Resilience of randomized RNS arithmetic with respect to side-channel leaks of cryptographic computation

Jérôme Courtois, Lokman A. Abbas-Turki, Jean-Claude Bajard

Resilience of randomized RNS arithmetic with respect to side-channel leaks of cryptographic computation

Jérôme Courtois[1], Lokman Abbas-Turki[2], Jean-Claude Bajard[1]

Abstract—In this paper, we want to promote the influence of randomized arithmetic on the leaks during a code execution. When somebody wants to extract some specific information from these leaks, one can observe different emanations of the device like power consumption. These leaks mostly come from the variations of the Hamming distances of the successive states of the system. This phenomenon is particularly critical for cryptographic devices.
Our work evaluates the resilience of randomized moduli in Residue Number System (RNS) against Correlation Power Analysis (CPA), Differential Power Analysis (DPA). Our analysis is illustrated through the evaluation of scalar multiplication on an elliptic curve using the Montgomery Powering Ladder (MPL) algorithm which protects from Simple Power Analysis (SPA).
We also propose an evaluation based on the Maximum Likelihood Estimator (MLE), which crosses the information of the whole state vector, instead of analysing only the current state like with CPA or DPA. Furthermore, MLE gives better performance and smooths the results allowing a better evaluation of the behaviour of the leakage. Our experimental evaluation suggests that the number of observations, needed to perform exploitable information leakage, is proportional to the number of possible RNS bases.

Index Terms—RNS, moduli randomization, Monte Carlo, ECC, side channel, DPA, CPA, information leakage, Hamming weight, Hamming distance, Maximum Likelihood Estimator

1 Introduction

In asymmetric cryptography, whatever the level of sophistication of the mathematics validating the robustness of the model, the transfer to implementation remains a very delicate point. A naive implementation can seriously compromise the security of a system.

Randomized arithmetic is a possible solution to ameliorate the security. But until now, we had no precise studies on the quality of the randomness generated by randomized arithmetic. We therefore want to fill this lack through this study, by giving elements to measure this quality.

Leaks are mainly due to the variations of the Hamming weights of successive execution states [1], in other words, to the Hamming distance of successive states. We assume the worst case scenario: a malicious person would have access to Hamming distances measurements without any added material noise. We place ourselves clearly on the side of protection by creating this noise intrinsically linked to arithmetic, independently of the specifications of the support hosting the implementation.

The only weakness of this approach is the random draw of the base which, if it is corrupted, makes randomization obsolete. For example, if there is a seed with an entropy defect for a supposedly good quality generator [2], or vice versa a weak generator with a good quality seed [3]. In our study, we consider only attacks by observation, thus physical attacks on the random generator are not tackled.

1 *J. Courtois and J.C. Bajard are with LIP6, Sorbonne Université, Paris. E-mail: jerome.courtois@lip6.fr*
2 *L. Abbas-Turki is with LPSM, Sorbonne Université, Paris.*

We find in literature different ways of using the observed leakages: usually DPA or CPA [4], [5] or other more recent methods like second-order DPA [6], template attacks [7] and Mutual Information Analysis (MIA) [8]. For a survey on different attacks and countermeasures, we refer the reader to [9].

The state transitions depend clearly on the data representation which can be an assumption made by a malicious person. The randomization with respect to an arithmetic system ensures that the computations use different representations from one execution to another. This reduces significantly predictions on the state transitions. We focus in this paper on RNS representation based on the Chinese Remainder Theorem. Each value is known by its residues over a set of co-prime numbers which represents the RNS base. The authors of [10], [11] suggested randomization using curve isomorphisms as counter measure against template attacks on ECDSA. This work was extended in 2016 [12] on attacks on doubling point operation on elliptic curves in "mbed TLS". Therefore, a randomized RNS arithmetic offers a good opportunity to randomize independently from the cryptographic algorithm used. Thus, the methods presented in the paper are suitable for any cryptosystem such as RSA, ECC, Euclidean Lattice or others. RNS is scalable, and can be adapted to key sizes which could increase in function of the cryptanalysis progress. Furthermore, RNS computations can be efficiently parallelized [13]. For further benefits of RNS, we refer to [14].

In [15], the authors showed that we can draw randomly a RNS base from a set of moduli, to randomize an execution with a small cost. Since their publication, this work was used and cited in different papers [14], [16], [17], [18], [19]. To

our knowledge, no one established a complete study of the randomness behavior of such approach, and what kind of protection it can get.

We wish to fill this gap. We present here a study establishing the link between the number of elements n of the base participating in the draw and the size S of the sample necessary for exploitable information. We use 112 bits ECC curve essentially to illustrate the results and conjecture that $S = O((2n)!/(n!)^2)$. The results are quite similar when dealing with Edwards curves of 255 bits [20] or ECCsecp256r1 [21].

The layout of this paper is as follows: in Section 2, we briefly introduce the moduli randomization of the RNS representation in the Montgomery algorithm applied for ECC and the goal of randomization. Section 3 explains the main reasons why the resilience of a system should rather focus on about 10 successive Hamming distances. Section 4 recalls the Maximum Likelihood Estimator (MLE) and studies the size S of observations needed to achieve the analysis.

2 Key elements of the study: Montgomery Power Ladder (MPL) using RNS representation applied to ECC and randomization.

In Section 2.1, we explain briefly the randomization technique based on RNS representation for Montgomery multiplication. Section 2.2 clarifies the way the Hamming distances are computed through the successive steps of MPL. Because ECC uses the main arithmetic operations as additions and multiplications, this makes the randomization efficient. We give in Section 2.3 some elements about our evaluation of the randomness.

2.1 Montgomery for RNS modular multiplication

In [22], P. Montgomery introduced an algorithm of modular multiplication to avoid trial division by large numbers. The RNS version of this algorithm is the starting point of the randomization used in [23]. We summarize this method with a presentation that is quite similar to the one in [24].

