Vertical and horizontal correlation attacks on RNS-based exponentiations

Guilherme Perin, Laurent Imbert, Philippe Maurine, Lionel Torres

Noname manuscript No.
(will be inserted by the editor)

Vertical and Horizontal Correlation Attacks on RNS-Based Exponentiations

Guilherme Perin · Laurent Imbert · Philippe Maurine · Lionel Torres

Received: date / Accepted: date

Abstract Side-channel attacks are a serious threat for physical implementations of public-key cryptosystems and notably for the RSA. Side-channel leakages can be explored from unprotected cryptodevices and several power or electromagnetic traces are collected in order to construct (vertical) differential side-channel attacks. On exponentiations, the so-called *horizontal correlation attacks* originally proposed by Walter in 2001 and improved by Clavier *et al* in 2010 demonstrated to be efficient even in the presence of strong countermeasures like the exponent and message blinding. In particular, a single trace is sufficient to recover the secret if the modular exponentiation features *long-integer multiplications*. In this paper, we consider the application of vertical and horizontal correlation attacks on RNS-based approaches. The Montgomery multiplication, which is widely adopted in the finite ring of an exponentiation, has different construction details in the RNS domain. Experiments are conducted on hardware (parallel) and software (sequential) and leakage models for known and masked inputs are constructed for the regular and SPA-protected Montgomery ladder algorithm.

1 Introduction

Public-key cryptographic algorithms, like RSA [1] or ECC [2,3], when running on hardware devices leak confidential information through unavoidable side-channels (time, power consumption, electromagnetic radiations, etc). Passive and non-invasive attacks are able of recovering secrets from crypto-devices by analyzing side-channel leakages.

In the context of exponentiation-based public-key algorithms, two categories of attacks have been widely investigated in the recent years: differential power analysis, through different classes of distinguishers (difference-of-means [7], CPA [8], templates [19], MIA [20], clustering [21]) has been largely investigated and need to process multiple traces in order to retrieve the secret. The exponent blinding countermeasure [6], which randomizes the bits of the exponent at each exponentiation, is the main countermeasure. On the other hand, single-trace analyses, based on a single execution of an exponentiation, exploit the leakage by analyzing the trace in a horizontal manner. Since the pioneer work of Kocher [6], sophisticated *horizontal attacks* have been proposed which exploits regularity of long-integer multiplications (LIM) [22–27], targeting the square-and-multiply atomicity [10] or the square-and-multiply always [9]. Their basic principle relies on the identification of a particular operand during modular multiplications.

In this paper, we present a detailed analysis of both vertical and horizontal correlation attacks on RNS-based exponentiations, typically encountered in an RNS-RSA implementation [14]. In this case, the leakage model must be constructed according to the features of the RNS modular exponentiation algorithm. The main goal is to demonstrate the pros and cons of RNS-based countermeasures in hardware and software against correlation electromagnetic analysis.

The rest of the paper is organized as follows. In Section 2.1, we give a brief overview of *Residue Number Systems* (RNS) as well as implementation details about the devices under test. Known-input and masked-input

This work was partly funded by the French Agence Nationale de la Recherche under the grant PAVOIS-ANR-12-BS02-002.

LIRMM - 161, Rue Ada
34095 Montpellier
E-mail: author@lirmm.fr

correlation attacks and their respective leakage models are detailed in Sections 3 and 4. We present the results of our vertical and horizontal correlation attacks in Section 5 and 7 respectively.

2 Preliminaries

2.1 Device Under Test

Experiments were conducted on hardware and software RNS-based RSA implementations. The devices under attack were a Spartan-3E xc3s1600 FPGA and an ARM Cortex M3 32-bit microcontroller STM32F103ZET6. In both cases, we implemented an RNS version of RSA protected against simple analysis with the Montgomery ladder [11] and randomized using the *Leak Resistant Arithmetic* (LRA) approach [12]. Finally, the secret exponent d is protected using the usual exponent blinding strategy: $\delta = d + r\phi(N)$, for a (small) random value r [7]. The term N is the RSA modulus.

In a Residue Number System integers are represented according to a predefined base $\mathcal{B} = (b_1, b_2, ..., b_k)$ of pairwise prime integers, called moduli. Any integer X is thus expressed by a k-tuple of positive integers $\langle X \rangle_{\mathcal{B}} = (x_1, x_2, \ldots, x_k)$, where $x_i \equiv X \bmod b_i$, and $0 \leq x_i < b_i$, (*i.e.* x_i is the remainder in the euclidean division of X by b_i, denoted $|X|_{b_i}$ in the sequel). We know from the Chinese Remainder Theorem, that this representation is unique modulo the product of the elements of $\mathcal{B}$. Arithmetic operations ($\pm$, $\times$) are implicitly performed $B = \prod_{i=1}^{k} b_i$.

In order to perform computations modulo N (with $\gcd(N, B) = 1$), several RNS variants of Montgomery multiplication have been proposed [13–17]. They are all derivations of the original algorithm, where the Montgomery constant is replaced by the constant B, and the usual division by B, otherwise undefined, is computed in an extra RNS basis, say $\mathcal{A}$. The efficiency of an RNS modular multiplication heavily depends on the two implied base extensions between $\mathcal{A}$ and $\mathcal{B}$.

In Algorithm 1, the operation $MM(x, y, N, \mathcal{B}, \mathcal{A})$ returns the Montgomery product $xyB^{-1} \bmod N$ in the two RNS bases $\mathcal{A}$ and $\mathcal{B}$.

Algorithm 1 is easily embedded into any exponentiation algorithm. For this study, we considered the Montgomery ladder that we reinforced with Bajard *et al* Leak Resistant Arithmetic concept [12]. In the LRA, the RNS moduli can be randomized before and/or during the course of an exponentiation. This countermeasure acts as a message blinding technique because it offers a high degree of randomization to the data. The LRA adaptation of the Montgomery ladder is illustrated in Algorithm 2.

Algorithm 1: RNS Montgomery Multiplication (MM) with Fast Approximation Base Extensions [16,15]

Data: x, y ; $1 \leq x, y < N$, in $\mathcal{A} \cup \mathcal{B}$, where $\mathcal{A} = (a_1, a_2, ..., a_k)$, $\mathcal{B} = (b_1, b_2, ..., b_k)$, $A = \prod_{i=1}^{k} a_i$, $B = \prod_{i=1}^{k} b_i$, $\gcd(A, B) = \gcd(B, N) = 1$, $B > 4N$ and $A > 2N$

Result: $w = xyB^{-1} \bmod N$, in $\mathcal{A} \cup \mathcal{B}$

Pre-Computations in $\mathcal{A}$ (*i.e.* mod A): B^{-1}, $-BNB^{-1}$, $A_j^{-1} = (A/a_j)^{-1}$ for all $j = 1, \ldots, k$, and $B_{i,j}NB^{-1}$ for all $i, j = 1, \ldots, k$, where $B_{i,j} = B/b_i \bmod a_j$

Pre-Computations in $\mathcal{B}$ (*i.e.* mod B): $-N^{-1}B_i^{-1}$ for all $i = 1, \ldots, k$, and $A_{i,j}$ for all $i, j = 1, \ldots, k$

$s = |x \times y|_{\mathcal{B} \cup \mathcal{A}}$

#——— Base extension 1 ———

$q_{b_i} = \left| s_i(-N^{-1}B_i^{-1}) \right|_{b_i}$, for $i = 1, \ldots, k$

$f = \lfloor \left(\sum_{i=1}^{k} q_{b_i} \right) / 2^{\omega} \rfloor$

$w_{a_i} = \left| s_i B^{-1} + \sum_{j=1}^{k} q_{b_j}(B_{i,j}NB^{-1}) - fBNB^{-1} \right|_{a_i}$, for $i = 1, \ldots, k$

#——— Base extension 2 ———

$q_i = \left| w_{a_i}(A_i^{-1}) \right|_{a_i}$, for $i = 1, \ldots, k$

$f = \lfloor \left(2^{\omega - 1} + \sum_{i=1}^{k} q_{b_i} \right) / 2^{w} \rfloor$

$w_{b_i} = \left| \sum_{j=1}^{k} q_j A_{i,j} - fA \right|_{b_i}$, for $i = 1, \ldots, k$

Algorithm 2: LRA-RNS Montgomery Powering Ladder [12]

Data: x in $\mathcal{A} \cup \mathcal{B}$, where $\mathcal{A} = (a_1, a_2, ..., a_k)$, $\mathcal{B} = (b_1, b_2, ..., b_k)$, $A = \prod_{i=1}^{k} a_i$, $B = \prod_{i=1}^{k} b_i$, $\gcd(A, B) = \gcd(B, N) = 1$ and $\delta = (\delta_{\ell-1} \ldots \delta_1 \delta_0)_2$

Result: $z = x^{\delta} \bmod N$ in $\mathcal{A} \cup \mathcal{B}$

Pre-Computations: $\langle AB \bmod N \rangle_{\mathcal{A} \cup \mathcal{B}}$

Randomize$(\mathcal{A}, \mathcal{B})$

$A_0 = MM(1, AB \bmod N, N, \mathcal{A}, \mathcal{B})$, 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 $i = \ell - 1$ **to** 0 **do**

  $A_{\overline{\delta_i}} = MM(A_{\overline{\delta_i}}, A_{\delta_i}, N, \mathcal{B}, \mathcal{A})$, in $\mathcal{A} \cup \mathcal{B}$

  $A_{\delta_i} = MM(A_{\delta_i}, A_{\delta_i}, N, \mathcal{B}, \mathcal{A})$, in $\mathcal{A} \cup \mathcal{B}$

end

$A_0 = MM(A_0, 1, N, \mathcal{B}, \mathcal{A})$, in $\mathcal{A} \cup \mathcal{B}$

2.2 Leakage models

In this section, we define the univariate leakage models based on the vertical and horizontal correlation attacks for the RSA-RNS implementations. The leakage modelling is based on the available information to an adversary.

