4. Final exponent estimation: the authors of [23] proposed different approaches to provide an approximate exponent for a single trace. Step 3 returns clustered data for J_{poi} sets of samples. It means that J_{poi} candidates for the exponent are given for a single trace. These candidates are combined into one final exponent $\lambda_i^n \in \{0, 1\}$, $i = 1..aL$, using statistical classifiers (majority rule, likelihood estimation and Bayes's estimator).

As described in [23], an optimized selection of points of interest is required by this horizontal attack.

Optimizing the Points of Interest Selection. After one approximate exponents is found for each exponentiation trace $\mathbf{t}^n$, we use these outputs to develop a second leakage assessment step which allows us to refine the points of interest selection.

Now, considering the whole set of N measured exponentiation traces and their respective approximate exponents $\lambda_i^n \in \{0, 1\}$, $i = 1..aL$, new means and variances are obtained:

$$\mu_{0,j} = \sum_{n=1}^{N} \frac{\sum_{i=1}^{aL}(1-\lambda_i^n)t_{i,j}^n}{\sum_{i=1}^{aL}(1-\lambda_i^n)} \qquad \mu_{1,j} = \sum_{n=1}^{N} \frac{\sum_{i=1}^{aL}\lambda_i^n t_{i,j}^n}{\sum_{i=1}^{aL}\lambda_i^n} \tag{14}$$

$$\sigma_{0,j}^2 = \sum_{n=1}^{N} \frac{\sum_{i=1}^{aL}(1-\lambda_i^n)(t_{i,j}^n-\mu_{0,j})^2}{\sum_{i=1}^{aL}(1-\lambda_i^n)} \qquad \sigma_{1,j}^2 = \sum_{n=1}^{N} \frac{\sum_{i=1}^{aL}\lambda_i^n (t_{i,j}^n-\mu_{1,j})^2}{\sum_{i=1}^{aL}\lambda_i^n} \tag{15}$$

Having the class parameters (means and variances) and assuming they are drawn from a mixture of Gaussian distributions, the analysis should infer the difference between these distribution parameters and be able to decide whether they lead to observable dissimilarities.

A t-test calculation is a proper solution for this case. Results presenting important peaks indicate that the classes (e.g., squares and multiplications) have observable difference with respect to their distribution parameters. The new t-values τ_j^{new} are obtained with the following equation:

$$\tau_j^{new} = \frac{|\mu_{0,j}-\mu_{1,j}|}{\sqrt{\frac{\sigma_{0,j}^2}{h_{0,j}}+\frac{\sigma_{1,j}^2}{h_{1,j}}}} \qquad h_{0,j} = \sum_{n=1}^{N}\sum_{i=1}^{aL}(1-\lambda_i^n) \qquad h_{1,j} = \sum_{n=1}^{N}\sum_{i=1}^{aL}\lambda_i^n \tag{16}$$

Figure 4 compares the first and second leakage assessment using t-test. This figure highlights only the part of the modular operation that showed distinct peaks during the first assessment. The optimization step presents more distinct peaks, which are selected as the new points of interest.

After the refinement of points of interest selection, *the horizontal attack is repeated* and supposed to produce new set of approximate exponents $\lambda_i^{new(n)}$ with lower error rates. Table 4.1 presents examples of error rate results for the approximate exponents before and after the points of interest optimization. All the results were obtained from the same trace set and are provided for all possible combinations of clustering algorithms, leakage assessment method and statistical classifier during the horizontal attack. All combinations of techniques provide fairly similar results. However, the decision for specific methods and algorithms depends on the trace and target features.

4.2 Attacking Phase

The second phase, the attacking phase, aims at recovering the full correct blinded exponent from a single trace.

