Attacking Randomized Exponentiations Using Unsupervised Learning

Guilherme Perin[1](✉), Laurent Imbert[1](✉), Lionel Torres[1],
and Philippe Maurine[1,2]

[1] LIRMM/UM2, 161, Rue Ada, 34095 Montpellier, France
{perin,laurent.imbert}@lirmm.fr
[2] CEA-TECH LSAS Laboratory, 880 Avenue de Mimet, 13541 Gardanne, France

Abstract. Countermeasures to defeat most of side-channel attacks on exponentiations are based on randomization of processed data. The exponent and the message blinding are particular techniques to thwart simple, collisions, differential and correlation analyses. Attacks based on a single (trace) execution of exponentiations, like horizontal correlation analysis and profiled template attacks, have shown to be efficient against most of popular countermeasures. In this paper we show how an unsupervised learning can explore the remaining leakages caused by conditional control tests and memory addressing in a RNS-based implementation of the RSA. The device under attack is protected with the exponent blinding and the leak resistant arithmetic. The developed attack combines the leakage of several samples over the segments of the exponentiation in order to recover the entire exponent. We demonstrate how to find the points of interest using trace pre-processing and clustering algorithms. This attack can recover the exponent using a single trace.

Keywords: RSA · Randomized exponentiation · Electromagnetic analysis · Unsupervised learning · Clustering algorithms · Single-execution attacks

1 Introduction

Not only designers of cryptographic devices have to implement the algorithms efficiently, they also have to ensure that sensible information that leaks through several side-channels (time, temperature, power consumption, electromagnetic emanations, etc.) during the execution of an algorithm, remains unexploited by an attacker. If not sufficiently protected, both symmetric and asymmetric cryptographic implementations are vulnerable to these so-called side-channel attacks (SCA). For public-key algorithms such as RSA, the main operation to be armoured consists of a multi-digit exponentiation over a finite ring. In this paper, we present an improved single-execution attack on a randomized implementation of RSA. However, the ideas and tools that we exploit would also apply in the context of CRT-RSA and (hyper)elliptic curves.

E. Prouff (Ed.): COSADE 2014, LNCS 8622, pp. 144–160, 2014.
DOI: 10.1007/978-3-319-10175-0_11

Attacking an exponentiation consists of identifying the bits of the exponent, a value that is often to be kept secret (it is either a secret key or a random secret value). Simple side-channel attacks [2], which uses a single trace of execution, are easily protected using so-called constant-time algorithms such as square-and-multiply-always [4], the Montgomery ladder [7] or atomicity [23].

However, these constant-time algorithms are not sufficient to defeat the more powerful differential [3] and correlation attacks [5]. Although very efficient on not sufficiently protected implementations, these attacks suffer from the very large number of traces to be collected in order to recover (part of) the secret. Collision attacks proposed by Fouque in 2003 [6] are very efficient; they only require two traces of execution on well chosen inputs. All these attacks are generally protected using exponent and/or message blinding using elementary algebraic manipulations. For example, randomization of an RSA exponent relies on the fact that $m^d \equiv m^{d+r\phi(n)} \bmod n$ for any (random) value r (see Sect. 5). Apart from these well known tricks, randomization can also take place at the arithmetic level. The LRA concept [9], based on the Residue Number System, seems to be a robust, yet efficient [24,25] alternative to more expensive (hardware) countermeasures.

Novel attacks [14–17] have recently emerged. Unlike the well studied family of differential [3] and correlation attacks [5], these so-called horizontal correlation attacks (HCA), aim at correlating the Hamming Weight HW(m) of a known message m, with a set of well-chosen sample points t_i from one single trace. Some of them [15,17] are indeed efficient in the presence of message blinding. They exploit the very high regularity of multi-digit exponentiation algorithms and represent a very serious threat against classical randomization countermeasures. A major advantage of single-trace-based attacks is their natural immunity to exponent blinding, since, in many cases, recovering a random exponent is sufficient to break the cryptosystem (see Sect. 5).

Profiled template attacks can recover the exponent using few traces. As the original template attack [11] suggests, the attacker must have full control of the device. In particular, he must be able to send plain-texts of his choice to a known key device. In the case of public-key algorithms, the public-key is known and also can be used in the profiling phase. In this case, the pre-computations whose objective is to build the template set is refereed to as supervised learning. In [13] supervised template attacks are successfully applied on modular exponentiations in order to differentiate squarings from multiplications. More recently, a template attack on constant-time exponentiation algorithms was presented in [12], while [19] suggests a technique to attack the exponent blinding. A template attack targeting the memory addressing was presented in [22]. All those methods fall into the class of supervised attacks, i.e., a learning phase is required during which the adversary constructs templates by exploring the statistical characteristics of various types of operations.

When the adversary does not have a full control of the device, unsupervised methods are necessary. In [18], unsupervised learning has been presented to demonstrate the efficiency of localized EM attacks on exponentiations using a

k-means clustering algorithm to differentiate the attacked samples. Their attack is performed on an ECC [27] implementation over a binary field using Lopez-Dahab coordinates [26]. The scalar is recovered using leakages collected during the execution of a single scalar multiplication ($k \in \mathbb{Z}, P \in E(\mathbb{F}_{2^m}) \longrightarrow [k]P \in E(\mathbb{F}_{2^m})$. However, their attack relies on the ability to acquire several simultaneous EM traces from different probe positions[1]. The leakages obtained from these multi-measurement sources are then combined together in order to reduce the signal-to-noise ratio. By doing so, they managed to classify the sampled points into two distinct sets which correspond to the zero bits (resp. non-zero bits) of the scalar k.

In this paper, we present a single-trace, single-probe unsupervised attack, i.e. the side-channel data is collected from one EM probe only. In the next sections, we present the setting and statistical tools that we used to recover the entire exponent of a constant-time, randomized RSA implementation. Our attack is unsupervised because it does not require any a priori knowledge of the device, in particular we did not use the public key or send any chosen messages in order to learn the characteristics of the device. The chip under attack is a constant-time, RNS-based FPGA implementation of RSA protected with the Leak Resistant Arithmetic [9] and exponent blinding. Since all manipulated data is randomized, we explore the remaining leakages due to control instructions and memory activities. As previously demonstrated in the literature [20], memory and register addresses leak information related to the secret key. Instead of using simultaneous measurements as in [18], we combine the cluster classifications of several samples from each bit of the exponent. We thus process the probabilities obtained from this first phase to recover the entire exponent. Our attack requires four phases: trace pre-processing, points of interest identification, fuzzy k-means clustering, and exponent recovery. For this final phase, we present results obtained with three different statistical techniques (majority rule, normal probability density function and Bayesian classifier).

The paper is organized as follows: Sect. 2 gives details about the randomized exponentiation and the device under attack. The unsupervised learning based on clustering algorithms is detailed in Sect. 3. Section 4 presents the attack in details and the results that we obtained with the three statistical tools mentioned above. Possible countermeasures are suggested in Sect. 6.