We denote $|a|_m = a \ mod \ m$ and $[\![1,n]\!] = \{1,...n\}$. When a and m are coprime, we set $|a|_m^{-1} = a^{-1} \ mod \ m$ to be the inverse of a modulo m. Introducing the RNS base $\mathcal{B}_n = \{m_1, ..., m_n\}$ of pairwise coprime moduli, the Chinese Remainder Theorem ensures the existence of a ring isomorphism between $\mathbb{Z}_M$ and $\mathbb{Z}_{m_1} \times \cdots \times \mathbb{Z}_{m_n}$ with $M = \prod_{i=1}^{n} m_i$. Thus, for any positive integer X strictly smaller than M

$$X = \left(\sum_{i=1}^{n} x_i \left| M_i \right|_{m_i}^{-1} M_i \right) \bmod M \tag{1}$$

with $x_i = |X|_{m_i} = X \bmod m_i$ and $M_i = M/m_i$.

Let $\widetilde{\mathcal{B}}_n = \{\widetilde{m}_1, ..., \widetilde{m}_n\}$ be another RNS base of pairwise coprime moduli that are also coprime with $\mathcal{B}_n$, i.e. m_i and $\widetilde{m_j}$ are coprime for each $i \in [\![1,n]\!]$ and $j \in [\![1,n]\!]$. For a number X that is strictly smaller than $\widetilde{M} = \prod_{i=1}^{n} \widetilde{m}_i$, we use the notation $\{\widetilde{x}_1, ..., \widetilde{x}_n\}$ for the decomposition of X on $\widetilde{\mathcal{B}}_n$. Using these notations as well as the standard definition of the usual RNS operations, Algorithm 1 presents the modular multiplication. Addition $+_{RNS}$, multiplication $\times_{RNS}$ and opposite $(-X)_{RNS}$ are explained in [24].

Algorithm 1 RNSn modular multiplication

Require:
A residue base $\mathcal{B}_n = \{m_1, ..., m_n\}$ where $M = \prod_{i=1}^{n} m_i$
A residue base $\widetilde{\mathcal{B}}_n = \{\widetilde{m}_1, ..., \widetilde{m}_n\}$ where $\widetilde{M} = \prod_{i=1}^{n} \widetilde{m}_i$ with $gcd(M, \widetilde{M}) = 1$
A modulus N expressed in $\mathcal{B}_n$ and $\widetilde{\mathcal{B}}_n$ with $gcd(N, M) = 1$ and $gcd(N, \widetilde{M}) = 1$,
$0 < (n+2)^2 N < M$ and $0 < (n+2)^2 N < \widetilde{M}$
An Integer A expressed in $\mathcal{B}_n$ and $\widetilde{\mathcal{B}}_n$
An Integer B expressed in $\mathcal{B}_n$ and $\widetilde{\mathcal{B}}_n$ with $AB < NM$
Ensure: An integer R expressed in $\mathcal{B}_n$ and $\widetilde{\mathcal{B}}_n$ such that $R \bmod N = ABM^{-1} \bmod N$
function
$Q \leftarrow ((-(A \times_{RNS} B))_{RNS}) \times_{RNS} N^{-1}$ in base $\mathcal{B}_n$
Extension of Q from $\mathcal{B}_n$ to $\widetilde{\mathcal{B}}_n$
$R \leftarrow (A \times_{RNS} B +_{RNS} Q \times_{RNS} N) \times_{RNS} M^{-1}$ in base $\widetilde{\mathcal{B}}_n$
Extension of R from $\widetilde{\mathcal{B}}_n$ to $\mathcal{B}_n$
end function

Before each modular exponentiation, we perform a random selection of n moduli $\{m_1, ..., m_n\}$ among $\{\mu_1, .., \mu_{2n}\}$ for base $\mathcal{B}_n$. The remaining moduli form the base $\widetilde{\mathcal{B}}_n$. This random choice is based on a standard drawing without replacement.

Since many modular multiplications are needed in ECC or RSA, one should consider the Montgomery form of A and B as inputs to Algorithm 1. This trick allows to circumvent dealing with $ABM^{-1} \bmod N$ as an output. We recall that the Montgomery form of A is given by $AM \bmod N$. As proposed in [15], once $M\widetilde{M} \bmod N = \prod_{i=1}^{2n} \mu_i \bmod N$ is known, the Montgomery form can be obtained with Algorithm 1. It is applied to A and $M\widetilde{M} \bmod N$ provided that we exchange $\mathcal{B}_n$ and $\widetilde{\mathcal{B}}_n$, since

$$A \times |M\widetilde{M}|_N \times \widetilde{M}^{-1} = AM \bmod N.$$

To recover the appropriate expression, we need to perform a final pass in Algorithm 1 to multiplicate 1 and $(AM)(BM)M^{-1} \bmod N$ that yields

$$|(AM)(BM)M^{-1}|_N \times |1|_N \times M^{-1} = AB \bmod \mathrm{N}.$$

We point out that pre-computing $|M\widetilde{M}|_N$, proposed in [15] instead of $|M^2|_N$, is justified by the randomization procedure.

We refer to Appendix A for extensions used in the RNS modular multiplication.

Remark: RNS has become a standard for randomization, especially since there is a great diversity of moduli. Moreover, with Montgomery multiplication algorithm [22], the Montgomery factor strengthens the random behaviour of Hamming distances.

2.2 Example on scalar multiplication on elliptic curve

We target the scalar product of a point of an elliptic curve. This operation is found in several Elliptic Curves Cryptographic protocols: encryption, signature, etc... [25], [26]. It offers the benefit of being quite complete in terms of arithmetic operations. Several scenarii are possible: the secret key is used several times as for decryption in PSEC-KEM [21], or only once as the random secret in ECDSA [26]. Different attacks are then available in the first case via CPA or DPA, in the second case via learning.