First, we assume that the adversary knows implementation details (i.e., RNS Montgomery algorithm and

the two RNS bases $\mathcal{A}$ and $\mathcal{B}$) and LRA countermeasure is disabled in order to construct a leakage model based on known inputs correlation attacks. We assume that the targeted devices leak the Hamming Weight of internal registers. Therefore, an adversary is able of predicting the intermediate data in the course of the exponentiation by guessing the exponent bits. The leakage is assumed as having deterministic and independent (Gaussian) noise parts. Further, he/she tries to correlate these predicted data with the deterministic part of the leakage.

Second, inputs are randomized by adopting the LRA as message blinding countermeasure. Therefore, a more robust attack based on exploitable leakage of a masked RNS Montgomery ladder (Algorithm 2) is constructed. The proposed attack is based on [29][24][26] and shows how a leakage model can be constructed for randomized inputs if the adversary knows the modular exponentiation algorithm.

The usage of the exponent blinding as an additional countermeasure defines the attack setting as vertical (multiple traces) or horizontal (single traces).

3 Known-Inputs Correlation Attacks

In this section, we detail the Known-input correlation attacks on RNS-based RSA implementations. The term *known* has the same meaning of *unmasked*. We keep the term *known* to be aligned with the terms used in the references [25, 26].

Correlation electromagnetic analysis (CEMA) is the electromagnetic version of CPA and aims at revealing the secret key manipulated by a circuit by analyzing the correlation between its EM emanations and predicted values of an internal register according to the guesses on the secret key.

In order to apply a CEMA, an attacker has to measure the variations of the EM field during the RSA exponentiation using the appropriate equipment. Our setup is composed by an EM station equipped with a XYZ motorized table, a home-made EM probe with a 2mm coil, an 200MHz-BW amplifier, an oscilloscope, an evaluation board and a computer to control the whole setting.

The CEMA procedure also depends on the proper choice for the selection function. It is related to the targeted algorithm, its implementation details and the randomly generated input data x.

3.1 Key guesses and selection function

The first assumption is that the adversary knows the target exponentiation algorithm, in our case the Montgomery ladder (Alg. 2). He generates 8-bit (or less) guesses on the exponent, starting from the MSB. Then, for each guess $h \in \{0, 1, \ldots, 2^8 - 1\}$, he computes a leakage model according to the variations of the electromagnetic field, sampled over a time period $[1, \mathcal{J}]$ and the predicted values of an internal register. Here, the adopted model is based on the Hamming Weight of the register value (we consider that the Hamming Distance Model cannot be applied because the reference state R is unknown). It typically corresponds to the computation of an intermediate value by the algorithm according to the input data x and the sub-key guess h. For any RSA, these intermediate values might be the Montgomery multiplication results. However, for an RNS implementation of RSA,the adversary must know the moduli inside the sets of RNS bases $\mathcal{A}$ and $\mathcal{B}$, as well as construction details of the Montgomery multiplication algorithm (e.g., the base extension method), in order to predict intermediate results. The sets of moduli $\{a_k\}$ and $\{b_k\}$ can be recovered by performing a long and tedious CEMA on the forward conversion (radix to RNS). In this case, the guesses h on the selection function are the values of the moduli itself, instead of the 8-bit portions of the exponent. The set of RNS moduli is usually composed by pseudo-mersenne numbers of the form $2^\omega - c$ where $0 < |c| < 2^{\lfloor \omega/2 \rfloor}$. Considering ω as usually 32 bits, the amount of possible RNS moduli, and consequently the number of guesses, is 2^{16}. Assuming known these RNS bases and the construction details of the Montgomery algorithm, the adversary may now compute the selection function value $d(x_i, h)$ for each input message x_i and for each 8-bit exponent guess h. Because the RNS Montgomery multiplication results are obtained in parallel, he has to choose one RNS channel from $\mathcal{A}$ or $\mathcal{B}$ to compute the selection function value $d(x_i, h)$. Furthermore, assuming υ as being the width of the targeted register D, the selection function follows the linear model $d(x_i, h) = HW(D) - \upsilon/2$.

3.2 Correlation coefficient

The correlation coefficient is computed with Pearson linear correlation between a set of predicted values $d(x_i, h)$ and a set of traces $T_i = \{t_{i,j}\}$, where $1 \leq i \leq N_t$ denotes the trace index and j the sample index. The correlation coefficient $\rho(h, j)$ for each guess h and each sample j is computed as:

$$\rho(h,j) = \frac{\text{cov}\left(d(x_i,h), t_{i,j}\right)}{\sigma_{d(x_i,h)}\sigma_{t_{i,j}}} \quad (1)$$

Since the guesses are made for 8-bit portions of the exponent, the result ρ is a matrix with dimension $2^8 \times \mathcal{J}$. The CEMA is expected to return an estimate $\widehat{h}$ of the key by identifying the row of ρ that gives greater correlation values for some samples j. It is noteworthy that the interval $[1, \mathcal{J}]$ can be reduced by computing the correlation coefficient over the samples j presenting the greatest variability with respect to different input messages x_i.

Algorithm 3: Side-Channel Atomicity - SPA-Resistant Square-and-Multiply

Data: $\tilde{x}$, N, $d = (d_{\ell-1}...d_1 d_0)_2$.

Result: $x^d \bmod N$

$A_0 = 1$
$A_1 = \tilde{x}$
$i = \ell - 1$
$k = 0$
while $i \geq 0$ **do**
  $A_0 = A_0 A_k \bmod N$
  $k = k \oplus d_i$
  $i = i - \neg k$
end
Return A_0

4 Masked-Inputs Correlation Attacks

In the presence of masking countermeasures (or message blinding), intermediate variables are hardly predicted. Techniques used for known-input (unmasked) correlation attacks are thus impractical. However, distinguishers can still exploit information leakage if the masking procedure is executed only once before the exponentiation.

In the sequence of modular multiplications, the goal of an adversary is to detect when two modular multiplications use the same operand (*i.e.*, a register receives the same data at two different, non-consecutive clock cycles). Attacks based on this strategy are called *collision-correlation* attacks [29,24,26–28]. They work differently for each modular exponentiation algorithm. Next, we review collision-correlation attacks on SPA-protected, atomic-square-and-multiply and square-and-multiply-always. The Montgomery ladder is a regular, SPA-protected algorithm, too. Vertical and horizontal Collision-correlation attacks on the Montgomery ladder have been proposed in [32] and [33], respectively. In this work, we provide practical collision-correlation for this algorithm. In section 4.3 we present a leakage model for masked input correlation attacks on the RNS Montgomery ladder.

4.1 Masked-Inputs Correlation Attacks on Square-and-Multiply Atomicity

Side-channel atomicity is SPA-resistant square-and-multiply solution and was proposed in [10] in order to counteract simple side-channel attacks. It consists in removing conditional branching from the left-to-right square-and-multiply algorithm such that the same sequence of instructions are executed whether the exponent bit is a 0 or a 1. It is shown in Algorithm 3.

The ROSETTA (Recovery Of the Secret Exponent by Triangular Trace Analysis) attack was presented in [24]. It is efficient against message and exponent blinding countermeasures and can be considered as an improvement of the horizontal correlation [23] and Big Mac [22] attacks. It is based on two different distinguishers: euclidean distance and collision-correlation. They are constructed so that the result indicates if an operation is a squaring or a multiplication. These distinguishers can also detect if a modular operation performs the long-integer multiplication with a specific input operand by using templates.

Initially, the adversary segments a trace representing a long integer multiplication as illustrated in Fig. 1.

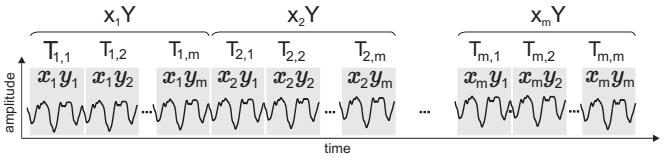

Fig. 1: Long-Integer Multiplication Trace.