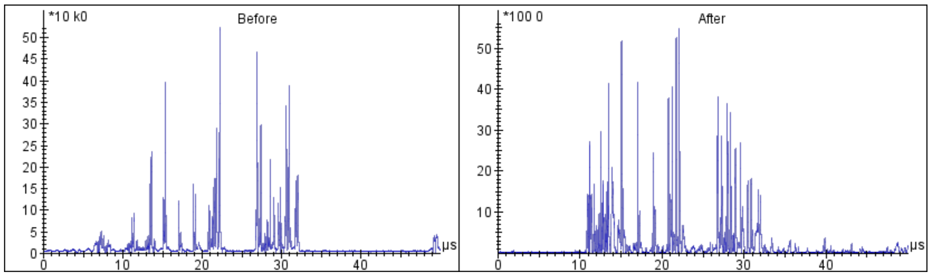

Fig. 4. T-test results before and after the optimized selection of points of interest.

Computing Final Probabilities In this section, we create a metric to estimate how much the class parameters (mean, variance) were correctly learned from the blind RSA implementation. The input for the attacking phase is a set of N traces $\mathbf{t}^n$ labeled with approximate exponent values λ^n. The output of this phase are so-called final probabilities; for each modular operation O_i^n a final probability indicates the probability O_i^n is square. Analogously, this part of the attack framework follows the same methodology of template attacks [15].

Initially, new class parameters (μ_k, σ_k^2, Σ_k, $k \in \{0, 1\}$) are built (they can be seen as templates) from the set of labeled traces. We acquired 10000 power traces from the target device. Therefore, in average $10000 \times 576 = 5.76M$ squares and $10000 \times 288 = 2.88M$ multiplications are used to build templates. Following, we apply parametric and multivariate attack models on the same sets of traces. We compute the likelihood $\mathcal{L}(\theta_k|O_i^n)$ that each modular operation O_i^n belongs to a specific class θ_k. The likelihoods are used to estimate the final probability for each modular operation O_i^n.

A *parametric model* is the univariate model where the class parameters are the mean and the variance. After the determination of the class parameters for squares ($\mu_{0,j}$, $\sigma_{0,j}^2$) and multiplications ($\mu_{1,j,}$, $\sigma_{1,j}^2$) over the set of points of interest $j \in \{1, .., J_{poi}\}$, the log-likelihood of a set of parameters for the operation O_i^n is determined by:

$$\mathcal{L}(\theta_{k,j}|O_i^n) = \mathcal{L}(\mu_{k,j}, \sigma_{k,j}^2|O_i^n) = \sum_{j=1}^{J_{poi}} \log p(t_{i,j}^n|\mu_{k,j}, \sigma_{k,j}^2) \tag{17}$$

where $p(t_{i,j}^n|\mu_{k,j}, \sigma_{k,j}^2)$ is the probability density function based on the Gaussian normal density. The *multivariate model* involves the mean vector $\boldsymbol{\mu}_k = \{\mu_{k,j}\}$, for $j \in \{1, .., J_{poi}\}$, and the covariance Σ_k in the computation of the probability density function. The log-likelihood is given by:

$$\begin{aligned} &\mathcal{L}(\theta_k|O_i^n) = \mathcal{L}(\mu_k, \Sigma_{k,j}|O_i^n) = \\ &\log\left(\frac{1}{\sqrt{2\pi^{J_{poi}}|\Sigma|}} exp(-\frac{1}{2}(O_i^n - \boldsymbol{\mu}_k)\Sigma_k^{-1}(O_i^n - \boldsymbol{\mu}_k)^T)\right) \end{aligned} \tag{18}$$