Anyway, the secret is in the form of a scalar K applied to a point G of an elliptic curve E. Thus the computation of $[K]G$ must remain secret by being leak resistant.

To compute $[K]G$ on an elliptic curve and protect against Simple Power Analysis (SPA) [27], we use the binary version of MPL detailed in Algorithm 2. First, we compute both the Montgomery Form A_0 of G and A_1 the double of A_0. Then, if the bit value b_i of K is one, A_0 is added to A_1 memorized in A_0 and A_1 is doubled. Otherwise, A_1 is added to A_0 memorized in A_1 and A_0 is doubled.

Algorithm 2 Montgomery Powering Ladder for ECC in RNSn

Require:
A point $G = (X; Y; 1)$ in RNS representation
A key $K = 2^{d-1}b_0 + 2^{d-2}b_1 + ... + 2b_{d-2} + b_{d-1}$
Ensure:
$A_0 = [K]G$
$(H_i)_{i\in\{0,..,d-1\}}$, the Hamming distances
function
Choose a random base permutation
$A_0 = (|XM|_N, |YM|_N, |M|_N)$, *Montgomery form of G*
$A_1 = [2]A_0$
H_0=**Hamming weight of** (A_0, A_1)
for i=1 to d-1 **do**
$A'_0 = A_0$ and $A'_1 = A_1$
$A_{\overline{b_i}} = A_{\overline{b_i}} + A_{b_i}$
$A_{b_i} = [2]A_{b_i}$
H_i = **Hamming distance between** (A_0, A_1) **and** (A'_0, A'_1)
end for
Result $A_0 = (|X'M|_N, |Y'M|_N, |Z'M|_N)$ *in Montgomery form*
Return to the Non-Montgomery form [28]
$A_0 = (|X'|_N, |Y'|_N, |Z'|_N)$
end function

The elliptic curve domain of $E(\mathbb{F}_N)$ is defined by a finite field $\mathbb{F}_N$ with N a prime number, two elements a and b $\in \mathbb{F}_N$, an equation $E : y^2 \equiv x^3 + ax + b \bmod N$, $G(x_G, y_G)$ a point base of $E(\mathbb{F}_N)$ and n_G is a prime number that is the order of G on $E(\mathbb{F}_N)$.

In our implementation, we use the elliptic curves recommended by Certicom [21] employing Jacobian coordinates that avoid the division and reduce computations [29], [30], [31]. Each point is defined by three Jacobian coordinates $(X; Y; Z)$ with the affine representation $(X/Z^2; Y/Z^3)$.

Associated to the equation E is $Y^2 = X^3 + aXZ^4 + bZ^6$, $(X; -Y; Z)$ is the inverse of $(X; Y; Z)$ and the infinite point is chosen to be equal to $(1; 1; 0)$. The addition and doubling operations can be found in [29].

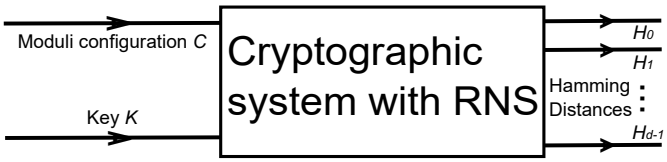

Fig. 1. Hamming distances with respect to randomness sources

Algorithm 2 shows exactly at which step of MPL we choose to compute the Hamming distances for ECC in RNS. We remind that M is the product of the moduli of base $\mathcal{B}_n$.

2.3 Goal of randomization

The goal of randomization according to the involved moduli is to make as unpredictable as possible the secret from the Hamming distance (number of different bits) between two consecutive states. As sketched on Figure 1, we distinguish two randomness sources given by both the configuration of moduli C and the key K. K is also considered as a random variable in our analysis.

$H = (H_0, ..., H_{d-1})$ are the Hamming distances observed through the execution of a cryptographic algorithm. For example, we study here MPL algorithm associated to ECC. The random vector $H = (H_0, ..., H_{d-1})$ can be considered as a deterministic function of the couple of random variables (K, C). Consequently, the link between K and H is difficult to establish when the noise generated by C is significant. The perfect noise would be the one that mimics an independence between K and H. Without loss of generality, let us denote informally:

- $L(H, K)$ the joint distribution of (H, K),
- $L(H|K)$ the conditional distribution of H given K,
- $L(H)$ and $L(K)$ the marginal distributions of H and K.

The perfect noise must fulfill

$$L(H, K) = L(H|K)L(K) = L(H)L(K) \tag{2}$$

or equivalently $L(H|K) = L(H)$, meaning that K must not provide any information on H. Although (2) is impossible to obtain because each H_i as a function of K and C, will always depend on K, it tells us that the independence of the coordinates of $H = (H_0, ..., H_{d-1})$ is not necessary to have a perfect noise.

This paper studies the distinguishability between $L(H|K)$ and $L(H|K')$ $(K \neq K')$ with respect to n, the number of moduli in the RNS representation denoted by RNSn. Nevertheless, because studying $L(H|K)$ for the whole vector H is computationally barely possible, we develop a strategy based on few Hamming distances that provide the most valuable information on the key K. Unlike DPA and CPA that use only the marginal information associated to each step, our conditional strategy combined with MLE uses a cross-information based on about 10 Hamming distances. Therefore, our main contribution can be summarized by:

1) We show that a cryptographic system has to be resilient with respect to the cross-information.
2) The randomization makes CPA inefficient and we explain why MIA applied to the randomization fails.
3) The DPA results are inconsistent with the level of randomization and we show that second-order DPA does not overcome this inconsistency.

4) We used the Maximum Likelihood Estimator (MLE) to measure the level of information leakage. We show that the information leakage, is monotonous, unlike the DPA.
5) We propose a number of randomized moduli to protect ECC from a template attack.