Then, he constructs a matrix representing the long integer multiplication $X \times Y$. The squaring matrix, corresponding to the long integer multiplication $X \times X$), is shown in equation (2).

$$T_{X\times X} = \begin{bmatrix} T_{1,1} & \cdots & T_{1,m} \\ T_{2,1} & \cdots & T_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ T_{m,1} & \cdots & T_{m,m} \end{bmatrix} = \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_m \\ x_2x_1 & x_2x_2 & \cdots & x_2x_m \\ \vdots & \vdots & \ddots & \vdots \\ x_mx_1 & x_mx_2 & \cdots & x_mx_m \end{bmatrix} \quad (2)$$

Since the square matrix is symmetric, the adversary may apply an Euclidean distance distinguisher, on both the upper and lower part of the matrix.

$$T_{ED} = \sqrt{\frac{2}{m^2 - m} \sum_{0 \leq i < j < m} (T_{i,j} - T_{j,i})^2} \tag{3}$$

where m is the number of elementary multiplications or *sub-traces*.

If the evaluated long integer multiplication is a squaring, T_{ED} is expected to be close to zero. Otherwise, the operation is likely to be a multiplication. The collision-correlation distinguisher, which computes the Pearson correlation coefficient between two sets of sub-traces, can be used for the same purpose:

$$\rho(t) = \frac{\mathrm{cov}\left(T_{i,j}(t), T_{j,i}(t)\right)}{\sigma_{T_{i,j}(t)} \sigma_{T_{j,i}(t)}} \tag{4}$$

Note that the two series of sub-traces $T_{i,j}$ and $T_{j,i}$ can also be extracted from two different long integer multiplications. Then, an adversary is able of identifying if both modular multiplications are employing the same input operand or not.

4.2 Masked-Inputs Correlation Attacks on Square-and-Multiply Always

The square-and-multiply-always algorithm [9] was designed to address the issue of regularity. As shown in Algorithm 4, a *dummy* multiplication is computed after the squaring when the exponent bit is zero.

Algorithm 4: Square-and-Multiply-Always

Data: $\tilde{x}$, N, $d = (d_{\ell-1}...d_1 d_0)_2$.

Result: $x^d \bmod N$

```
A_0 = 1
for i = ℓ − 1 to 0 do
    A_0 = A_0 A_0 mod N
    if d_i == 1 then
        A_0 = A_0 x mod N
    else
        D = A_0 x mod N
    end
end
Return A_0
```

The result D is always discarded. Hence, even if the attacker is able of distinguish between squarings and multiplications, this would not lead to the recovering of the exponent.

An extension of collision-correlation attack [24] to the square-and-multiply-always algorithm was presented in [26], in which the adversary may identify if the input operand of a multiplication is employed as the input operand of the subsequent squaring, making possible to identify the occurrence of *dummy* multiplications. Since the attack solution is proposed for single traces, it can be considered as an horizontal attack. In [29], the authors presented a cross-correlation attack which can defeat the combination of message blinding (masked-inputs) and square-and-multiply-always countermeasures.

4.3 Masked-Inputs Correlation Attacks on Montgomery Ladder

The Montgomery powering ladder was initially proposed for fast scalar multiplication on elliptic curves in [4] and adapted to any exponentiation in abelian groups in [11]. The Montgomery ladder is also very regular. Moreover, it is more secure against fault attacks (safe-error attacks) than the square-and-multiply-always. For this reason, it is considered as an SPA-protected exponentiation. The Montgomery ladder is shown in Algorithm 5.

Algorithm 5: Montgomery ladder

Data: x, N, $d = (d_{\ell-1}...d_1 d_0)_2$.

Result: $x^d \bmod N$

```
A_0 = 1
A_1 = x
for i = ℓ − 1 to 0 do
    if d_i == 1 then
        A_0 = A_0 A_1 mod N
        A_1 = A_1 A_1 mod N
    else
        A_1 = A_0 A_1 mod N
        A_0 = A_0 A_0 mod N
    end
end
Return A_0
```

One can observe that both registers A_0 and A_1 are updated at each iteration. Hence, the modular multiplications receive different input operands at each iteration. This is illustrated in Table 1 with the four modular multiplications (M1 to M4) performed by two consecutive iterations of the algorithm. Attacks based on the

identification of common input operands in two different modular multiplications are thus impractical.

Table 1: Sequence of modular multiplication in the Montgomery Ladder for consecutive exponent bits.

$d_i d_{i-1} = \ldots 00 \ldots$	$d_i d_{i-1} = \ldots 01 \ldots$
M1: $(x)^i(x)^{i+1} = (x)^{2i+1}$	M1: $(x)^i(x)^{i+1} = (x)^{2i+1}$
M2: $(x)^i(x)^i = (x)^{2i}$	M2: $(x)^i(x)^i = (x)^{2i}$
M3: $(x)^{2i}(x)^{2i+1} = (x)^{4i+1}$	M3: $(x)^{2i}(x)^{2i+1} = (x)^{4i+1}$
M4: $(x)^{2i}(x)^{2i} = (x)^{4i}$	M4: $(x)^{2i+1}(x)^{2i+1} = (x)^{4i+2}$
$d_i d_{i-1} = \ldots 10 \ldots$	$d_i d_{i-1} = \ldots 11 \ldots$
M1: $(x)^i(x)^{i+1} = (x)^{2i+1}$	M1: $(x)^i(x)^{i+1} = (x)^{2i+1}$
M2: $(x)^{i+1}(x)^{i+1} = (x)^{2i+2}$	M2: $(x)^{i+1}(x)^{i+1} = (x)^{2i+2}$
M3: $(x)^{2i+1}(x)^{2i+2} = (x)^{4i+3}$	M3: $(x)^{2i+1}(x)^{2i+2} = (x)^{4i+3}$
M4: $(x)^{2i+1}(x)^{2i+1} = (x)^{4i+2}$	M4: $(x)^{2i+2}(x)^{2i+2} = (x)^{4i+4}$

A way out is to correlate the leakage associated with the clock cycle when the result of M1 (resp. M2) is written in the memory, together with the leakage associated to the loading of the input operands of M4. Let $L1_w$ (resp. $L2_w$) be the leakage associated to the writing of the result of M1 (resp. M2), and let $L4_l$ be the leakage associated to the loading of operands of M4 (See Figure 2.).

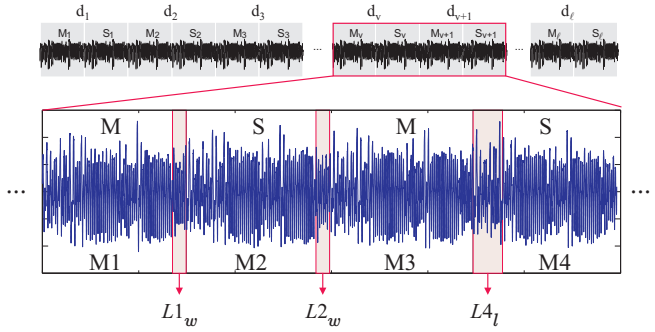

Fig. 2: Leakage model for the masked Montgomery ladder.

A distinguisher may be defined according to Table 1. Simply observe that the correlation coefficient:

$$\rho_{L_2,L_4} = \frac{\mathrm{cov}\,(L2_w, L4_l)}{\sigma_{L2_w}\sigma_{L4_l}}$$

is expected to be greater than:

$$\rho_{L_1,L_4} = \frac{\mathrm{cov}\,(L1_w, L4_l)}{\sigma_{L1_w}\sigma_{L4_l}}$$

when two consecutive bits of the exponent are 0.

In the next sections, vertical and horizontal correlation attacks on RNS-based exponentiation are presented.

5 Vertical Correlation Attacks on RNS-based Exponentiations

5.1 Known inputs correlation attack on hardware

A typical RNS hardware implementation of RSA computes all the instructions over parallel and independent data-paths. Thus, intermediate results are available for all the RNS moduli at the same clock cycle. In this section, we will show how this may be an advantage for the known-input CEMA compared to usual multiple-precision implementations.

In order to set up an attack, the adversary first chooses a channel of $\mathcal{B}$. Then he constructs the leakage model based on a selection function $d(x, h)$ by only considering ω bits of the intermediate result (because the moduli ω-bit integers). In the case of the device under test described in section 2, there are 16 moduli of 32 bits each. Thus the adversary must computes 16 correlation coefficients, one per modulus, to obtain the estimate $\widehat{h}$ for all the RNS channels of $\mathcal{B}$. The presented attack was performed with the acquisition of $N_t = 500$ traces. Fig. 3 shows the results for the electromagnetic correlation analysis for each of the 16 moduli. Note that, for each modulus, the correlation coefficient from (1) is computed by processing the same set of traces.