2 The Randomized Exponentiation and the Device Under Test

The device under attack is a RNS-based implementation of RSA mapped onto a Spartan-3E xc3s1600 FPGA. For demonstration purposes, we considered a very weak 512-bit RSA. The modular exponentiation is computed with the regular and SPA-protected Montgomery ladder [8] using two sets of RNS bases

[1] Their setting simulates the use of 9 probes uniformly positioned over the chip under attack.

$\mathcal{A}$ and $\mathcal{B}$ [10]. The atomic square-and-multiply [23] is also a regular and faster exponentiation. However as proposed in [15], randomized exponentiations can be explored through horizontal correlation attacks (HCA) if one of the intermediate operands, in the case the randomized input message, is used in several modular multiplications.

According to the leak resistant arithmetic (LRA) concepts [9], the RNS moduli can be randomized before each exponentiation. This countermeasure acts as a message blinding technique because it offers a high degree of randomization to the data. Furthermore, HCA exploits the regularity of long-integer multiplication (or squaring). The parallel RNS arithmetic is then a very limiting factor for this attack. Moreover, our hardware is protected with exponent blinding. Algorithm 1 shows the randomized exponentiation.

Algorithm 1. LRA-RNS Montgomery Powering Ladder [9]

```
Data: $x$ in $\mathcal{A} \cup \mathcal{B}$, where $\mathcal{A} = (a_1, a_2, ..., a_k)$, $\mathcal{B} = (b_1, b_2, ..., b_k)$, $A = \prod_{i=1}^{k} a_i$,
      $B = \prod_{i=1}^{k} b_i$, gcd$(A, B) = 1$, gcd$(B, N) = 1$ and $d = (d_\ell ... d_2 d_1)_2$.
Result: $z = x^d \bmod N$ in $\mathcal{A} \cup \mathcal{B}$
Pre-Computations: $|AB \bmod N|_{\mathcal{A} \cup \mathcal{B}}$
$randomize(\mathcal{A}, \mathcal{B})$
$d_r = d + r.\phi(N)$
$A_0 = MM(1, AB \bmod N, N, \mathcal{A}, \mathcal{B})$    (in $\mathcal{A} \cup \mathcal{B}$)
$A_1 = MM(x, AB \bmod N, N, \mathcal{A}, \mathcal{B})$    (in $\mathcal{A} \cup \mathcal{B}$)
for $i = \ell$ to 1 do
    $A_{\overline{d_{r_i}}} = MM(A_{\overline{d_{r_i}}}, A_{d_{r_i}}, N, \mathcal{B}, \mathcal{A})$    (in $\mathcal{A} \cup \mathcal{B}$)
    $A_{d_{r_i}} = MM(A_{d_{r_i}}, A_{d_{r_i}}, N, \mathcal{B}, \mathcal{A})$    (in $\mathcal{A} \cup \mathcal{B}$)
end
$A_0 = MM(A_0, 1, N, \mathcal{B}, \mathcal{A})$    (in $\mathcal{A} \cup \mathcal{B}$)
```

The operation $MM(x, y, N, \mathcal{B}, \mathcal{A})$ returns $xyB^{-1} \bmod N$ in the two RNS bases $\mathcal{A}$ and $\mathcal{B}$. Both squarings and multiplications are computed with the same number of clock cycles.

First, as the exponent is randomized, single-trace attack was the only option. Further, because the manipulated data is randomized with LRA, the target information of our unsupervised attack is not the data contribution in the EM traces. By data, we mean the intermediate variables which depend on the randomly selected RNS bases and the input message. Exponent-dependent decisions are taken by the architecture's control in order to determine the memory address for reading or writing operands before, during and after the modular multiplications. These conditional tests, as well as the accessed memory addresses, cause subtle leakages of information. These are the only sources of leakages that we exploit in the present unsupervised attack. We present the details of our attack in the next sections.

3 Unsupervised Learning and the Clustering Algorithms

Clustering is one of the most frequently used data mining techniques, which is an unsupervised learning process for partitioning a data set into sub-groups so that the instances within a group are similar to each other and are very dissimilar to the instances of other groups. That is, we shall see what can be done when the collected samples are unlabelled and must be grouped in homogeneous clusters. Two different clustering methods are used in this work: the k-means and the fuzzy k-means algorithms [28].

The k-means algorithm is a geometric procedure for finding c means or centers $(\mu_1, \ldots, \mu_c)$ considering a set of n samples x_j, where $1 \leq j \leq n$. The initialization phase consists in defining the number of clusters c and setting a random sample to each mean μ_i. Thereafter, the algorithm computes the Euclidean distances $ED_{i,j} = \| x_j - \mu_i \|^2$ for all n samples to obtain the maximum-likelihood estimation of the means μ_i. The k-means algorithm is shown in Algorithm 2.

Algorithm 2. K-Means Clustering Algorithm

```
begin initialize $\mathbf{x}, n, c, \mu_1, \ldots, \mu_c$
    do classify $n$ samples $x_j$ according to nearest $\mu_i$ by computing $ED_{i,j}$
        recompute $\mu_i$
    until no change in $\mu_i$
  return $\mu_1, \ldots, \mu_c$
end
```

The k-means algorithm iterates until no changes in μ_i are verified. In all iterations each sample is assumed to be in exactly one cluster. The fuzzy k-means algorithm relaxes this condition and assumes that each sample x_j has some membership with different clusters ω_i, rather than belonging completely to just one cluster.

Initially, the probabilities of cluster membership for each point x_j of a n sample vector $\mathbf{x}$ are normalized according to all clusters ω_i as:

$$\sum_{i=1}^{c} P(\omega_i | x_j) = 1 \tag{1}$$

where $P(\omega_i | x_j)$ is the probability that the sample x_j is in the cluster ω_i. At each iteration of the fuzzy k-means algorithm, the means (or centers) μ_i are recomputed according to the following equation:

$$\mu_j = \frac{\sum_{j=1}^{n} [P(\omega_i | x_j)]^b x_j}{\sum_{j=1}^{n} [P(\omega_i | x_j)]^b} \tag{2}$$

and the new probabilities are recomputed:

$$P(\omega_i | x_j) = \frac{(1/ED_{ij})^{1/(b-1)}}{\sum_{r=1}^{c} (1/ED_{rj})^{1/(b-1)}}, \quad ED_{ij} = \| x_j - \mu_i \|^2 \tag{3}$$

where $b > 1$ is a free parameter chosen to adjust the "blending" of different clusters. Its appropriate choice can improve the cluster classification if the analyzed

Algorithm 3. Fuzzy K-Means Clustering Algorithm

```
begin initialize $n, c, \mu_1, \ldots, \mu_c, P(\omega_i|x_j)$
    normalize probabilities of cluster memberships by Eq. 1
    do classify $n$ samples according to nearest $\mu_i$
        recompute $\mu_i$ by Eq. 2
        recompute $P(\omega_i|x_j)$ by Eq. 3
    until no change in $\mu_i$ and $P(\omega_i|x_j)$
  return $\mu_1, \ldots, \mu_c$
end
```

data set is too much noisy. In this work, this parameter is set to 2. Algorithm 3 shows the fuzzy k-means algorithm.