In real implementation, physical hardware noises will be added to the noise generated by the random selection of RNS bases. This makes harder the detection of the Hamming distances. In our study, as we want to measure the impact of the randomization of the RNS bases, we consider only the ideal case when the Hamming distances are known. We focus on ECC as it is well suited for the evaluation of moduli randomization countermeasure because its arithmetic is dominant and performing DPA can be efficient. Indeed, all the probability tools like: total variation, covariance matrix, asymptotic error of Monte Carlo (Cf. Appendix B) can be reused in the same fashion for other systems. We point out also that our work is different from [16] where the authors study protection against memory addressing attack. Indeed, our paper focuses on the resilience of randomization when the system is supposed to be protected against memory addressing attacks. We use 112 bits ECC curve essentially to illustrate the results and conjecture that $S = O((2n)!/(n!)^2)$. The results are quite similar when dealing with Edwards curves of 255 bits [20] or ECCsecp256r1 [21].

3 A CONDITIONAL STRATEGY AND LIMITATIONS OF CPA, DPA, SECOND-ORDER DPA AND MIA

With randomized RNS, we verify that Hamming distance has a Gaussian distribution (see Figure 4). Unfortunately, most of the statistic tests, like NIST's ones [32], evaluate uniform distribution. Therefore, we mainly use methods from side-channel attacks as tools for assessing randomization.

The randomization of moduli effectively generates noisy data. Consequently, we should target the most sensitive values that provide exploitable information on the secret key K. When the latter fact is studied in Section 3.1, Section 3.2 shows that CPA is impossible to use and the size S of observations to achieve a DPA is not monotonous with respect to the number of moduli. Section 3.3 discusses the adaptation of other methods to RNS randomization.

3.1 Sufficient information and conditional strategy

From now on, we denote S the sample size of simulations. The following three properties of Hamming distances are presented:

α) For fixed choice of moduli, the first Hamming distances are the one that provide the strongest information (dependence) on the secret K.
β) For fixed choice of moduli, the correlation between Hamming distances decreases significantly with respect to the gap l that separates H_i and $H_{i\pm l}$.
γ) Under randomization of moduli, each Hamming distance H_i has a normal distribution. This property shall not make absurd the assumption that the whole vector H is Gaussian.

Consequently, the first bits of a secret key K can be deduced from the information extracted from the first Hamming distances. Once these few first bits known, the following ones can be found step by step, until we recover the whole key.

At each step of Algorithm 2 (MPL), we evaluate the dependency of the random variables K and H_i. The values of K and H_i are integers belonging to the intervals $I = [0, 2^p[$ $(p \leq d)$ and $\mathcal{H}^i = [min(H_i), max(H_i)]$. In order to reduce the complexity of computations and increase the Monte Carlo accuracy (Cf. Appendix B), we use appropriate subdivisions of those intervals I and $\mathcal{H}^i$ respectively into $2^{p'}$ and q sub-intervals. The details of the subdivisions are given in Appendix C.

Monte Carlo is used for the estimation of probability terms involved in the Total Variation to Independence (TVI) [33] expression (3) given below. The Law of Large Numbers ensures the convergence and the Central Limit Theorem provides its rate. Consequently, we quantify the accuracy thanks to the 95% confidence interval, with 95% chance of having at most a 10% relative error. The relative error is defined as the width of the confidence interval normalized by the estimated value.

As introduced informally above, the perfect noise must fulfill $L(H, K) = L(H)L(K)$ (2) which yields

$$P\left(H_i \in \mathcal{H}_j^i, K \in I_k\right) = P\left(K \in I_k\right) \times P\left(H_i \in \mathcal{H}_j^i\right).$$

Thus the dependence is quantified through the distance between the probability of the product $P\left(H_i \in \mathcal{H}_j^i, K \in I_k\right) = P\left(K \in I_k\right) \times P\left(H_i \in \mathcal{H}_j^i | K \in I_k\right)$ and the product of probabilities $P\left(K \in I_k\right) \times P\left(H_i \in \mathcal{H}_j^i\right)$. We compute this distance using TVI [33] given by

$$\text{TVI}_i = \frac{1}{2} \sum_{k=0}^{2^{p'}-1} \sum_{j=0}^{q-1} P\left(K \in I_k\right) \left| P\left(H_i \in \mathcal{H}_j^i\right) - P\left(H_i \in \mathcal{H}_j^i | K \in I_k\right) \right|. \tag{3}$$

The value of $P\left(K \in I_k\right)$ is known since we draw uniformly an integer value on $[0, 2^p[$. However, the value $P\left(H_i \in \mathcal{H}_j^i | K \in I_k\right)$, and subsequently $P\left(H_i \in \mathcal{H}_j^i\right)$, is approximated using Monte Carlo simulation. For more mathematical details on Monte Carlo, we refer the reader to [34].

In Figure 2, we calculate TVI_i for each step in MPL either for a fixed choice of moduli or when they are randomized. When the moduli configuration is fixed we draw only independent keys $\{K^l\}_{1 \leq l \leq S}$. When the moduli are randomized we draw independent couples $\{(K^l, C^l)\}_{1 \leq l \leq S}$ of keys and moduli configurations. The Monte Carlo approximation is then given either by

$$P\left(H_i \in \mathcal{H}_j^i, K \in I_k\right) \approx \frac{1}{S} \sum_{l=1}^{S} 1_{\left\{H_i(K^l) \in \mathcal{H}_j^i \bigcap K^l \in I_k\right\}}.$$

or by

$$P\left(H_i \in \mathcal{H}_j^i, K \in I_k\right) \approx \frac{1}{S} \sum_{l=1}^{S} 1_{\left\{H_i(K^l, C^l) \in \mathcal{H}_j^i \bigcap K^l \in I_k\right\}}.$$