Clustering Algorithm	First Leakage Assessment Method	Horizontal Attack (Statistical Classifier)	Error rate before optimization	Error rate after optimization
K-MEANS	SOSD	Majority Rule	13.29%	2.35%
		Log Likelihood	13.01%	2.31%
		Bayes Estimation	12.96%	2.27%
	SOST	Majority Rule	11.46%	2.31%
		Log Likelihood	11.52%	1.91%
		Bayes Estimation	12.96%	2.38%
	DoM	Majority Rule	13.06%	2.34%
		Log Likelihood	12.46%	2.36%
		Bayes Estimation	12.90%	2.34%
	MIA	Majority Rule	14.81%	2.77%
		Log Likelihood	14.05%	2.68%
		Bayes Estimation	14.27%	2.71%
FUZZY K-MEANS	SOSD	Majority Rule	10.15%	1.61%
		Log Likelihood	10.28%	1.48%
		Bayes Estimation	11.37%	1.83%
	SOST	Majority Rule	9.45%	2.11%
		Log Likelihood	9.33%	1.27%
		Bayes Estimation	9.35%	1.41%
	DoM	Majority Rule	10.41%	2.12%
		Log Likelihood	10.33%	2.31%
		Bayes Estimation	10.44%	1.91%
	MIA	Majority Rule	10.98%	1.65%
		Log Likelihood	10.77%	1.99%
		Bayes Estimation	10.40%	1.85%
EM ALGORITHM	SOSD	Majority Rule	11.27%	2.23%
		Log Likelihood	12.04%	2.27%
		Bayes Estimation	11.95%	2.19%
	SOST	Majority Rule	10.01%	1.75%
		Log Likelihood	9.88%	1.66%
		Bayes Estimation	9.97%	2.12%
	DoM	Majority Rule	11.98%	2.47%
		Log Likelihood	12.01%	2.43%
		Bayes Estimation	12.15%	2.47%
	MIA	Majority Rule	12.70%	2.33%
		Log Likelihood	12.29%	2.44%
		Bayes Estimation	12.34%	2.37%

Table 1. Error rate before and after the optimized selection of points of interest.

The final probability that an operation O_i^n is a square can be given by:

$$p(O_i^n|\theta_0) = \frac{\mathcal{L}(\theta_0|O_i^n)}{\mathcal{L}(\theta_0|O_i^n) + \mathcal{L}(\theta_1|O_i^n)} \tag{19}$$

Error Detection and Correction. Due to noise and other aspects that interfere the side-channel analysis (misalignment, clock jitter, etc), the derivation of the final exponent for a single exponentiation trace may contain errors. If the amount of wrong bits is sufficiently small, a brute-force attack is still feasible to finally recover the correct exponent. However, the attacker needs a metric to indicate the location of the possible wrong bits in the recovered exponent. The final probabilities can be used as a reference for the exponent bits selection with respect to a brute-force attack.

Note that in Table 4.1, in the column "errors after optimization", the best result, with smallest error rate 1.27%, is achieved for fuzzy k-means, SOST, and Log Likelihood. Furthermore, for any clustering algorithm used with SOST and Log Likelihood, the error rates are less than 2%. We are uncertain why the combination of fuzzy k-means, SOST, and Log Likelihood provides the best result, but we suspect the that the reason is target-specific.

Observe that the 1.27% error rate implies that only $864 \cdot 1.27\% \approx 11$ modular operations are identified incorrectly. In general brute forcing 11 bits is easy, however, the following problem arises: which modular operations are recognized incorrectly? The localization of wrong bits in the recovered exponent is based on calculated likelihood using parametric or multivariate attacks; the probabilities $p(O_i^n|\theta_0)$ are computed for each modular operation. The obtained results indicate less than 20 wrong bits in the exponent if the probabilities interval between 0.45 and 0.55 is considered as the wrong bits interval. A known-key analysis, for this specific case, confirmed that exponent bits with final probabilities between 0.45 and 0.55 contain all wrong guesses. Brute-forcing 20 bits of the RSA key is practical since it can take up to a few hours on a modern PC. Therefore, we confirm that our attack framework recovers the exponent successfully.

5 Conclusions and Future Work

In this work, we presented an attack framework for side-channel attacks on protected RSA implementations. The methodology combines the concepts of unsupervised learning, horizontal attacks, maximum likelihood estimation and template attacks in order to recover the exponent bits.

We proposed an unsupervised learning approach to assess the side-channel leakage even in the presence of exponent and message blinding. We demonstrate that the order of the modular operations inside each exponentiation trace is irrelevant for the leakage analysis; therefore, multiple traces can be processed together in order to find class parameters. If the device leaks SPA-related information, a horizontal attack is able to produce approximate exponents for each exponentiation trace. We demonstrate how these approximate exponents can be used to optimize the leakage location and, finally, refine the determination of approximate exponents. As a final step, the attacker can recover the error bits through the computation of final probabilities for the exponent bits.