The next section describes the unsupervised attack in four phases. The k-means algorithm is used in the search for the points of interest. The fuzzy k-means is employed in the cluster classification after having selected the points of interest.

4 The Unsupervised Attack

In a realistic assumption for single-execution attacks on exponentiations, the adversary works in a noisy environment and, as already stated in [19], *"single bits are never known with certainty* [...] *and an SPA attacker* [...] *can only give a probability that any particular operation is a squaring or a multiplication"*, if the attacked device executes the square-and-multiply algorithm. If a single-execution attack is able of recovering 98 % of the 1024 exponent bits and the adversary does not know the wrong bit positions inside the exponent, a brute force attack requires $\sum_{j=0}^{21} C_j^{1024} = 2^{144}$ steps to retrieve the incorrect bits. Therefore, the number of wrong bits in the recovered exponent must be at least very low, otherwise a single-execution attack is impracticable.

When applying non-profiled attacks on a single trace of an exponentiation, the adversary has no knowledge about the operation features (mean μ, variance σ^2). All information must be recovered in a unsupervised manner. Regular binary algorithms [8,23] compute the exponentiation iteratively and for each bit of the exponent (or segment) same operations are performed. Thus, a initial partitioning step is applied to the measured electromagnetic exponentiation trace in order to have ℓ segments, each one representing one exponent bit interpretation. The segments are aligned and compressed to reduce the noise and clock jitter effects. Thereafter, as proposed in this attack, several points of interest are identified in each segment by computing an estimated and approximated difference of means. The cluster classification using the fuzzy k-means algorithm is applied in each set of compressed samples, each set representing a selected point of interest and providing an estimated exponent. The last step consists in retrieving the randomized exponent using all estimated exponents obtained

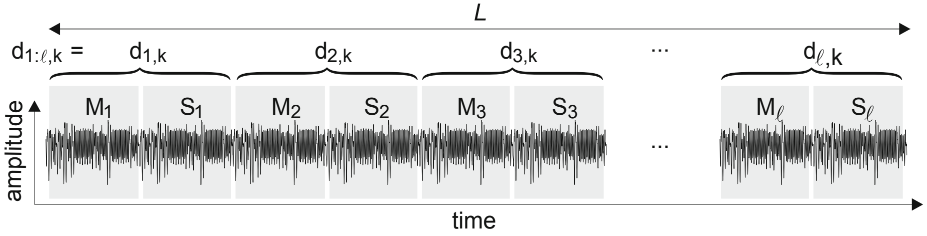

Fig. 1. Exponentiation trace and the segmentation in multiplications and squarings.

with the cluster classification. The proposed attack, divided in four phases, is detailed below.

4.1 Phase 1: Trace Pre-processings

The attack starts by acquiring a single execution exponentiation trace from the device. Let us consider that the randomized exponent is $d_{1:\ell,k}$, where ℓ is the length of the exponent and k is index of the exponentiation trace. In our case, the exponentiation is computed using the regular Montgomery ladder algorithm. The EM trace, with size L, is sliced in ℓ operations of multiplications and ℓ operations of squarings, as depicted in Fig. 1.

Each multiplication (M_i) or squaring (S_i) in the acquired EM trace contains 74 clock cycles. The oscilloscope sampling rate was set to 20 GS/s during the acquisition step and the hardware computes the exponentiation at a clock frequency of 50 MHz, resulting in 59200 samples per exponent bit interpretation ($\mathrm{M}_i\mathrm{S}_i$). The device under attack does not feature any hardware countermeasure, e.g., time disarrangement, dummy cycles or frequency dividers. Therefore, the ℓ segments of multiplication-squarings $\mathrm{M}_i\mathrm{S}_i$, can be easily identified and combined to explore the leakage of information. However, the clock jitter effect is still present in the acquired EM trace and must be suppressed using a trace alignment procedure.

Another important role in unsupervised single-execution attacks is to identify the points of interest which present greater leakages. A simple solution consists in averaging the 400 samples of one clock cycle into 1 sample and taking each averaged sample as a point of interest. Here, in order to preserve the information over smaller windows, the trace is compressed by averaging 100 samples into 1 sample. Then, this allows reducing the amount of data from 59200 to 592 compressed samples during an exponent bit interpretation $d_{i,k}$ (denoted by operation $\langle \mathrm{MS} \rangle_i$ in the sequel).

Now, the ℓ operations are represented by a matrix T:

$$T = \begin{bmatrix} \langle \mathrm{MS} \rangle_1 \\ \langle \mathrm{MS} \rangle_2 \\ \vdots \\ \langle \mathrm{MS} \rangle_\ell \end{bmatrix} = \begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,592} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,592} \\ \vdots & \vdots & \ddots & \vdots \\ t_{\ell,1} & t_{\ell,2} & \cdots & t_{\ell,592} \end{bmatrix} \qquad (4)$$

Each row of the matrix T is a set of compressed samples $\langle \mathrm{MS} \rangle_i = \{t_{i,j}\}$ representing an exponent bit interpretation $d_{i,k}$. The term ℓ is the exponent bit length and, of course, is the iterations number in the Algorithm 1 (steps 6–9). After the trace pre-processing, the attack enters in the second phase that consists in finding the points of interest.

4.2 Phase 2: Finding the Points of Interest

The success of the attack depends on the choice of the points of interest. With profiling or supervised attacks, these points can be found by computing a difference of means and observing the highest peaks of amplitude. In such a case, the adversary has a known key d and computes averaged traces Tr_0 and Tr_1 representing the truncated windows of sampled points when the exponent bit is zero and one, respectively and according to:

$$\mathrm{Tr}_0 = \sum_i \langle \mathrm{MS} \rangle_{d_i=0} \qquad\qquad \mathrm{Tr}_1 = \sum_i \langle \mathrm{MS} \rangle_{d_i=1} \tag{5}$$

Because the presented attack aims at revealing the exponent through an unsupervised manner, the attacker should be considered as having minimal knowledge about the target implementation to identify the points of interest. Because all data are randomized, the remaining leakage is related to addressing and control executions. Therefore, by observing and studying the collected EM trace, the attacker can, for instance, localize the time points where the device performs such operations and discard the points that clearly show no compromising information.