Observing Fig. 3, the correlation coefficient for the correct (portion of) 8-bit guess is more evidently detected for the channels b_2 and b_{14}. For the other channels, the low correlation coefficients may occur for different reasons: first, the probe position may be far from the targeted register, decreasing the magnitude contribution in the measured electromagnetic field during the time interval the data inside this register is processed; second, the number of measured electromagnetic traces is insufficient to detect this magnitude contribution. Third, a different leakage model may lead to different results. Results of Fig. 3 were obtained with a standard leakage model, i.e., the Hamming weight of each RNS channel of the Montgomery multiplication output result. For a more precise evaluation of the efficiency of the correlation electromagnetic analysis against the hardware implementation, Fig. 4 shows the success rate evolution for all RNS channels with respect to the number of traces. RNS channels b_2 and b_{14} present a success rate of 95% after the processing of 500 traces. The better results obtained for b_2 and b_{14} are related to the EM probe positioning.

Let us now consider a combination of the leakage models from all the moduli in the same correlation coefficient calculation. More precisely, we evaluate:

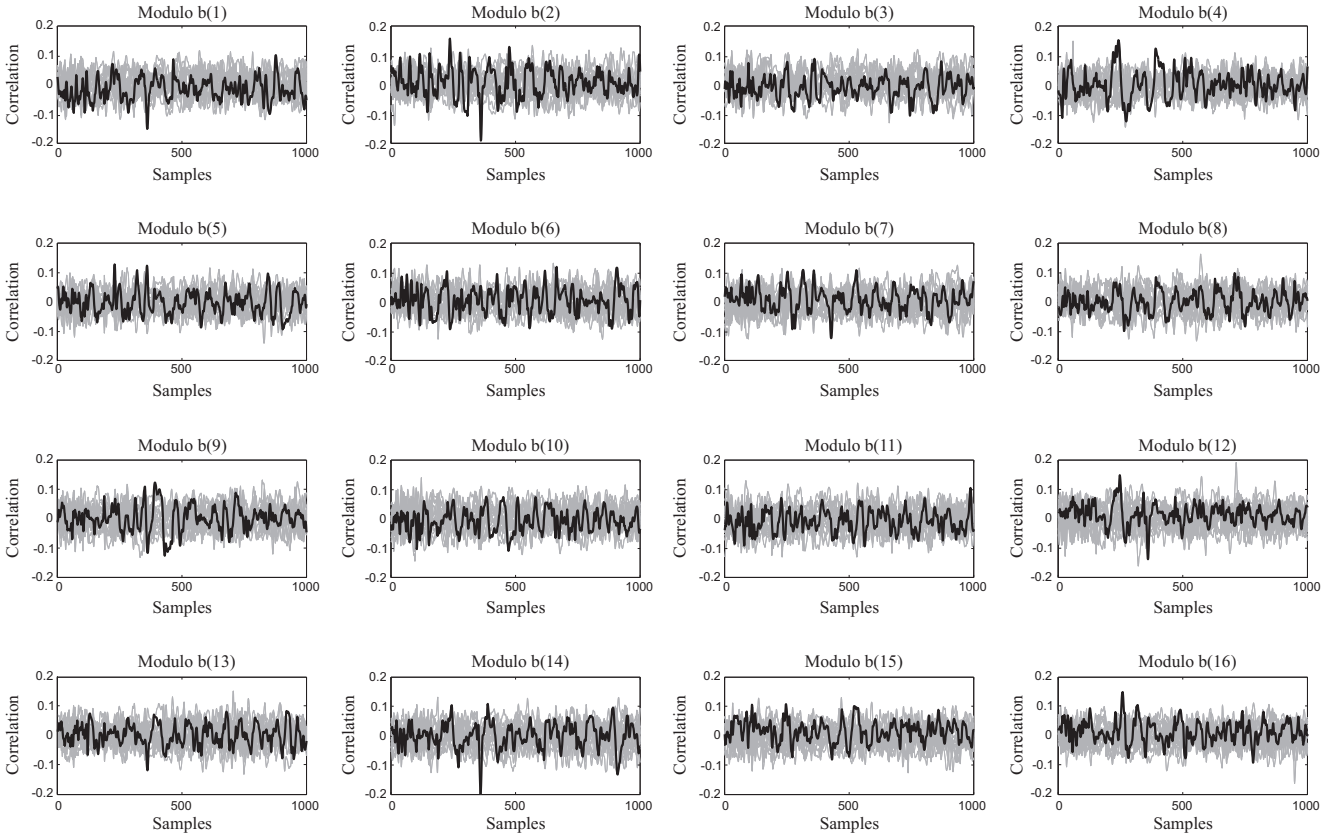

Fig. 3: Correlation electromagnetic analyses on the RNS hardware implementation of RSA for each of the 16 RNS channels.

$$\rho(h,j) = \frac{\sum_{k=1}^{16} \mathrm{cov}\left(d(x_{1:N_t},h)^{(b_k)}, t_{1:N_t,j}\right)}{\sum_{k=1}^{16} \sigma\left(d(x_{1:N_t},h)^{(b_k)}\right) \sum_{k=1}^{16} \sigma\left(t_{1:N_t,j}\right)}, \quad (5)$$

where the function $d(x_{1:N_t},h)^{(b_k)}$ is a vector of predicted intermediate Hamming Weight values according to the input message x and key hypothesis h considering the RNS moduli b_k, for N_t executions of the algorithm. The term $t_{1:N_t,j}$ is a vector in which each element is the sample j over all N_t traces.

The covariances between all the sets of selection function values $d(x_i,h)^{(b_k)}$ and the set of traces $T_i = \{t_{i,j}\}$ are summed up as well as the standard deviations in the denominator. As a result, this formulation returns the combination of leakages from all the RNS channels of base $\mathcal{B}$. It is interesting to note that the number of measured traces remains unchanged. Fig. 5 shows the correlation coefficient obtained from equation (5) for 200 and 500 traces.

The correlation coefficient computation for the correct guess h and for the combined leakage models is detectable by processing around 100 traces (80% of success rate), as shown in Fig. 5(c). Note however that the targeted implementation was unprotected. The RNS bases were fixed during the successive executions of the exponentiation algorithm. This proves, once again, the importance of randomizing the RNS bases.

5.2 Known inputs correlation attack on software

The operations of a modular exponentiation are computed sequentially. The intermediate results in the RNS bases $\mathcal{A}$ and $\mathcal{B}$ are provided at different clock cycles. As a consequence, the leakage model containing the set of selection function values $d(x,h)^{(b_k)}$, for an RNS channel, is associated to the set of traces $T_i = \{t_{i,j}\}$, where the time interval $[1,\mathcal{J}]$ comprises the leakage of this RNS channel only. It is not possible to combine the leakage of all the RNS channels with the same set of trace samples as proposed in the previous subsection.

Nonetheless, the correlation index in software is higher than that of the hardware implementations when the EM probe is located at a proper position (greater leakages of information) over the integrated circuit. At the time of greater linear correlation peak, the microcontroller is only executing the computations regarding