To demonstrate the power of our approach we attacked the 1024-bit RSA-CRT implementation protected by the SPA, message blinding, and exponent blinding countermeasures, running on a 32-bit STM32F4 microcontroller. In our experiment the error rate was 1.27%.

As a future work we consider extending our framework to ECC-based cryptographic protocols [3, 4]; we believe that it should be possible due to similarities between ECC and RSA. Another interesting future work is applying our framework in the frequency domain; the goal would be to check whether it is possible to lower the error rate by using side-channel attacks in the frequency domain instead of the time domain. Furthermore, we believe that our framework can be further improved by recovering error bits faster than by using "brute-force", for example, approaches from [36, 37] can be incorporated into the framework.

References

1. R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.
2. N. Koblitz, "Elliptic curve cryptosystems", Mathematics of Computation, vol. 48, pp. 203–209,1987.
3. V. Miller, "Use of elliptic curves in cryptography", Advances in Cryptology-CRYPTO'85, (LCNS 218)[483], pp. 417-426, 1986.
4. P. L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization", Mathematics of Computation, 48(177), pp. 243-264, January, 1987.
5. P.C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", CRYPTO, pp. 104–1113, 1996.
6. P.C. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis", CRYPTO, pp. 388–397, 1999.
7. E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems, CHES'04*, ser. Lecture Notes in Computer Science, vol. 3156. Springer, 2004, pp. 16–29.
8. P.-A. Fouque and F. Valette, "The doubling attack - *why upwards is better than downwards*," in *Cryptographic Hardware and Embedded Systems, CHES'03*, ser. Lecture Notes in Computer Science, vol. 2523. Springer, 2003, pp. 269–280.
9. J.-C. Bajard, L. Imbert, P.-Y. Liardet, and Y. Teglia, "Leak resistant arithmetic," in *Cryptographic Hardware and Embedded Systems, CHES'04*, ser. Lecture Notes in Computer Science, vol. 3156. Springer, 2004, pp. 62–75.
10. J.C. Courrege, B. Feix and M. Roussellet, "Simple Power Analysis on Exponentiation Revisited", in 9th Smart Card Research and Advanced Application Conference (CARDIS), LNCS, vol. 6035, pp. 65-79, 2010.
11. N. Homma, A. Miyamoto, T. Aoki, A. Satoh, and A. Shamir, "Comparative power analysis of modular exponentiation algorithms," *IEEE Transactions on Computers*, vol. 59, no. 6, pp. 795–807, 2010.
12. J.-S. Coron, "Resistance against differential power analysis for elliptic curve cryptography," in *Cryptographic Hardware and Embedded Systems, CHES'99*, ser. Lecture Notes in Computer Science, vol. 1717. Springer, 1999, pp. 292–302.
13. V. Dupaquis and A. Venell, "Redundant Modular Reduction Algorithms", in Smart Card Research and Advanced Applications CARDIS, Lecture Notes in Computer Science, vol. 7079, pp. 102–114, 2011.