Our unsupervised analysis needs a set of points of interest in each segment $\langle \mathrm{MS} \rangle_i$ to retrieve the exponent. A basic idea is to initially apply a clustering algorithm over each set of compressed samples $\{t_{1:\ell,j}\}$ (each column of matrix T) and find 592 approximated exponents $\widehat{d_{1:\ell,j}}$, for $1 \leq j \leq 592$. In our practical experiments, this leads to the recovery of around 93 % of the entire exponent on the most leaking set of compressed samples $\{t_{1:\ell,j}\}$. It is insufficient. However, this result can be used for calculating approximated and averaged traces $\widehat{\mathrm{Tr}_0}$ and $\widehat{\mathrm{Tr}_1}$ from the approximated exponent $\widehat{d_{1:\ell,j}}$. For this initial step, we considered the k-means clustering algorithm because it is a simple and fast technique. Figure 2(a) shows the relation between the percentage of success recovery of the exponent and the analyzed set of compressed samples $\{t_{1:\ell,j}\}$, for $1 \leq j \leq 592$.

If the adversary selects the most likely exponent (in the case the set $\{t_{1:\ell,465}\}$) he computes the averaged traces $\widehat{\mathrm{Tr}_0}$ and $\widehat{\mathrm{Tr}_1}$. Figure 2(b) shows the approximated difference of mean traces $\widehat{D} = \widehat{\mathrm{Tr}_0} - \widehat{\mathrm{Tr}_1}$. The difference of means $D = \mathrm{Tr}_0 - \mathrm{Tr}_1$, for the real randomized exponent running in the device, is depicted in Fig. 2(c).

Note that the results in Fig. 2(b) and (c) are quite similar and the adversary can select points of interest observing the highest peaks of amplitude in $\widehat{D}$. In a worst case, the adversary would try to compute approximated difference of mean

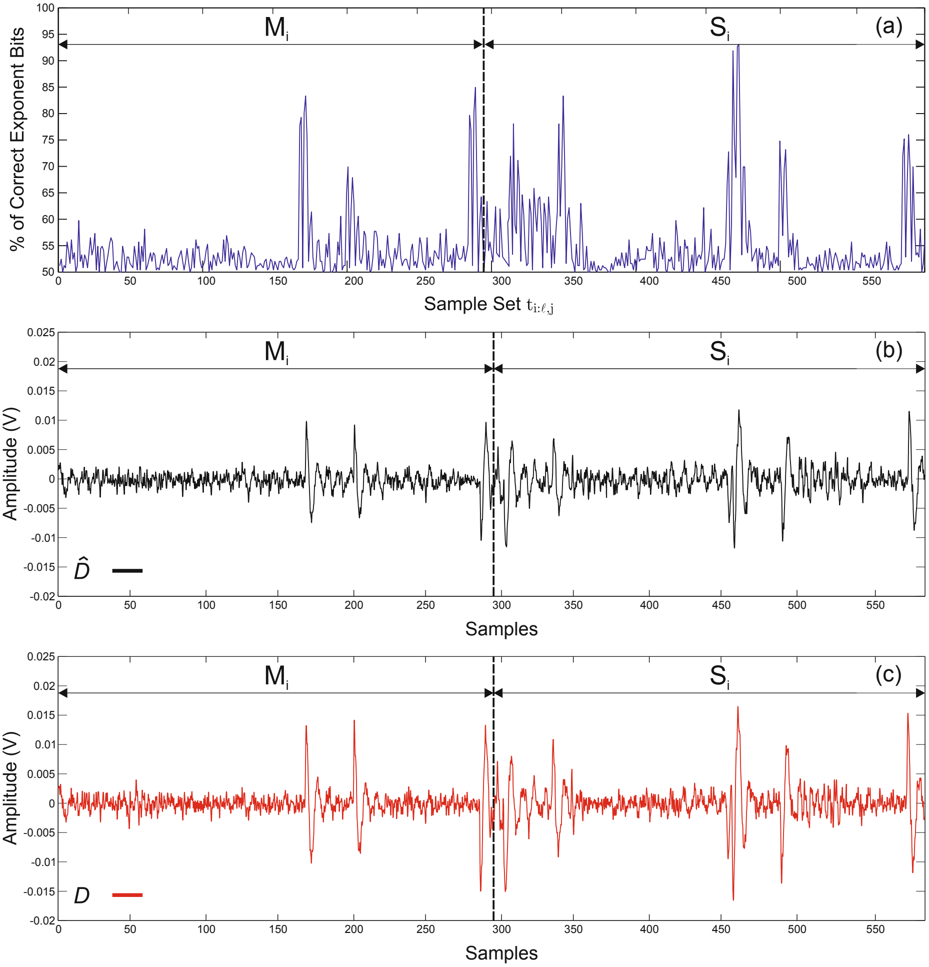

Fig. 2. (a) Percentage of correct exponent bits. (b) Approximated difference of mean traces $\widehat{D} = \widehat{\mathrm{Tr}_0} - \widehat{\mathrm{Tr}_1}$ (c) Difference of mean traces $D = \mathrm{Tr}_0 - \mathrm{Tr}_1$.

traces, and selecting points of interest, from each one of the 592 possibilities. It is clear that the selection of the most leaking point of interest reduces the computational time of the unsupervised attack. Besides, we observed (see Fig. 2) that the highest percentages of correct exponent recovery match with the highest peaks of amplitude in the approximated difference of means $\widehat{D}$. We used this observation as a heuristic in order to select the points of interest.

4.3 Phase 3: Cluster Classification

After computing the approximated difference of mean traces from the set of compressed samples $\{t_{1:\ell,465}\}$, let us suppose the selection of u points of interest

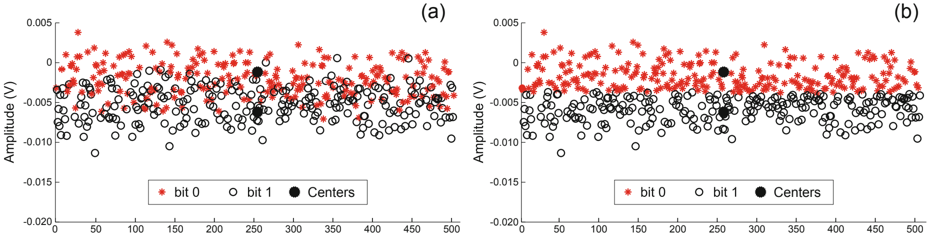

Fig. 3. Errors of cluster classification: (a) Correct classification. (b) Fuzzy k-means classification.

$P = \{p_j\}$, for $1 \leq j \leq u$, among the 592 possibilities. Observing the approximated difference of means $\widehat{D}$ in Fig. 2(b), 17 points of interest were selected ($p_j = 165, 166, 169, 281, 282, 284, 285, 342, 461, 462, 464, 465, 497, 498, 577, 580, 581$), which evidently are the most leaking points.

A clustering is computed for all set of compressed samples $\{t_{1:\ell,p_j}\}$, for $1 \leq j \leq u$ by applying the fuzzy k-means algorithm. Thus, a classification for these samples in two classes (bits zeros and ones), which leads to one estimated exponent $\widehat{d_{1:\ell,p_j}}$ per set of samples $\{t_{1:\ell,p_j}\}$, is obtained. Because real attacks work on noisy environments, the clustering over each point of interest p_j contains errors of classification. Figure 3 illustrates the cluster classification error for the set of compressed samples $\{t_{1:\ell,169}\}$. Figure 3(a) shows the correct classification according to the real randomized exponent $d_{1:\ell,k}$ and the Fig. 3(b) presents the cluster classification returned by the fuzzy k-means algorithm.