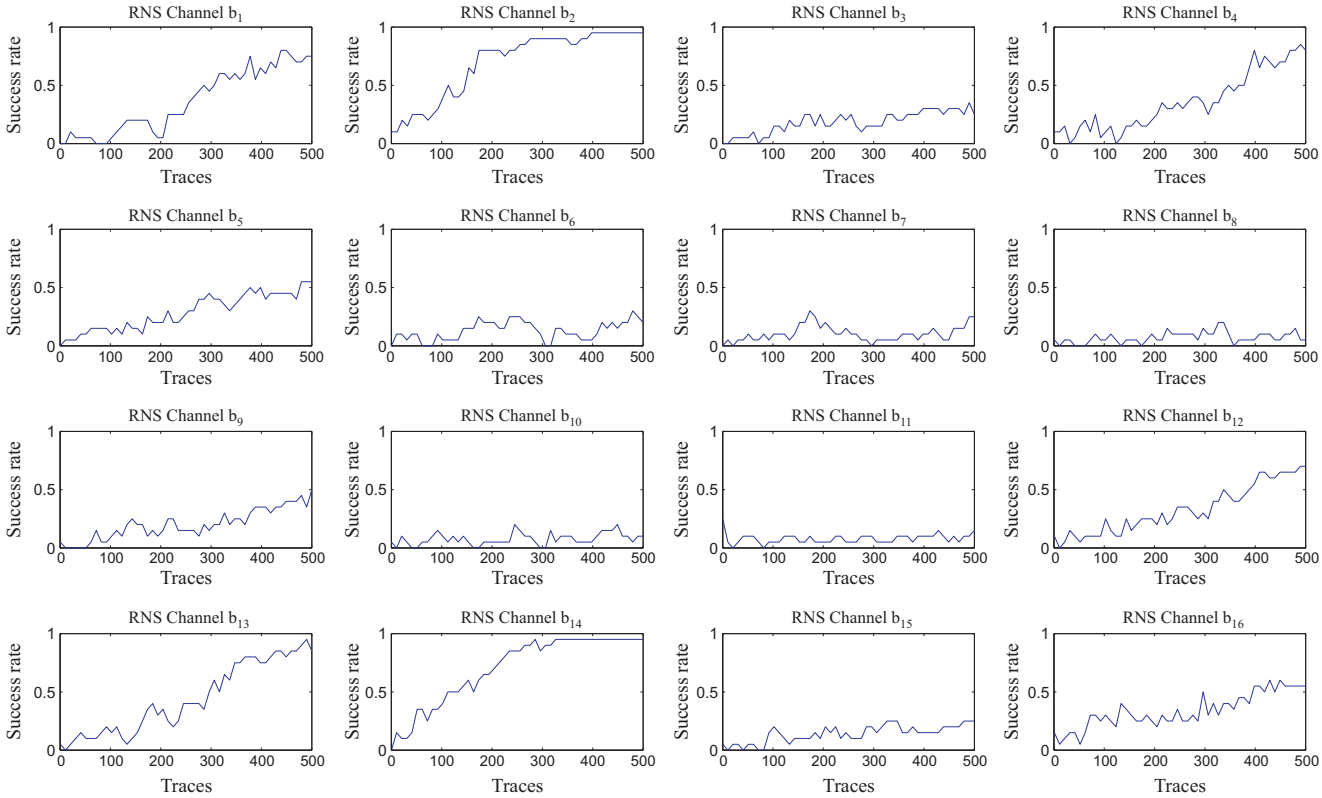

Fig. 4: Success rate for the correlation electromagnetic analyses on the RNS hardware implementation of RSA for each of the 16 RNS channels.

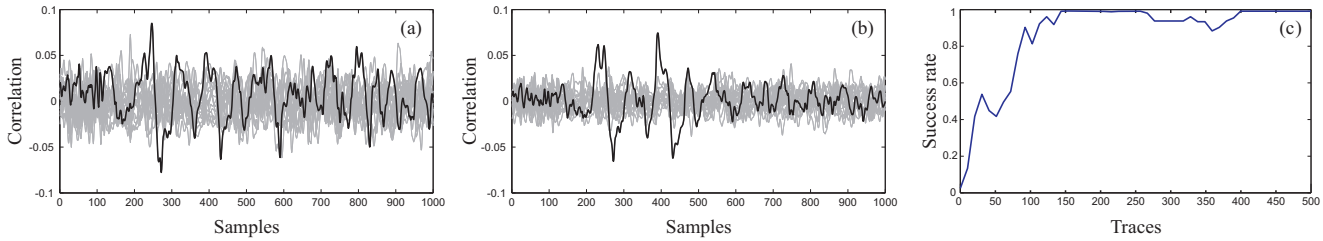

Fig. 5: Correlation Electromagnetic Analyses on the RNS hardware implementation of RSA for all moduli. (a) 200 traces. (b) 500 traces. (c) Success Rate vs Number of Traces.

the targeted intermediate data. There is no magnitude contribution from parallel algorithmic executions (algorithmic noise) during this short time interval. Figure 6 shows the correlation index for CEMA on an unprotected software implementation. Applying (1), the obtained correlation index is equal to 0.82, *i.e.*, much higher than for the incorrect guesses. The success rate of this attack is illustrated in Fig. 6(c). Only 30 traces are necessary to achieve 80% of success.

5.3 Masked inputs correlation attack on hardware

We used the leak resistant arithmetic approach, which acts as a message masking countermeasure. Intermediate results are thus hardly predicted by an adversary. In this case, we considered the leakage model described in Section 4.3 for a masked Montgomery ladder. Note that in the RNS version of the Montgomery ladder (see Algorithm 2), the outputs of operations M1-to-M4, are the registers w_{a_i} and w_{b_i}, containing the partial results of the exponentiation in the RNS bases $\mathcal{A}$ and $\mathcal{B}$.

Assuming the first ν bits of the exponent known, the adversary aims at recovering the exponent bit $d_{\nu+1}$.

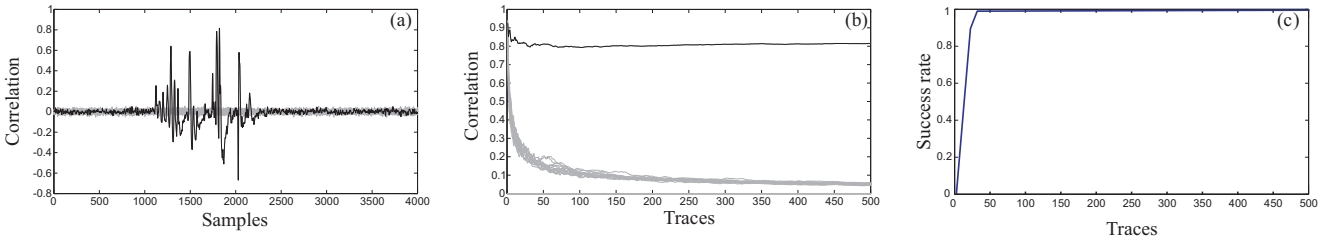

Fig. 6: Correlation Electromagnetic Analyses on the software implementation of RSA without countermeasures (a) Correlation vs time, (b) Correlation vs Number of Traces, (c) Success Rate vs Number of Traces.

Figures 7 and 8 show the correlation coefficients ρ_{L_2,L_4} and ρ_{L_1,L_4} when $d_v = d_{v+1}$, and when $d_v \neq d_{v+1}$ respectively. The correlation coefficients are computed considering the time intervals depicted by $L1_w$, $L2_w$ and $L4_l$ in Fig. 2.

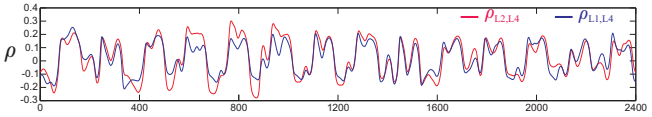

Fig. 7: Masked input correlation electromagnetic analysis on the hardware implementation of RSA ($d_v = d_{v+1}$).

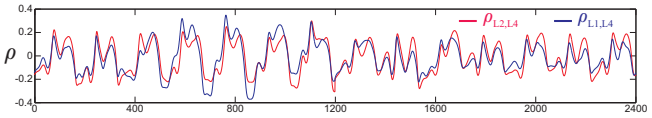

Fig. 8: Masked input correlation electromagnetic analysis on the hardware implementation of RSA ($d_v \neq d_{v+1}$).

In Fig. 7, one can observe that the correlation coefficient ρ_{L_2,L_4} is greater than ρ_{L_1,L_4} for some trace samples (between 400 and 1200). We observe the opposite situation in Fig. 8. Hence, an adversary can compare the obtained correlation results with Table 1 and deduce the exponent bits one by one. Besides the results presented here for the Montgomery ladder, the leakage model for this masked-input vertical correlation attack can be easily extended to square-and-multiply-always and atomic-square-and-multiply exponentiations.

6 Opimizing probe positioning

Electromagnetic side-channel attacks are heavily dependent on the probe position over the surface of the integrated circuit. In order to set the best localization, we performed a scanning procedure by collecting a set of EM traces from each (x, y) position and by computing the correlation coefficient for the correct key value at each position. As will be seen in section 7, horizontal correlation attacks have a limitation of available information, which is related to the key size and implementation aspects of the targeted algorithm. Thus, horizontal attacks results are performed by placing the EM probe at a position providing the best correlation coefficient.

The components of the EM measurement setup is described in Section 3. The EM scans were performed over the surface of the chips, considering an area of 10mm × 10mm. The configured step was 0.33mm. Traces were acquired with a sampling rate of 20GS/sec.

The first evaluation concerns the FPGA Spartan-3E xc3s1600. The clock frequency of the circuit operates at a frequency of 50MHz. One thousand EM traces were measured from each (x, y) position, $1 \leq x, y \leq 30$, and the linear correlation coefficient was computed according to the combined formula (5). The results are illustrated in Fig. 9.

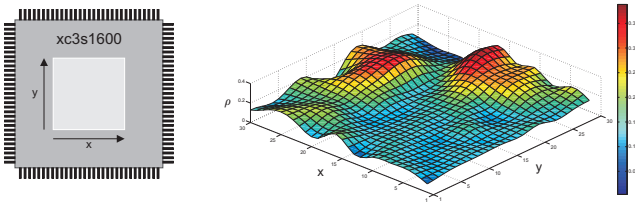

Fig. 9: Correlation coefficient for different (x, y) positions for the FPGA Spartan-3E xc3s1600.

For the STM32F103ZET6 microcontroller, 500 EM traces were acquired for each (x, y) position. The clock frequency was configured to operate at a 48MHz. Fig. 10

shows the scanning results with respect to the linear correlation coefficient obtained from (1).