14. M. Joye and S.-M. Yen, "The Montgomery powering ladder", in *Cryptographic Hardware and Embedded Systems, CHES'02*, ser. Lecture Notes in Computer Science, vol. 2523. Springer, 2002, pp. 291–302.
15. S. Chari, J.R. Rao and P. Rohatgi, "Template Attacks", *Cryptographic Hardware and Embedded Systems, CHES'02*, ser. Lectur Notes in Computer Science, vol. 2523. Springer, 2002, pp. 13–28.
16. C. Walter, "Sliding Windows Succumbs to Big Mac Attack", in *Cryptographic Hardware and Embedded Systems, CHES'01*, ser. LNCS, vol. 2165, pp. 286–299, Springer, 2001.
17. C. Clavier, B. Feix, G. Gagnerot, M. Roussellet and V. Verneuil, "Horizontal Correlation Analysis on Exponentiation", Proc. ICICS, pp. 46-61, 2010.
18. C. Clavier, B. Feix, G. Gagnerot, M. Roussellet and V. Verneuil, "ROSETTA for Single Trace Analysis", Proc. INDOCRYPT, pp. 140-155, 2012.
19. A. Bauer,E. Jaulmes, E. Prouff and J. Wild, "Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations", Proc. CT-RSA, pp. 1-17, 2013.
20. A. Bauer and E. Jaulmes, "Correlation Analysis against Protected SFM Implementations of RSA", Proc. INDOCRYPT, pp. 98-115, 2013.
21. A. Bauer, E. Jaulmes, E. Prouff and J. Wild, "Horizontal Collision Correlation Attack on Elliptic Curves", Reasearch Gate, 2014.
22. J. Heyszl, A. Ibing, S. Mangard, F. Santis and G. Sigl "Clustering Algorithms for Non-Profiled Single-Execution Attacks on Exponentiations", *IACR Cryptology ePrint Archive*, vol. 2013, pages 438, 2013.
23. G. Perin, L. Imbert, L. Torres and P. Maurine, "Attacking Randomized Exponentiations Using Unsupervised Learning", in Constructive Side-Channel Analysis and Secure Design COSADE, proc. Lecture Notes in Computer Science, vol. 8622, pp. 140-160, 2014.
24. S. Bauer, "Attacking Exponent Blinding in RSA without CRT", Constructive Side-Channel Analysis and Secure Design COSADE, proc. Lecture Notes in Computer Science, vol. 7275, pp. 82–88, 2012.
25. L. Batina, B. Gierlichs and K. L. Rust, "Differential Cluster Analysis", in Cryptographic Hardware and Embedded Systems (CHES), proc. Lecture Notes in Computer Science, vol. 5747, pp. 112–127, 2009.
26. L. Batina, B. Gierlichs, E. Prouff, M. Rivain F. X. Standaert and N. V. Charvillon, "Mutual Information Analysis: a Comprehensive Study", in Journal of Cryptology, vol. 24, number 2, pp. 269–291, 2011.
27. O. Meynard, D. Réal, F. Flament, S. Guilley, N. Homma and J. L. Danger, "Enhancement of simple electro-magnetic attacks by pre-characterization in frequency domain and demodulation techniques", in Design, Automation and Test in Europe (DATE), proc. IEEE, pp. 1004–1009, 2011.
28. J. Krämer, D. Nedospasov and J. P. Seifert, "Weaknesses in Current RSA Signature Schemes", in Information Security and Cryptology (ICISC), proc. Lecture Notes in Computer Science, vol. 7259, pp. 155–168, 2011.
29. L. Mather, E. Oswald J. Bandenburg and M. Wójcik, "Does My Device Leak Information? An a priori Statistical Power Analysis of Leakage Detection Tests", in 19th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), proc. Lecture Notes in Computer Science, vol. 8269, pp. 486–505, 2013.
30. J. Jaffe, P. Rohatgi and M. Witteman, "Efficient side-channel testing for public key algorithms: RSA case study", report, 2011.
31. E. Alpaydin, "Introduction to Machine Learning", 3rd Edition, The MIT Press, London, 2014.

32. N. Hanley, H. Kim, and M. Tunstall, “Exploiting Collisions in Addition Chain-Based Exponentiation Algorithms Using a Single Trace”, Proc. CT-RSA, pp. 431-448, 2015.
33. R.O. Duda, P.E. Hart, D.G. Stork, “Pattern Classification”, 2nd Edition, Wiley-Interscience, 2001.
34. C. M. Bishop, “Pattern Recognition and Machine Learning (Information Science and Statistics)”, Springer, USA, 2007.
35. G. Goodwill, B. Jun, J. Jaffe and P. Rohatgi “A testing methodology for sidechannel resistance validation”, in Non-Invasive Attack Testing Workshop — NIAT 2011.
36. S. Bauer, “Attacking Exponent Blinding in RSA without CRT”, in Constructive Side-Channel Analysis and Secure Design COSADE, proc. Lecture Notes in Computer Science, vol. 7275, pp. 82-88, 2012.
37. N. Heninger and H. Shacham, “Reconstructing RSA Private Keys from Random Key Bits”, in Advances in Cryptology CRYPTO, proc. Lecture Notes in Computer Science, vol. 5677, pp. 1-17, 2009.
38. EMV, “EMVCo Security Evaluation Process”, Security Guidelines, Version 0.5, March 2005.