For each point of interest p_j, the fuzzy k-means clustering algorithm returns two centers μ_1 and mu_2 and two groups of clustered samples. A common problem would be to identify what class or operation (exponent bit zero or one) each cluster center represents. With $u = 17$ cluster classifications into two classes, there will be $2^{17} = 131072$ different possibilities. The identification of the classes can be performed in two different ways:

1. Instead of selecting random samples to initialize the values μ_1 and μ_2 in the Algorithm 3, we select the minimum and maximum samples from the set $\{t_{1:\ell,p_j}\}$ according to their amplitudes. The initialization in Algorithm 3 is done by assigning $\mu_1 = min\{t_{1:\ell,p_j}\}$ and $\mu_2 = max\{t_{1:\ell,p_j}\}$. It ensures that $\mu_1 < \mu_2$ after the clustering. Then, comparing the resulting cluster means μ_1 and μ_2 with the amplitude of the approximated difference of means $\widehat{D}$, and also $\widehat{\mathrm{Tr}_0}$ and $\widehat{\mathrm{Tr}_1}$, it is straightforward to identify the classes.
2. As all selected leaking points may lead to more than 50 % of exponent recovery, we take one recovered exponent $\widehat{d_{1:\ell,v}}$ from one point of interest v, $v \in \{p_j\}$, and compute the bitwise XOR between this exponent and the other estimated exponent values. Let ℓ be the size of the exponent, $\widehat{d_{1:\ell,p_j}}$ all the recovered exponents for $1 \leq j \leq u$, $p_j \neq v$, and the bitwise results

Table 1. Cluster classification of the (first 40) exponent bits and the recovery of $\widehat{d_{1:\ell,k}}$ using the majority rule.

exponent	classified exponent bits $\widehat{d_{i,k}}$																																							correct
$\widehat{d_{1:40,497}}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	1	0	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	76.02%
$\widehat{d_{1:40,498}}$	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	1	1	76.42%
$\widehat{d_{1:40,166}}$	1	1	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	1	0	0	1	1	0	0	0	0	0	1	76.42%
$\widehat{d_{1:40,462}}$	1	0	0	0	0	1	1	1	1	1	0	0	0	1	1	1	0	1	1	1	1	1	1	0	0	1	1	1	0	1	0	0	1	1	1	1	0	0	1	0	76.42%
$\widehat{d_{1:40,580}}$	1	1	0	0	0	0	0	1	0	1	0	1	1	0	0	1	0	1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	1	0	0	1	0	0	0	1	78.86%
$\widehat{d_{1:40,342}}$	1	0	1	1	0	0	0	0	1	0	0	0	1	0	1	1	0	0	0	1	1	0	1	0	1	0	1	1	0	1	0	0	1	0	1	0	0	1	0	1	79.27%
$\widehat{d_{1:40,282}}$	1	0	0	0	0	0	0	1	0	0	0	1	1	0	1	1	0	1	1	1	1	1	1	0	0	1	1	1	0	1	0	0	1	0	0	1	0	0	0	1	78.86%
$\widehat{d_{1:40,281}}$	1	1	0	1	0	0	0	1	0	0	0	0	1	0	1	1	0	1	1	1	0	1	1	0	1	1	1	1	0	1	1	1	1	0	0	0	0	0	0	1	80.08%
$\widehat{d_{1:40,169}}$	1	0	0	0	0	0	1	1	0	1	0	0	1	1	1	1	0	0	1	1	0	1	1	0	1	1	0	1	0	0	0	0	1	0	0	0	1	0	0	0	80.08%
$\widehat{d_{1:40,165}}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	1	1	1	1	1	0	0	1	1	1	0	1	1	0	0	1	0	0	0	1	1	1	80.89%
$\widehat{d_{1:40,581}}$	1	0	1	0	0	0	0	1	1	0	1	0	1	0	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	1	82.52%
$\widehat{d_{1:40,577}}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	0	1	1	1	1	0	0	1	1	1	0	1	1	0	1	1	0	0	0	0	0	1	82.52%
$\widehat{d_{1:40,284}}$	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	1	0	0	0	0	1	0	1	83.74%
$\widehat{d_{1:40,285}}$	1	0	0	0	0	1	0	1	0	0	0	0	1	0	1	1	0	1	0	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	89.43%
$\widehat{d_{1:40,465}}$	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0	1	1	1	0	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	1	90.65%
$\widehat{d_{1:40,461}}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	1	1	1	1	1	1	0	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	1	91.25%
$\widehat{d_{1:40,464}}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	0	1	1	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	1	93.06%
$\widehat{d_{1:40,k}}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	1	100%
$d_{1:40,k}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	0	0	0	0	0	0	1	100%

$h_{1:\ell}$ =XOR($\widehat{d_{1:\ell,p_j}}, \widehat{d_{1:\ell,v}}$) for $p_i \neq v$. If $\sum_{i=1}^{\ell} h_i < \ell/2$ then returns NOT ($\widehat{d_{1:\ell,p_j}}$), otherwise keep unchanged.

After the cluster classifications and respective association of the classes, the attack enters in the last step in order to combine all estimated exponents into one final exponent.

4.4 Phase 4: Exponent Recovery

The recovery of the final randomized exponent is computed through three different statistical techniques: majority decision, probability density function (pdf) and Bayesian classifier.

Majority Decision. Table 1 shows the cluster classification results for the first 40 bits of each estimated exponent $\widehat{d_{1:\ell,p_j}}$ considering the $u = 17$ points of interest. Using the majority decision we can retrieve a randomized exponent $\widehat{d_{1:\ell,k}}$.

Because the majority rule is a simple statistical procedure, it requires more points of interest for achieving the correct exponent if compared to the next two adopted techniques, as will be demonstrated at the end of this section.