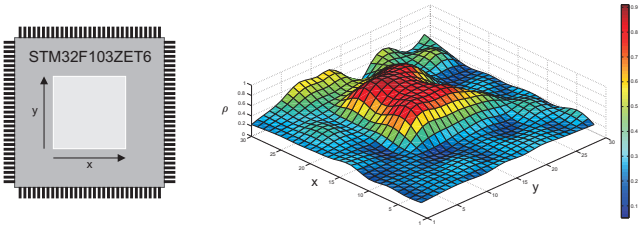

Fig. 10: Correlation coefficient for different (x, y) positions for the STM32F103ZET6 microcontroller.

7 Horizontal correlation attacks on RNS-based exponentiations

Data-dependent attacks, which require hundreds or thousands exponentiation traces cannot be mounted in the presence of the exponent randomization. Likewise, the LRA countermeasure randomizes the intermediate results, and reduces the correlation between the predicted Hamming Weight of a register and source of leakage monitored. In those cases, an adversary must consider an attack based on a single exponentiation. The collected trace should then be analyzed in a horizontal manner.

The first single-execution attack on exponentiations was proposed by Walter in [22]. The so-called *Big Mac* attack considers that long integer multiplications take a large number of cycles and consequently a large number of consecutive elementary multiplications, interleaved with the modular reduction such as the multiple-precision Montgomery multiplication algorithm [5]. This attack detects if an observed operation is a squaring or a multiplication. The basic idea consists in the construction of a template trace characterizing a specific operation (multiplication or squaring) during the long-integer multiplication $X \times Y$ and considering that at least one of the operands (X or Y) will be recalled in the subsequent modular operations. Then, the Euclidean distance between this template trace and each long integer multiplication gives a metric to conclude that such operation is a squaring or a multiplication. The Big Mac attack is a template-based analysis and it is efficient against exponent and message blinding countermeasures.

Clavier *et al* proposed an horizontal correlation attack in [23] based on known inputs for the exponentiations. It also uses the differences between long-integer squarings and real multiplications. The method proposed in [23] correlates the selection function – for example the Hamming Weight $HW(x)$ of a known message x – with carefully selected sampled points of a single trace. It is assumed to be efficient even in the presence of exponent blinding countermeasure because the randomized exponent $\delta = d + r.\phi(N)$ can be used in the place of d for decryption and digital signature. Proposed countermeasures consist in randomizing the loops of the modular multiplication algorithm.

The ROSETTA [24], an improved Big Mac attack, was already recalled in section 4.

In the next sections, we analyse the performance of horizontal attacks on hardware coprocessors implemented with parallel multipliers and we demonstrate that horizontal correlation attacks are inefficient against RNS implementations of RSA. We show that only a sequential implementation of RSA-RNS is vulnerable to horizontal attack when no message blinding or hardware countermeasure are considered.

7.1 Horizontal correlation attack on unprotected RNS implementations of RSA

In this subsection, the Clavier *et at*'s horizontal correlation attack is applied on the RNS implementations of RSA, and its performance and feasibility on parallel (FPGA) and sequential (microcontroller) designs are compared. The implementations are considered as unprotected because the RNS bases are always fixed and the adversary knows them (LRA is disabled).

In the multiple-precision arithmetic, each modular multiplication can be characterized by a long-integer multiplication $X \times Y$. In the RNS context, the modular multiplication involves complex operations as elementary modular reductions and base extensions. Thus, the application of a horizontal correlation analysis requires more knowledge about of implementation aspects of a targeted cryptocore if compared to a multiple-precision-based approach.

Because the efficiency of this attack is related to the amount of elementary multiplications $x_i y_j$, the RNS allows that such computations can be done in parallel and it limits the quantity of available information to the adversary. Therefore, if the application of horizontal correlation attacks on multiple-precision hardware coprocessors is already very difficult, on RNS-based approaches the scenario is even more critical.

Again, we consider that the adversary has a full knowledge about the design and the only unknown value is the exponent. Since he knows the instants of time at which each specific operation is computed, he can associate each predicted results, according to his guesses, to the samples of measured traces.

Hardware - Trace Characterization: Let us consider an EM trace acquired from a single-execution of an exponentiation, running on an FPGA and executed with the RNS Montgomery Ladder algorithm, as illustrated in Fig. 11. The multiplication $X \times Y$ is computed in two RNS bases $\mathcal{A}$ and $\mathcal{B}$. Because the elementary modular multiplications $x_{\mathcal{A}}y_{\mathcal{A}}$ (resp. $x_{\mathcal{B}}y_{\mathcal{B}}$) are computed in parallel and over all the moduli, the adversary has no sufficient information to construct a known-input horizontal correlation attack, as demonstrated by practical results.

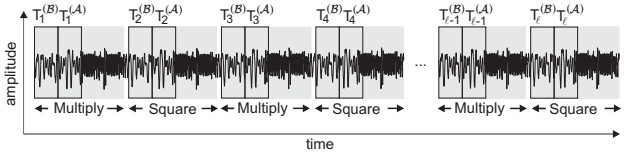

Fig. 11: Montgomery Ladder Exponentiation Trace.

According to the Fig. 11, the adversary can truncate the elementary modular multiplications $x_{\mathcal{A}}y_{\mathcal{A}}$ and $x_{\mathcal{B}}y_{\mathcal{B}}$ of each RNS Montgomery multiplication. These truncated windows are named *sub-traces* herein and denoted by $T_i^{\mathcal{A}}$ (resp. $T_i^{\mathcal{B}}$), which are the time interval of the elementary modular multiplication $x_{\mathcal{A}}y_{\mathcal{A}}$ (resp. $x_{\mathcal{B}}y_{\mathcal{B}}$) of the i-th RNS Montgomery multiplication. Thus, a matrix T of sub-traces $T_i^{\mathcal{A}} = \{t_{i,j}^{\mathcal{A}}\}$ and a matrix H of selection function results $d(x_{\mathcal{B}}y_{\mathcal{B}}, h)$ (resp. $d(x_{\mathcal{A}}y_{\mathcal{A}}, h)$), computed according to the input message x and the guesses h for these intermediate elementary results, are constructed:

$$T = \begin{bmatrix} T_1^{\mathcal{A}} \\ T_1^{\mathcal{B}} \\ T_2^{\mathcal{A}} \\ T_2^{\mathcal{B}} \\ \vdots \\ T_\ell^{\mathcal{A}} \\ T_\ell^{\mathcal{B}} \end{bmatrix} = \begin{bmatrix} t_{1,1}^{\mathcal{A}} & t_{1,2}^{\mathcal{A}} & \cdots & t_{1,\mathcal{J}}^{\mathcal{A}} \\ t_{1,1}^{\mathcal{B}} & t_{1,2}^{\mathcal{B}} & \cdots & t_{1,\mathcal{J}}^{\mathcal{B}} \\ t_{2,1}^{\mathcal{A}} & t_{2,2}^{\mathcal{A}} & \cdots & t_{2,\mathcal{J}}^{\mathcal{A}} \\ t_{2,1}^{\mathcal{B}} & t_{2,2}^{\mathcal{B}} & \cdots & t_{2,\mathcal{J}}^{\mathcal{B}} \\ \vdots & \vdots & \ddots & \vdots \\ t_{\ell,1}^{\mathcal{A}} & t_{\ell,2}^{\mathcal{A}} & \cdots & t_{\ell,\mathcal{J}}^{\mathcal{A}} \\ t_{\ell,1}^{\mathcal{B}} & t_{\ell,2}^{\mathcal{B}} & \cdots & t_{\ell,\mathcal{J}}^{\mathcal{B}} \end{bmatrix} \qquad H = \begin{bmatrix} d(x_{\mathcal{A}}y_{\mathcal{A}}, h) \\ d(x_{\mathcal{B}}y_{\mathcal{B}}, h) \\ d(x_{\mathcal{A}}y_{\mathcal{A}}, h) \\ d(x_{\mathcal{B}}y_{\mathcal{B}}, h) \\ \vdots \\ d(x_{\mathcal{A}}y_{\mathcal{A}}, h) \\ d(x_{\mathcal{B}}y_{\mathcal{B}}, h) \end{bmatrix} \tag{6}$$

The question now is: how many sub-traces are necessary to mount the attack? To demonstrate more precisely this analysis, the entire correct exponent is considered known. As in the classical correlation attack, the adversary can associate each element or row of matrix H to each element of the matrix T, in the case, each sub-trace. Fig. 12(a) presents the correlation coefficient calculation for different guesses for the MSB 8-bit exponent values and considering 250 sub-traces. The success rate is illustrated in 12(b) and for achieving approximately 80% of success rate, at least 165 sub-traces are needed.

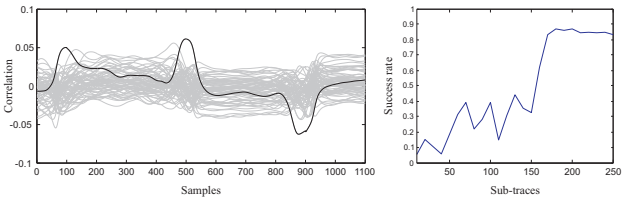

Fig. 12: Horizontal correlation attack on RNS hardware implementation of RSA. (a) Correlation coefficient for the processing of 250 sub-traces. (b) Success rate.