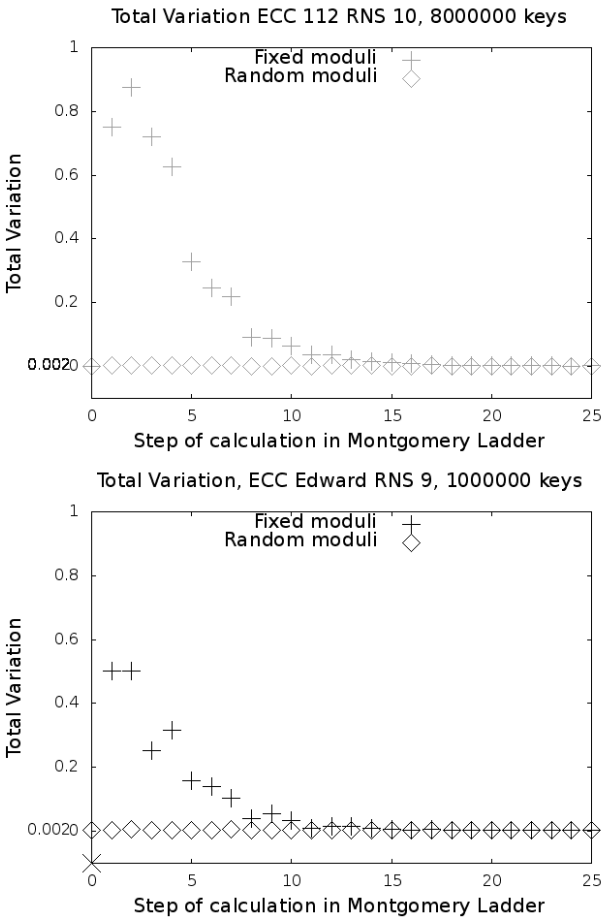

Fig. 2. Total variation as a function of the calculation step.

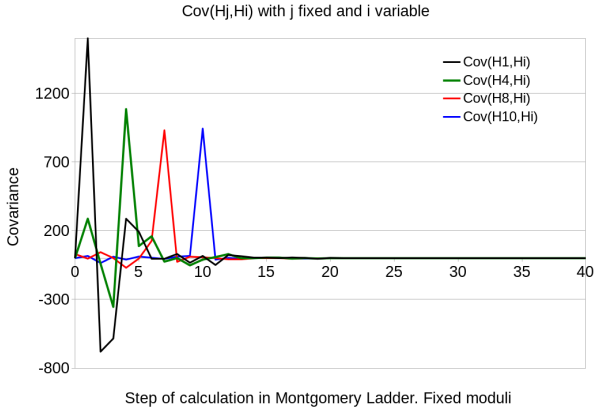

Fig. 3. ECC112 RNS10, $Cov(H_j, H_i)_{j=1,4,8,10}$.Fixed moduli

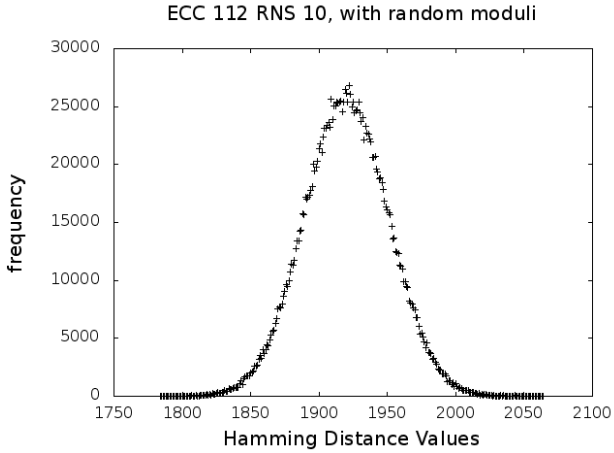

Fig. 4. Frequency of H_{10}, 2×10^6 computations.

We simulate with $S = 8 \times 10^6$ or $S = 10^6$ in order to have a sufficiently accurate results to compute TVI. We use the random number generator proposed in [35]. This choice is appropriate computationally and statistically for Monte Carlo simulation.

According to Figure 2, randomizing moduli reduces effectively TVI for all Hamming distances. Also, according to Figure 2, we see clearly that TVI almost vanishes for Hamming distances of a rank bigger than 10 which confirms property α). This observation can be explained by the fact that the first ~ 10 Hamming distances depend strongly on the first ~ 10 bits of the key. Because a large choice of combinations of bits can produce the same value on each $\{H_i\}_{i>10}$, the dependence between $\{H_i\}_{i>10}$ and the key is reduced significantly.

Regarding property β), we approximate the covariance of each couple of Hamming distances with a Monte Carlo simulation on keys for a fixed choice of moduli:

$$\mathrm{Cov}(H_i, H_j) \approx \frac{1}{S}\sum_{l=1}^{S} H_i(K^l)H_j(K^l) - \left[\frac{1}{S}\sum_{l=1}^{S} H_i(K^l)\right]^2.$$

We get the results presented in Figure 3 for the covariance H_1, H_4, H_8 and H_{10} with the other Hamming distances. In Figure 3, the fact that $|\mathrm{Cov}(H_i, H_{i\pm l})|_{l\geq 0}$ decreases with respect to the gap l is due to the difference in term of bits that separates H_i and $H_{i\pm l}$.

The importance of the covariance comes from the property γ. Indeed, in a multivariate Gaussian vector, each two coordinates are independent if and only if their covariance is equal to zero. A Chi Square test does not disapprove the Gaussian distribution of each Hamming distance H_i when moduli are randomized. We also present in Figure 4 an histogram associated to H_{10} that shows a bell-shaped distribution. This property does not contradict the assumption that the whole vector H is Gaussian. For more mathematical details on multivariate Normal distribution, we refer the reader to [34].