Probability Density Function. In [19], the probability density function, which is based on the normal distributions parameters $\mathcal{N}(\mu_0, \sigma_0)$ and $\mathcal{N}(\mu_1, \sigma_1)$, where μ and σ are the mean and the standard deviation, returns the likelihood that a sample $t_{i,j}$ is the operation when the exponent bit $d_{i,k} = 0$. As the presented analysis is unsupervised, we do not know μ_0 and μ_1. However, the fuzzy k-means cluster classification returns two means or centers μ_1 and μ_2 for each set of compressed samples $\{t_{1:\ell,p_j}\}$ which can be used in place of the means. The standard deviation σ is computed from all the set of samples $\{t_{1:\ell,p_j}\}$, considering the evaluated point of interest p_i. Then, the likelihood that a sample t_{i,p_j} is an operation when $d_{i,k} = 0$ is given by the equation below:

$$p(t_{i,p_j}, \mu_1) = \frac{e^{-\frac{1}{2}(t_{i,p_j}-\mu_1)^2/2\sigma^2}}{e^{-\frac{1}{2}(t_{i,p_j}-\mu_1)^2/2\sigma^2} + e^{-\frac{1}{2}(t_{i,p_j}-\mu_2)^2/2\sigma^2}}, \quad 1 \leq i \leq \ell, 1 \leq j \leq u \quad (6)$$

Following, the defined sum of probabilities gives the likelihood that a set of points of interest $\{t_{i,p_{1:u}}\}$, representing the operation $\langle \mathrm{MS} \rangle_i$, is an operation performed when the randomized exponent bit $d_{i,k} = 0$ and is computed by:

$$S_{0,1:u} = \frac{1}{u} \sum_{j=1}^{u} p(t_{i,p_j}, \mu_1) \quad 1 \leq i \leq \ell \quad (7)$$

Then, for $1 \leq i \leq \ell$, the following decision returns the estimated randomized exponent bit $\widehat{d_{i,k}}$:

$$\widehat{d_{i,k}} = \begin{cases} 0, \text{ if } S_{0,1:u} \geq 0.5) \\ 1, \text{ if } S_{0,1:u} < 0.5) \end{cases} \quad (8)$$

Table 2 shows the final sum of probabilities $S_{0,1:u}$ and the exponent decision from Eq. 8 considering the 20 first exponent bits $\widehat{d_{1:20,k}}$ (for space in Table 2) and $u = 17$ points of interest.

Bayesian Classifier. The Bayesian decision theory makes the assumption that the decision problem is posed in probability terms. The classifier lies on the computation of the posterior probabilities $P(\mu_c|t_{i,p_j})$ which is computed from the prior probabilities $P(\mu_c)$ and the probability density function for normal distributions $p(t_{i,p_j}, \mu_c)$, where $c = \{0, 1\}$ and $p(t_{i,p_j}, \mu_c) \in [0, 1]$. Thus, the classification starts by obtaining the pdf estimation for each point of interest t_{i,p_j} of each operation i. Again, this analysis considers the two cluster centers μ_1 and μ_2 in the place of means and the standard deviation is computed from all the set of compressed samples $\{t_{1:\ell,p_j}\}$:

$$p(t_{i,p_j}, \mu_1) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t_{i,p_j}-\mu_1)^2}{2\sigma^2}} \quad (9)$$

$$p(t_{i,p_j}, \mu_2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t_{i,p_j}-\mu_2)^2}{2\sigma^2}} \quad (10)$$

The probability density functions $p(t_{i,p_j}, \mu_1)$ and $p(t_{i,p_j}, \mu_2)$ are obtained for $1 \leq i \leq \ell$ and $1 \leq j \leq u$. Considering $P(\mu_c)$ as the prior probabilities for

Table 2. Cluster classification of the (first 20) exponent bits and the recovery of $\widehat{d_{1:\ell,k}}$ using the probability density function for normal distributions.

point	probabilities $p(t_{i,p_j}, \mu_1)$																				correct
$\widehat{d_{1:20,p_1}}$	0.1	1.0	0.5	0.6	0.7	0.9	0.9	0.4	1.0	0.7	0.6	0.9	0.0	0.8	0.0	0.2	0.3	0.7	0.5	0.1	76.02%
$\widehat{d_{1:20,p_2}}$	0.3	1.0	0.8	0.6	0.8	0.5	0.4	0.1	0.9	0.7	0.9	0.5	0.7	0.9	0.5	0.2	0.8	0.4	0.1	0.2	76.42%
$\widehat{d_{1:20,p_3}}$	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	76.83%
$\widehat{d_{1:20,p_4}}$	0.2	0.9	0.9	0.8	0.6	0.3	0.5	0.2	0.5	0.3	0.7	0.9	0.6	0.2	0.1	0.2	0.8	0.2	0.1	0.1	76.42%
$\widehat{d_{1:20,p_5}}$	0.5	0.5	0.9	0.9	0.9	0.6	0.8	0.3	0.9	0.1	0.8	0.3	0.2	0.6	0.7	0.1	0.8	0.0	0.3	0.3	78.86%
$\widehat{d_{1:20,p_6}}$	0.1	0.7	0.5	0.4	0.8	0.7	0.7	0.9	0.5	1.0	0.8	0.6	0.2	0.9	0.5	0.3	0.9	0.6	0.7	0.4	78.86%
$\widehat{d_{1:20,p_7}}$	0.5	0.8	0.9	0.8	0.9	0.8	0.8	0.0	0.9	0.6	0.9	0.5	0.1	0.6	0.2	0.1	0.9	0.0	0.3	0.1	78.86%
$\widehat{d_{1:20,p_8}}$	0.3	0.4	0.8	0.5	0.6	0.9	0.7	0.0	0.7	0.5	1.0	0.8	0.2	0.8	0.4	0.1	0.8	0.3	0.0	0.2	80.08%
$\widehat{d_{1:20,p_9}}$	0.4	0.7	1.0	0.9	0.9	0.5	0.4	0.1	0.8	0.2	1.0	0.8	0.1	0.4	0.3	0.1	0.7	0.7	0.1	0.4	80.08%
$\widehat{d_{1:20,p_{10}}}$	0.2	0.9	0.8	0.5	0.7	0.8	1.0	0.4	0.7	1.0	0.8	0.7	0.2	0.5	0.8	0.3	0.6	0.5	0.4	0.1	80.89%
$\widehat{d_{1:20,p_{11}}}$	0.1	0.7	0.3	0.9	0.7	0.7	0.7	0.2	0.5	0.9	0.3	0.8	0.1	0.5	0.0	0.2	0.2	0.3	0.0	0.2	82.52%
$\widehat{d_{1:20,p_{12}}}$	0.1	0.6	1.0	0.8	0.6	0.9	0.8	0.2	0.8	0.9	0.9	0.8	0.1	0.9	0.3	0.6	0.8	0.2	0.5	0.1	82.52%
$\widehat{d_{1:20,p_{13}}}$	0.1	0.8	0.7	1.0	0.7	0.9	0.8	0.7	0.5	0.8	0.9	1.0	0.1	0.2	0.2	0.6	0.6	0.3	0.2	0.3	83.74%
$\widehat{d_{1:20,p_{14}}}$	0.3	0.7	0.7	1.0	0.9	0.3	0.8	0.2	0.9	0.7	0.9	0.6	0.2	0.6	0.4	0.1	0.9	0.1	0.6	0.2	89.43%
$\widehat{d_{1:20,p_{15}}}$	0.1	0.7	0.7	0.9	0.7	0.7	0.7	0.6	0.7	0.9	0.6	0.9	0.2	0.9	0.2	0.4	0.8	0.4	0.2	0.7	89.43%
$\widehat{d_{1:20,p_{16}}}$	0.2	0.8	1.0	0.8	1.0	0.9	0.9	0.3	0.5	0.9	0.7	0.7	0.0	0.8	0.1	0.7	0.8	0.2	0.0	0.2	90.65%
$\widehat{d_{1:20,p_{17}}}$	0.3	0.7	0.6	0.6	1.0	0.7	1.0	0.1	0.8	1.0	0.8	0.7	0.3	0.8	0.3	0.1	0.6	0.5	0.3	0.1	93.06%
$S_{0,1:17}$	0.3	0.6	0.6	0.6	0.7	0.6	0.6	0.3	0.7	0.7	0.7	0.6	0.2	0.6	0.3	0.3	0.7	0.3	0.2	0.3	100%
$\widehat{d_{1:20,k}}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	1	1	100%
$d_{1:20,k}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	1	1	100%