These results lead to the following conclusions: each RNS Montgomery multiplication provides 2 sub-traces. Considering that the adversary wants to recover the exponent byte-by-byte by attacking the Montgomery ladder, he has 32 sub-traces for computing the correlation coefficient for each one of all the 2^8 possible guesses for the exponent bytes. As demonstrated here, the single-execution exponentiation trace was collected from the highest correlation position and with 32 sub-traces the success rate is approximately 10%. Thus, the Montgomery ladder implemented with parallel residue number systems offers strong limitations for the horizontal correlation analysis.

The attack set-up proposed in [26] would be seen as a solution to deal with this limited number of sub-traces. It takes advantage of partial and exposed information about the RSA private key; the upper half part of the exponent (private key) d is exposed (if public key e is small and the) as well as the upper half part of randomized exponents when the exponent blinding equation $\delta = d + r\phi(N)$ is adopted as countermeasure. This is done by approximating $\phi(N)$ by N in the well-known RSA equation $de \equiv 1 \bmod \phi(N)$. There exists a integer $k \in \mathbb{Z}$ such that:

$$ed = 1 + k\phi(N) \tag{7}$$

The Euler's totient function $\phi(N)$, of size ℓ, can be represented by $pq - q - p + 1$. Considering that the size of primes p and q are roughly approximated by $\ell/2$, because $p, q \approx \sqrt{N}$. Then, the upper half part of $\phi(N)$ is equal to the upper half part of the modulus N. For small public key values, such as 3, 17 or $2^{16} + 1$, this condition allows us to deduce the half most significant bits of d. Then, by running all the possible values for $r \in [0, 2^{32} - 1]$, the adversary derives 2^{32} different sequences of modular multiplications for the upper half part of δ. In the Montgomery ladder case, if ℓ is the exponent length, the adversary has an amount of 2ℓ sub-traces. For the target implementation where the bit length is 512 bits, 1024 sub-traces related to the upper

half part of δ would be available to the adversary. Finally, the correlation coefficient is computed for each one of all the 2^{32} possible values of r; the result giving the highest absolute value for the correlation coefficient indicates the best candidate for the random number r. However, this attack would be limited when the adversary must recover the lower half part of d. According to [26], the adversary should guess portions of ω bits of the lower half part of d and verify the correspondent value δ according to the estimated random numbers r in the previous step. To limit the exhaustive search, ω must be small (*e.g.* 8 bits). Therefore, the same problem related to the minimal number of sub-traces arises, limiting the attack.

Software - Trace Characterization: A sequential (software) RNS implementation of RSA is more vulnerable to (known inputs) horizontal correlation attacks. The multiplication $X \times Y$ in the two RNS bases $\mathcal{A}$ and $\mathcal{B}$ needs at least $2n$ elementary modular multiplications $|xy|_{a_k}$ and $|xy|_{b_k}$, for all k, which brings more information (sub-traces) to the adversary. To deal with this available information, the horizontal attack can be constructed by making guesses on portions of the key, thus obtaining all the possible intermediate results in the sequence of modular multiplications according to the guessed exponent and finally horizontally computing the Pearson correlation coefficient by considering a single trace. The RSA-RNS software is implemented with the Montgomery ladder, too. Therefore, considering portions of 8 bits for the exponent guesses, the number of sub-traces representing $|X \times Y|_{\mathcal{A}\cup\mathcal{B}}$ will be $32k$, which is sufficient for identifying the correct exponent bit guess.

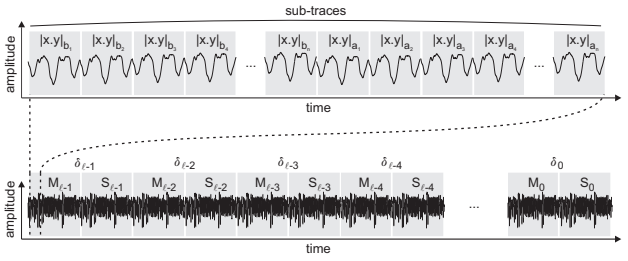

Fig. 13: Sequential RNS Exponentiation Trace.

Let us consider that the first u most significant bits of the randomized exponent, $\delta_{\ell-1:\ell-\nu-1}$, are already known. An adversary guesses all the possible 2^8 values for the next 8 bits, $\delta_{\ell-\nu-1:\ell-\nu-9}$, and store the series of Hamming Weight values of intermediate results $|X \times Y|_{\mathcal{A}\cup\mathcal{B}}$. Then, the matrix T and H are obtained as follows:

$$T = \begin{bmatrix} T^{(1)}_{|xy|_{b_1}} & T^{(1)}_{|xy|_{b_2}} & \dots & T^{(1)}_{|xy|_{b_k}} \\ T^{(1)}_{|xy|_{a_1}} & T^{(1)}_{|xy|_{a_2}} & \dots & T^{(1)}_{|xy|_{a_k}} \\ T^{(2)}_{|xy|_{b_1}} & T^{(2)}_{|xy|_{b_2}} & \dots & T^{(2)}_{|xy|_{b_k}} \\ T^{(2)}_{|xy|_{a_1}} & T^{(2)}_{|xy|_{a_2}} & \dots & T^{(2)}_{|xy|_{a_k}} \\ \vdots & \vdots & \ddots & \vdots \\ T^{(\ell)}_{|xy|_{b_1}} & T^{(\ell)}_{|xy|_{b_2}} & \dots & T^{(\ell)}_{|xy|_{b_k}} \\ T^{(\ell)}_{|xy|_{a_1}} & T^{(\ell)}_{|xy|_{a_2}} & \dots & T^{(\ell)}_{|xy|_{a_k}} \end{bmatrix}$$

$$H = \begin{bmatrix} d(|xy|_{b_1}, h)^{(1)} & d(|xy|_{b_2}, h)^{(1)} & \dots & d(|xy|_{b_k}, h)^{(1)} \\ d(|xy|_{a_1}, h)^{(1)} & d(|xy|_{a_2}, h)^{(1)} & \dots & d(|xy|_{a_k}, h)^{(1)} \\ d(|xy|_{b_1}, h)^{(2)} & d(|xy|_{b_2}, h)^{(2)} & \dots & d(|xy|_{b_k}, h)^{(2)} \\ d(|xy|_{a_1}, h)^{(2)} & d(|xy|_{a_2}, h)^{(2)} & \dots & d(|xy|_{a_k}, h)^{(2)} \\ \vdots & & & \\ d(|xy|_{b_1}, h)^{(\ell)} & d(|xy|_{b_2}, h)^{(\ell)} & \dots & d(|xy|_{b_k}, h)^{(\ell)} \\ d(|xy|_{a_1}, h^{(\ell)}) & d(|xy|_{a_2}, h)^{(\ell)} & \dots & d(|xy|_{a_k}, h)^{(\ell)} \end{bmatrix}$$

Then, the adversary computes the linear Pearson correlation coefficient $\rho(T, H)$. To demonstrate the efficiency of a horizontal correlation analysis in practice, we acquired a single trace from the STM32F1 (ARM) microcontroller implemented with the RSA-RNS algorithm. Fig. 14(a) shows the correlation coefficient during the interval of a sub-trace for all the 2^8 exponent bit guesses. Fig. 14(b) shows the success rate related to the number of sub-traces. Note that in both figures, the correlation coefficient for the correct guess of the exponent bits is the one presenting the highest index.

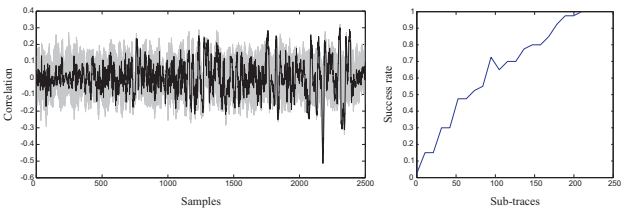

Fig. 14: Horizontal correlation attack on RSA-RNS.

Considerations about HCA: the practical experiments developed in this work revealed that in comparison with classical CPA, horizontal correlation attacks are very hard-working methods. Trace pre-processing is crucial when performing this attack, because the selection and truncation of the sub-traces should be done very carefully.

7.2 Horizontal correlation attacks vs LRA

LRA randomize internal variables in a RNS representation and due to the high amount of possible RNS

moduli combinations, the number of guesses is unaffordable. In this case, the attack proposed in [23] can not be mounted, since the adversary should know (or guess) the processed message x. The modular exponentiation using the LRA countermeasure starts by transforming the input message into the RNS Montgomery domain (Alg. 2, line 4):

$$MM(x, AB \bmod N, N, \mathcal{A}, \mathcal{B}) = (x)(AB \bmod N)(A^{-1}) = xB \bmod N$$

In this case, x is given in the two RNS bases $\mathcal{A}$ and $\mathcal{B}$ and may be known if the two sets of RNS bases are also known. The immunity against horizontal correlation attacks is due to the constant Montgomery B that is carried out in the result. This constant receives randomized values for each modular exponentiation. By doing so, the result xB is also randomized.