3.2 Unreliable CPA and inconsistent DPA

The essential result of Section 3.1 is that exploitable information leakage should be based on the first ~ 10 computation steps. Once the bits associated to some of these steps are known, one should fix them to continue with the following ~ 10 computation steps and so on. This conditional strategy (conditioning on the bits found) not only uses the marginal information of each step but must use the cross-information of the ~ 10 successive steps. Indeed, according to Figure 2, the first ~ 5 Hamming distances have almost the same strength of dependence on the secret key K and we waste information if we use only the marginal distributions. Nev-

ertheless, CPA and DPA are not conceived to take advantage of this cross-information.

As we study the effect of the randomized moduli on Hamming distances. Our approach can be justified by the leakage models introduced in [1] between the power consumption and the Hamming distances. We explore the link between the randomization of representation and Hamming distances, independently of the implementation support.

3.2.1 Methodology

We denote $K = \sum_{l=0}^{d-1} b_l 2^{d-1-l}$ the key of d bits to guess. Assuming that we know the j first bits, we denote $K'_j = \sum_{l=0}^{j-1} b_l 2^{d-1-l} + \sum_{l=j}^{d-1} 2^{d-1-l}$ a key of d bits which begins like K and have only ones after. At each step i, $H_i(K, C^l)$ is an observation associated to the real key K and $H_i(K'_j, C^{l+S})$ is the simulation associated to the guessed one K'_j. The term $+S$ in C^{l+S} expresses the independence between the moduli configurations $\{C^l\}_{1\leq l\leq S}$ and $\{C^{l+S}\}_{1\leq l\leq S}$. This results from the independece of the whole sequence $C = \{C^1, C^2, \ldots, C^S, C^{S+1}, \ldots, C^{2S}\}$.

Using the discrepancy induced by $\left\{H(K, C^l)\right\}_{l=1}^{S}$ and by $\left\{H(K'_0, C^{l+S})\right\}_{l=1}^{S}$, we determine the position j_1 of the first zero in the binary expansion of K. Then, we do the same for K and K'_{j_1} which provides the position j_2 of the second zero and so on till we find K. For example, when $K = 11101101110_2$:

- We get $j_1 = 3$ from $K = 111\,0\,1101110_2$ and $K'_0 = 11111111111_2$.
- We get $j_2 = 6$ from $K = 111011\,0\,1110_2$ and $K'_3 = 11101111111_2$.
- We get $j_3 = 10$ from $K = 1110110111\,0\,_2$ to $K'_6 = 11101101111_2$.

Formerly speaking, denoting $H_i^l(K) = H_i\left(K, C^l\right)$, $H_i^{l+S}\left(K'_{j_{p-1}}\right) = H_i\left(K'_{j_{p-1}}, C^{l+S}\right)$ and setting $\min(\emptyset) = d$ then

$$j_0 = 0 \tag{4}$$
$$j_p = \min\left\{i > j_{p-1}, \left| f_i\left(\left\{H_i^l(K), H_i^{l+S}\left(K'_{j_{p-1}}\right)\right\}_{l=1}^{S}\right)\right| > \mathcal{T}\right\}$$

and the key is recovered when $j_p = d$ which means that $K = K'_{j_{p-1}}$. $\mathcal{T} > 0$ is the distinguishing threshold and f_i is a distinguisher defined in Section 3.2.2 for CPA and DPA. In the worst case ($K = 2^{d-1}$), this method requires to check $d-1$ hypothesis instead of checking 2^{d-1} hypothesis.

With this approach, we search the number of moduli needed to make the key K indistinguishable from K'_j for a DPA or CPA. In classic DPA [1], [5], for a key hypothesis and a set of messages, messages are classified in two sets: those that give a high Hamming weight and those that give low Hamming weight. Thus, the leaks are classified and the difference of their average produces a peak if the hypothesis is correct. The randomization of moduli does not allow to classify Hamming distances with respect to messages.

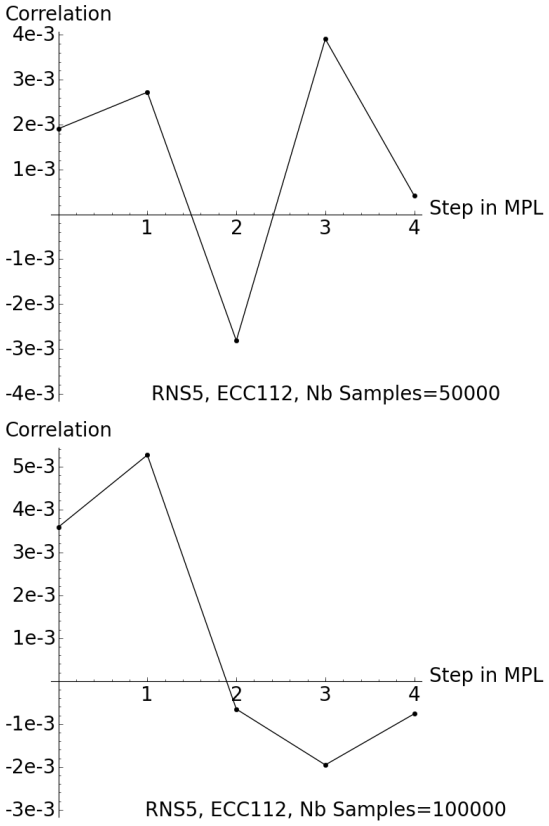

Fig. 5. RNS5, Correlation between 0xdeeefbf7 and 0xffffffff, 50000 and 100000 traces. Nothing appears at bit 2 and the correlations are too small.