the points of interest p_{j-1}, where $c = \{0, 1\}$, by Bayes's formula we obtain the posterior probabilities $P(\mu_c|t_{i,p_j})$ for the operations i and points of interest p_j:

$$P(\mu_1|t_{i,p_j}) = \frac{p(t_{i,p_j}, \mu_1)P(\mu_1)}{p(t_{i,p_j}, \mu_1)P(\mu_1) + p(t_{i,p_j}, \mu_2)P(\mu_2)} \tag{11}$$

$$P(\mu_2|t_{i,p_j}) = \frac{p(t_{i,p_j}, \mu_2)P(\mu_2)}{p(t_{i,p_j}, \mu_1)P(\mu_1) + p(t_{i,p_j}, \mu_2)P(\mu_2)} \tag{12}$$

The Bayes's formula is repeated for all points of interest p_j over the same operation i. At the end, this estimation returns the probabilities that a certain operation $\langle \text{MS} \rangle_i$ is being executed when the exponent bit $d_{i,k} = 0$. Table 3 shows the evolution of posterior probabilities $P(\mu_1|t_{i,p_j})$ over all points of interest t_{i,p_j}, for $1 \leq j \leq u$, and the respective percentage of correct exponent bits. Again, in this example we consider the first 20 exponent bits.

For the three presented methods, we showed the cluster classification results for $u = 17$ points of interest. Figure 4 demonstrates the evolution of the exponent recovery related to the number of points. In Fig. 4(a), it was considered the evolution from the least to the most leaking point. Note that using the Bayesian classifier 11 points are necessary to recover the entire exponent. The same result can be observed in Table 3. On the other hand, in Fig. 4(b), if the evolution is from the most to the least leaking point, the Bayesian classifier achieves 100 % of the exponent using only 4 points of interest per exponentiation segment.

Table 3. Cluster classification of the (first 20) exponent bits and the recovery of $\widehat{d_{i:\ell,k}}$ using the Bayesian classifier.

point	probabilities $P(\mu_1 \| t_{1:20,p_j})$																				correct
$P(\mu_1 \| t_{1:20,p_1})$	0.7	0.4	0.6	0.8	0.7	0.8	0.6	0.4	0.8	0.8	0.9	0.8	0.4	0.7	0.4	0.3	0.6	0.2	0.4	0.6	75.23%
$P(\mu_1 \| t_{1:20,p_2})$	0.5	1.0	0.8	0.9	0.7	0.9	0.4	0.2	1.0	1.0	0.9	0.8	0.2	0.9	0.0	0.0	0.8	0.2	0.1	0.6	82.89%
$P(\mu_1 \| t_{1:20,p_3})$	0.5	1.0	0.8	0.9	0.7	1.0	0.5	0.1	1.0	1.0	0.9	0.8	0.1	0.9	0.0	0.1	0.9	0.3	0.1	0.6	89.02%
$P(\mu_1 \| t_{1:20,p_4})$	0.1	1.0	0.9	1.0	0.7	0.9	0.9	0.0	1.0	1.0	1.0	0.8	0.0	1.0	0.0	0.0	1.0	0.1	0.0	0.1	93.45%
$P(\mu_1 \| t_{1:20,p_5})$	0.4	1.0	1.0	1.0	0.9	1.0	0.8	0.0	0.9	1.0	1.0	0.6	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.3	96.34%
$P(\mu_1 \| t_{1:20,p_6})$	0.4	1.0	1.0	1.0	0.9	1.0	0.8	0.0	0.9	1.0	1.0	0.6	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.3	97.56%
$P(\mu_1 \| t_{1:20,p_7})$	0.4	1.0	0.9	1.0	1.0	1.0	0.7	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.3	99.59%
$P(\mu_1 \| t_{1:20,p_8})$	0.0	1.0	1.0	1.0	1.0	1.0	0.7	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	99.59%
$P(\mu_1 \| t_{1:20,p_9})$	0.0	1.0	1.0	1.0	1.0	1.0	0.8	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	99.59%
$P(\mu_1 \| t_{1:20,p_{10}})$	0.0	1.0	1.0	1.0	1.0	1.0	0.9	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	99.18%
$P(\mu_1 \| t_{1:20,p_{11}})$	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	100.00%
$P(\mu_1 \| t_{1:20,p_{12}})$	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	100.00%
$P(\mu_1 \| t_{1:20,p_{13}})$	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	100.00%
$P(\mu_1 \| t_{1:20,p_{14}})$	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	100.00%
$P(\mu_1 \| t_{1:20,p_{15}})$	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	100.00%
$P(\mu_1 \| t_{1:20,p_{16}})$	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	100.00%
$P(\mu_1 \| t_{1:20,p_{17}})$	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	100.00%
$\widehat{d_{1:20,k}}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	1	1	100%
$d_{1:20,k}$	1	0	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	1	1	1	100%

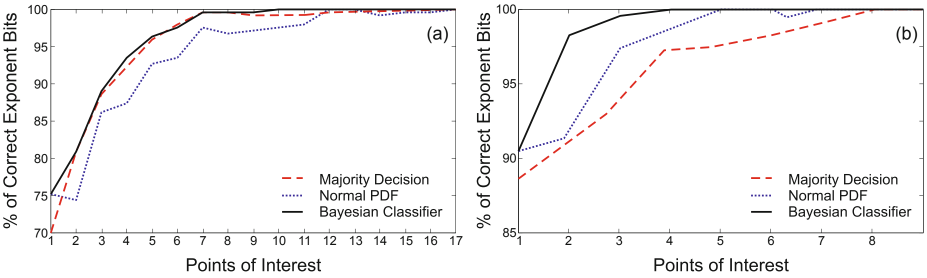

Fig. 4. Relation between the exponent recovery and the number of points of interest: (a) from the least to the most leaking point and (b) from the most to the least leaking point (this Figure is represented in a different scale).