The improved methods [24] and [26] are independent of the message blinding, because they correlate trace segments of two long integer multiplications by computing a collision-correlation coefficient between them, or even a distinguisher based on the Euclidean distance. In the following, we explain why ROSETTA analysis is limited by adding the RNS Montgomery Ladder as an algorithmic countermeasure in the first level of abstraction. The operand xB, represented in the Montgomery domain and adopted as a reference to find the collision-correlations, is not recalled as an input operand in the subsequent modular multiplications.

As shown in Table 1, the sequence of modular multiplications always have different input operands, providing different results. As a consequence, the leakage model described in 4.3 can be adapted to a horizontal attack. By doing so, with a practical experiment using an RSA-512 hardware implementation, 61% of the exponent was recovered in the application of the Euclidean distance or Pearson's linear correlation. Therefore, horizontal correlation attacks [23–26] are very limited by the combination of LRA and RNS Montgomery Ladder countermeasures on hardware devices. Moreover, RNS bases can be randomized during the exponentiation by randomizing the the moduli choice before each exponent bit processing. This solution was addressed in [12]. As a consequence, the exponentiation become twice slower, because two extra Montgomery multiplications must be computed each time new RNS basis are selected.

On a software implemented with LRA and RNS Montgomery ladder, the RNS bases *permutations* [30] can be freely jointed as an additional countermeasure. To mount the same attack, the adversary must again correlate the leakage of M1 and M2 with the leakage on M4. As seen in previous sections, our software implementation presented more leakage of information with correlation-based attacks. Permuting the RNS bases means a temporal displacement of computations, defeating this attack. Before the processing of each exponent bit, the position of the moduli inside each RNS base is randomly permuted. In software implementations, this countermeasure can be implemented without any additional time overhead, i.e., the permutations relies in a different indexation of arrays and variables. The Montgomery constants $\mathcal{A}$ and $\mathcal{B}$ remain unchanged and new precomputations each time the RNS bases are permuted are not necessary.

8 Conclusion

In this paper, we evaluated different classes of RSA countermeasures against vertical and horizontal attacks. The algorithmic and arithmetic countermeasures were coupled together with RNS features. We proposed leakage models for known and masked input correlation attacks. The vulnerabilities of a masked RNS implementation of the Montgomery ladder algorithm were demonstrated through the application of vertical correlation attacks.

Horizontal attacks were applied on RNS software and hardware implementations of RSA. The parallel feature of RNS offered resistance against known-input horizontal attacks. Masked input horizontal attacks were defeated by the combination of leak resistant arithmetic, Montgomery ladder and exponent blinding countermeasures. In this case, SPA-related leakages (control decisions, memory addresses) are still a remaining source of leakage and must be defeated by adding hardware countermeasures like RAM addressing randomization or random delays [18]. In [31], the authors demonstrate how to recover the leakage from address-bits (SPA-leakages) in exponentiation by using clustering algorithms. An alternative solution to defeat horizontal correlation analysis is to randomize the RNS bases during the exponentiation, as proposed in [12]. The insertion of more RNS cells in a hardware design or the implementation of dummy RNS channel computations in a software design increase the noise and misalignment in the measured traces, respectively. Redundant modular arithmetic [34] also offers different representation to the numbers, increasing the robustness side-channel attacks.

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. L. Montgomery, "Modular Multiplication Without Trial Division", Mathematics of Computation, 44(170), pp. 519-521, 1985.
6. P.C. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", CRYPTO, pp. 104–1113, 1996.
7. P.C. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis", CRYPTO, pp. 388–397, 1999.
8. 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.
9. 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.
10. B. Chevallier-Mames, M. Ciet and M. Joye, "Low-Cost Solutions for Preventing Simple Side-Channel Analysis: Side-Channel Atomicity", IACR Cryptology ePrint Archive, 2003.
11. 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.
12. 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.
13. J.-C. Bajard, L-Stéphane Didier and P. Kornerup "An RNS Montgomery Modular Multiplication Algorithm," in *IEEE Trans. Computers*, vol. 47, n.7, p. 766-776, 1998, pp. 62–75.
14. J.-C. Bajard and L. Imbert, "A full RNS implementation of RSA," *IEEE Transactions on Computers*, vol. 53, no. 6, pp. 769–774, 2004.
15. F. Gandino, F. Lamberti, P. Montuschi, and J.-C. Bajard, "A general approach for improving RNS montgomery exponentiation using pre-processing," in *Proceedings of the 20th IEEE Symposium on Computer Arithmetic, ARITH20*. IEEE Computer Society, 2011, pp. 195–204.
16. S. Kawamura, M. Koike, F. Sano, and A. Shimbo, "Cox-rower architecture for fast parallel Montgomery multiplication," in *Advances in Cryptology, EUROCRYPT'00*, ser. Lecture Notes in Computer Science, vol. 1807. Springer, 2000, pp. 523–538.
17. K. Posch and R. Posch, "Modulo Reduction in Residue Number Systems", *IEEE Trans. Parallel Distrib. Syst.*, vol 6, number 5, pages 449-454, 1995.
18. S. Mangard, "Hardware Countermeasures against DPA - A Statistical Analysis of Their Effectiveness", Proc. CT-RSA, pp. 222-235, 2004.
19. S. Chari, J.R. Rao and P. Rohatgi, "Template Attacks", *Cryptographic Hardware and Embedded Systems, CHES'02*, ser. Lecture Notes in Computer Science, vol. 2523. Springer, 2002, pp. 13–28.
20. B. Gierlichs, L. Batina, P. Tuyls, B. Preneel, "Mutual Information Analysis - A Generic Side-Channel Distinguisher", *Cryptographic Hardware and Embedded Systems, CHES'08*, Lecture Notes in Computer Science, vol. 5154, pp. 426-442, 2008.
21. L. Batina, B. Gierlichs, K. Lemke-Rust, "Differential Cluster Analysis", *Cryptographic Hardware and Embedded Systems, CHES'09*, ser. Lecture Notes in Computer Science, vol. 5747. Springer, 2009, pp. 112–127.
22. C. Walter, "Sliding Windows Succumbs to Big Mac Attack," *Cryptographic Hardware and Embedded Systems, CHES'01*, ser. LNCS, vol. 2165, pp. 286–299, Springer, 2001.
23. C. Clavier, B. Feix, G. Gagnerot, M. Roussellet and V. Verneuil, "Horizontal Correlation Analysis on Exponentiation," Proc. ICICS, pp. 46-61, 2010.
24. C. Clavier, B. Feix, G. Gagnerot, M. Roussellet and V. Verneuil, "ROSETTA for Single Trace Analysis", Proc. INDOCRYPT, pp. 140-155, 2012.
25. 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.
26. A. Bauer and E. Jaulmes, "Correlation Analysis against Protected SFM Implementations of RSA," Proc. INDOCRYPT, pp. 98-115, 2013.
27. A. Bauer, E. Jaulmes, E. Prouff and J. Wild, "Horizontal Collision Correlation Attack on Elliptic Curves," Reasearch Gate, 2014.
28. A. Moradi, O. Mischke and T. Eisenbarth, "Correlation-Enhanced Power Analysis Collision Attack", *Cryptographic Hardware and Embedded Systems, CHES'10*, ser. LNCS, vol. 6225, pp. 125–139, Springer, 2010.
29. M.F. Witteman, J.G.J. Woudenberg and F. Menarini, "Defeating RSA Multiply-Always and Message Blinding Countermeasures", Proc. CT-RSA, ser. LNCS, vol. 6558, pp. 77-88, 2011.
30. G. Perin, L. Imbert, L. Torres and P. Maurine, "Electromagnetic Analysis on RSA Algorithm Based on RNS", In *Proc. 16th Euromicro Conference on Digital System Design (DSD)*, pages 345–352. IEEE, September 2013.
31. 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, Report 2013/438, 2013.
32. H. Kim, T. H. Kim, J. C. Yoon, and S. Hong, "Practical second-order correlation power analysis on the message blinding method and its novel countermeasure for RSA", ETRI Journal, pages 102–111, vol. 32, no. 1, 2010.
33. N. Hanley, H. Kim, and M. Tunstall, "Exploiting Collisions in Addition Chain-based Exponentiation Algorithms Using a Single Trace", Cryptology ePrint Archive, Report 2012/485, 2012.
34. V. Dupaquis and A. Venelli, "Redundant Modular Reduction Algorithms", Proc. CARDIS, Lecture Notes in Computer Science, vol. 7079, pages 102–114, 2011.