Consequently, the average on messages has been replaced by the average on configurations of moduli:

$$\overline{H}_i(K, C) = \frac{1}{S}\sum_{l=1}^{S} H_i(K, C^l) \quad \text{and}$$
$$\overline{H}_i(K'_j, C) = \frac{1}{S}\sum_{l=1}^{S} H_i(K'_j, C^{l+S}). \tag{5}$$

.

We point out that we are not allowed to classify Hamming distances with respect to moduli configurations. Indeed, an attacker does not control the moduli configuration of the system.

3.2.2 Results

A CPA on Hamming distances is based on the correlation ξ_i (6) that exists at step i between observations $H_i(K, C^l)$ on the real key K and simulations $H_i(K'_j, C^{l+S})$ on the guessed one K'_j which yields

$$\xi_i = f_i\left(\left\{H_i^l(K), H_i^{l+S}\left(K'_{j_{p-1}}\right)\right\}_{l=1}^{S}\right)$$
$$= \frac{\sum_{l=1}^{S}\left[H_i(K, C^l) - \overline{H}_i(K, C)\right]\left[H_i(K'_j, C^{l+S}) - \overline{H}_i(K'_j, C)\right]}{\sqrt{\sum_{l_1=1}^{S}\left[H_i(K, C^{l_1}) - \overline{H}_i(K, C)\right]^2 \sum_{l_2=1}^{S}\left[H_i(K'_j, C^{l_2+S}) - \overline{H}_i(K'_j, C)\right]^2}}. \tag{6}$$

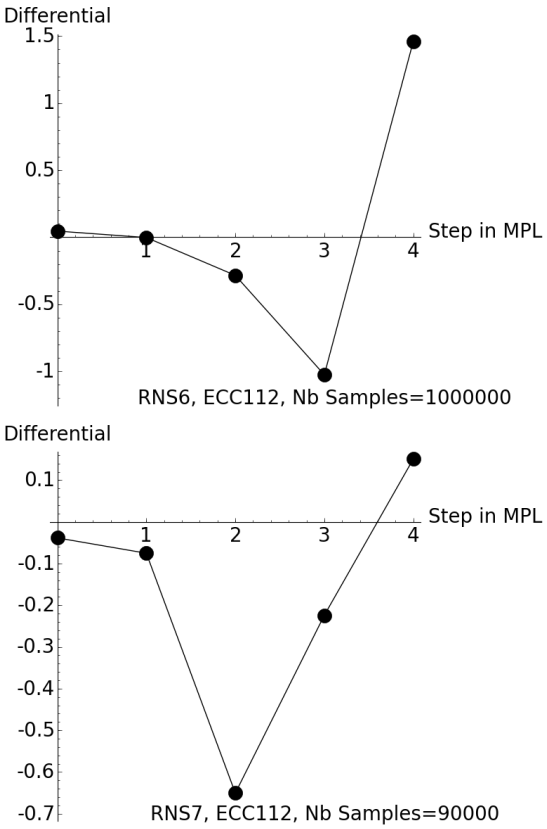

Fig. 6. RNS6 and RNS7: Differencial between 0xffffffff and 0xdeeefbf7 with respectively 1000000 and 90000 traces. A jump appears for the bit 2 for RNS7 as we expected but the jump is not obvious for RNS6 and we needed more traces to have this little jump

The independence of the sequence $\{C^l\}_{1\leq l\leq 2S}$ makes the use of CPA completely irrelevant as shown in Figure 5. We use $K = 0$xdeeefbf7 as a model key and $K'_0 = 0$xffffffff is used as a distinguisher.

Regarding the DPA based on Hamming distances, the differential value to evaluate the distinction is given at each step i of the MPL by

$$\begin{aligned}\text{DIFF}_i &= f_i\left(\left\{H_i^l\left(K\right), H_i^{l+S}\left(K'_{j_{p-1}}\right)\right\}_{l=1}^{S}\right)\\ &= \overline{H}_i(K,C) - \overline{H}_i(K'_j,C). \end{aligned} \quad (7)$$

Unlike for CPA, the lag $+S$ involved in the expression of DPA_i is less disturbing because, by the law of large numbers, $\frac{1}{S}\sum_{k=1}^{S} H_i(K'_j, C^{k+S})$ and $\frac{1}{S}\sum_{k=1}^{S} H_i(K'_j, C^{k})$ converge to the same value as $S \to \infty$. Consequently, DPA can be used to exploit information leaks when S is big enough. The fact that we do not know how big S must be (except doing very coarse domination) makes DPA difficult to use. Indeed, as shown in Figure 6, sometimes we even need a bigger S for an RNS with less randomized moduli!

3.3 Further analysis: Second order DPA and MIA

Like in Section 3.2, we focus only on Hamming distances. Generally, DPA and MIA are used when the information leakage is observed with a hardware noise. In our study, the noise is due to the RNS randomization that reduces the dependence between the secret K and Hamming distances.

3.3.1 Second order DPA

We use a simulation of second order DPA (2ODPA) as follows:

$$\begin{aligned} 2ODPA_0 &= DIFF_0\\ 2ODPA_i &= DIFF_{i+1} - DIFF_i \text{ if } i > 0 \end{aligned}$$

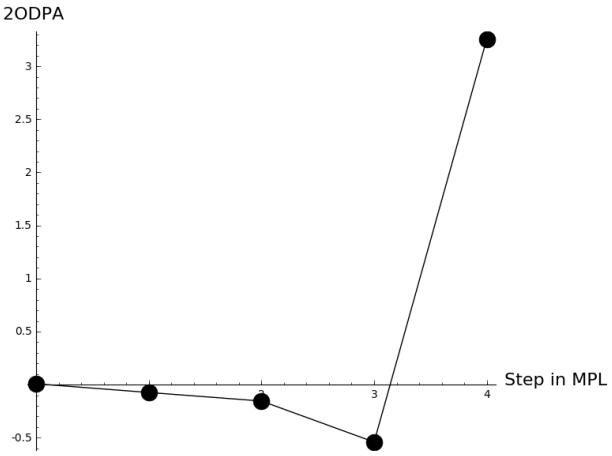

Fig. 7. RNS6: Second order DPA between 0xffffffff and 0xdeeefbf7 with 1000000 traces.