5 Obtaining the Private Key from Randomized Exponents

For decryption and message signing, the retrieval of the randomized exponent $d_r = d + r.\phi(N)$ is the same as retrieving d. Therefore, a single-execution attack is sufficient to break the target device. However, for non-CRT implementations of RSA and in the case when the recovered randomized exponents present few error bits, the adversary can also improve the procedure using a step-by-step attack, as proposed in [19]. In this case, the recovering of several blinding factors r in the exponent randomization is used to derive the exponent d.

Approximately the $\ell/2$ most significant bits of the exponent d are exposed when the public key e is small (3, 17 or $2^{16}+1$). In this procedure, the term $\phi(N)$ is approximated by N and the approximated exponent is given by, $k \in \mathbb{Z}$:

$$\tilde{d} = \left\lfloor \frac{1 + kN}{e} \right\rfloor$$

Consequently, the adversary can obtain the $\ell/2$ most significant bits of all the possible randomized exponents by computing $\tilde{d_{r_i}} = \tilde{d} + r_i.N$, for all i and $r_i \in [0, 2^{32} - 1]$. Considering λ as being the bit length of the blinding factor r, the λ most significant bits of $d + r.\phi(N)$ are equivalent to the most significant bits of $\tilde{d} + r.N$. Then, the adversary can compute all possible values for r and by observing the recovered randomized exponent d_r, he can deduce r. Finally, he constructs the attack on exponentiation traces containing possibly known blinding factors.

6 Countermeasures

The efficiency of single-execution attacks on exponentiations are related to the signal-to-noise ratio (SNR) of acquired traces. Theoretically, the Algorithm 1 is SPA-protected because it is regular and all manipulated data are randomized due to algorithmic (exponent blinding) and arithmetic (leak resistant arithmetic) countermeasures. If different operands are read and written depending on the exponent bits, in a practical implementation of the Montgomery ladder the memory accesses cause addressing related leakages, as demonstrated through the unsupervised attack.

Hardware countermeasures as the insertion of time disarrangement, dummy cycles or frequency dividers reduce the SNR. Balancing the power consumption is an alternative to avoid the leakage of information due conditional tests or control decisions.

If the leakage is related to memory accesses, a possible solution is the randomization of the RAM addresses during the exponentiation. By doing so, the unsupervised attack was unable to entirely recover the randomized exponent. By selecting the same points of interest $P = \{p_j\}$, we applied the fuzzy k-means clustering algorithm and recovered approximately 80 % of the exponent using the Bayesian classifier technique.

7 Conclusions

This paper presented an unsupervised attack on randomized exponentiations. The explored leakages are based on control executions and memory addressing. We proposed to combine the cluster classification for several points of interest over each exponent bit interpretation in order to derive the randomized exponent using a single EM trace. The results were presented through three different statistical techniques and specifically for the probability density function and Bayesian Classifier techniques, we showed the likelihood for the randomized exponent bits.

The presented unsupervised attack demonstrated the efficiency of clustering algorithms against single execution of exponentiations even in the presence of algorithmic and arithmetic countermeasures. The obtained results show the importance of employing hardware countermeasures in public-key architectures.

References

1. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
2. Kocher, P.C.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
3. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 388. Springer, Heidelberg (1999)
4. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
5. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
6. Fouque, P.-A., Valette, F.: The doubling attack – *why upwards is better than downwards*. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 269–280. Springer, Heidelberg (2003)
7. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comput. **48**(177), 243–264 (1987)
8. Joye, M., Yen, S.-M.: The Montgomery powering ladder. In: Kaliski, B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 291–302. Springer, Heidelberg (2003)
9. Bajard, J.-C., Imbert, L., Liardet, P.-Y., Teglia, Y.: Leak resistant arithmetic. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 62–75. Springer, Heidelberg (2004)
10. Bajard, J.-C., Didier, L.-S., Kornerup, P.: An RNS Montgomery modular multiplication algorithm. IEEE Trans. Comput. **47**(7), 766–776 (1998)
11. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski, B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)
12. Herbst, C., Medwed, M.: Using templates to attack masked Montgomery ladder implementations of modular exponentiation. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 1–13. Springer, Heidelberg (2009)
13. Hanley, N., Tunstall, M., Marnane, W.P.: Using templates to distinguish multiplications from squaring operations. IJIS **10**(4), 255–266 (2011)
14. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 2010. LNCS, vol. 6476, pp. 46–61. Springer, Heidelberg (2010)
15. Clavier, C., Feix, B., Gagnerot, G., Giraud, C., Roussellet, M., Verneuil, V.: ROSETTA for single trace analysis. In: Galbraith, S., Nandi, M. (eds.) INDOCRYPT 2012. LNCS, vol. 7668, pp. 140–155. Springer, Heidelberg (2012)
16. Bauer, A., Jaulmes, E., Prouff, E., Wild, J.: Horizontal and vertical side-channel attacks against secure RSA implementations. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 1–17. Springer, Heidelberg (2013)

17. Bauer, A., Jaulmes, É.: Correlation analysis against protected SFM implementations of RSA. In: Paul, G., Vaudenay, S. (eds.) INDOCRYPT 2013. LNCS, vol. 8250, pp. 98–115. Springer, Heidelberg (2013)
18. Heyszl, J., Ibing, A., Mangard, S., Santis, F., Sigl, G.: Clustering algorithms for non-profiled single-execution attacks on exponentiations. In: Francillon, A., Rohatgi, P. (eds.) CARDIS 2013. LNCS, pp. 79–93. Springer, Heidelberg (2014)
19. Bauer, S.: Attacking exponent blinding in RSA without CRT. In: Schindler, W., Huss, S.A. (eds.) COSADE 2012. LNCS, vol. 7275, pp. 82–88. Springer, Heidelberg (2012)
20. Itoh, K., Izu, T., Takenaka, M.: Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA. In: Kaliski, B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 129–143. Springer, Heidelberg (2003)
21. Walter, C.D.: Sliding windows succumbs to Big Mac attack. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, p. 286. Springer, Heidelberg (2001)
22. Dyrkolbotn, G.O., Snekkenes, E.: Modified template attack detecting address bus signals of equal hamming weight. In: The Norwegian Information Security Conference (NISK), pp. 43–56 (2009)
23. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: side-channel atomicity. IEEE Trans. Comput. **51**(6), 760–768 (2004)
24. Guillermin, N.: A coprocessor for secure and high speed modular arithmetic, Cryptology ePrint Archive, report 2011/354 (2011). http://eprint.iacr.org/
25. Perin, G., Imbert, L., Torres, L., Maurine, P.: Electromagnetic analysis on RSA algorithm based on RNS. In: Proceedings of 16th Euromicro Conference on Digital System Design (DSD), pp. 345–352. IEEE, September 2013
26. Lopez, L., Dahab, R.: Fast multiplication on elliptic curves over $GF(2^m)$ without precomputation. In: Koç, Ç.K., Paar, C. (eds.) CHES'99. LNCS, vol. 1717, pp. 316–327. Springer, Heidelberg (1999)
27. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography, Springer Professional Computing (2004)
28. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, Boston (2001)