According to Figure 7, we see that the second order DPA on Hamming distances does not improve the results of DPA presented in Figure 6 (RNS6). This can be explained by the absence of a heterogeneity in the code between two steps of computations and thus between two successive Hamming distances. Moreover, the second order DPA defined in [6] (Proposition 2) involves marginal information since it averages on the realizations of one random variable defined as the difference between the power consumptions of two successive steps.

3.3.2 Mutual Information Analysis

Introduced in Section 3.1, the TVI computation involved the estimation of $P_1 = P\left(H_i \in \mathcal{H}_j^i\right)$ and of $P_2 = P\left(H_i \in \mathcal{H}_j^i | K \in I_k\right)$. These quantities have unbiased estimators with a Mean Square Error MSE (cf. [36]) equal to $\sigma^2\left(1_{H_i\in\mathcal{H}_j^i}\right)/S$ and to $\sigma^2\left(1_{H_i\in\mathcal{H}_j^i}|K\in I_k\right)/S$ respectively, where $\sigma^2(\cdot)$ is the variance and $\sigma^2(\cdot|K\in I_k)$ is the conditional variance. The approximation of P_1 and P_2 required many traces to reach an error smaller than 10%; more than 200000 traces in RNS10 for the first 10 Hamming distances, and more than 1000000 for 112 Hamming distances. Despite these relatively heavy computations when we randomized moduli, TVI was not able to measure the dependence structure. Randomized moduli TVI does not distinguish between the distribution of H_i and of H_i conditionally on $K \in I_k$, i.e between P_1 and P_2.

The MIA distinguisher (cf. [8]) is based on the mutual information of two random variables $(X, Y) \in \mathcal{X} \times \mathcal{Y}$: $I(X,Y) =$

$$\sum_{x\in\mathcal{X}} P(X=x) \sum_{y\in\mathcal{Y}} P(Y=y|X=x) \log\left(\frac{P(Y=y|X=x)}{P(Y=y)}\right).$$

In our case, using the subdivision described in Appendix C, one has to approximate:

$$\sum_{k=0}^{2^{p'}-1} P(K \in I_k) \sum_{j=0}^{q-1} P(H_i \in \mathcal{H}_j^i | K \in I_k) \log\left(\frac{P(H_i \in \mathcal{H}_j^i | K \in I_k)}{P(H_i \in \mathcal{H}_j^i)}\right).$$

Therefore to apply MIA, we have to compute $\log\left(P\left(H_i \in \mathcal{H}_j^i\right)\right)$ and $\log\left(P\left(H_i \in \mathcal{H}_j^i | K \in I_k\right)\right)$ that have biased Monte Carlo estimators as pointed out in Appendix B. The quality of these biased estimators can be measured with MSE. For example, according to equality (15) in Appendix D, the MSE of $\log\left(P\left(H_i \in \mathcal{H}_j^i\right)\right)$ is approximately $\frac{\sigma^2\left(\mathbf{1}_{\{H_i \in \mathcal{H}_j^i\}}\right)}{SP^2(H_i \in \mathcal{H}_j^i)}$. We notice that is divided by the number $P^2\left(H_i \in \mathcal{H}_j^i\right) << 1$. The logarithm increases the distances but amplifies significantly the variance; it becomes difficult to estimate. This makes the use of MIA even harder than TVI to distinguish the dependence on the key K. Thanks to the Gaussian assumption on the distribution of Hamming distances, MLE distinguisher requires less computations.

4 Evaluation with Maximum Likelihood Estimator (MLE)

4.1 Gaussian model and MLE to evaluate RNS randomization

Applying the Maximum Likelihood Estimator (MLE) [37], used for the Template attack [7], on Hamming distances provides better results than DPA. MLE requires a learning phase that precomputes the mean vector and the covariance matrix $(m^{k,i}, \Gamma_{k,i})$ of $H^i = (H_0, ..., H_i)^T$. Given the value of the secret $K = k$, for each $i \in \{0, ..., d-1\}$, we suppose $H^i = (H_0, ..., H_i)^T$ has a multivariate normal distribution with density

$$p_{k,i}(\mathbf{x}^i) = \frac{\exp\left(-\frac{(\mathbf{x}^i - m^{k,i})^T \Gamma_{k,i}^{-1} (\mathbf{x}^i - m^{k,i})}{2}\right)}{\left(\sqrt{2\pi}\right)^{i+1} \sqrt{\det(\Gamma_{k,i})}}, \quad (8)$$

where $\mathbf{x}^i = (x_0, ..., x_i)^T \in \mathbb{R}^{i+1}$.

The MLE analysis that we perform can be summarized in the following steps: for $i = 0, 1, ..., d-1$,

- Either we compute the exact value of $(m^{k,i}, \Gamma_{k,i})$ with all $\frac{(2n)!}{n!^2}$ combinations in RNSn [1], or we estimate $(m^{k,i}, \Gamma_{k,i})$ using fewer combinations simulated with a Monte Carlo method. This is the learning phase.
- We observe S realizations $\mathbf{x}_{1 \le j \le S}^i$ of $H^i = (H_0, ..., H_i)^T$.
- We select the secret $K = k$ that maximizes $\prod_{j=1}^{S} p_{k,i}(\mathbf{x}_j^i)$. Due to the floating-point precision, it is better to select the value that maximizes $\sum_{j=1}^{S} \log(p_{k,i}(\mathbf{x}_j^i))$. This is the estimation phase.

1. This is our default choice in all the following figures and tables.

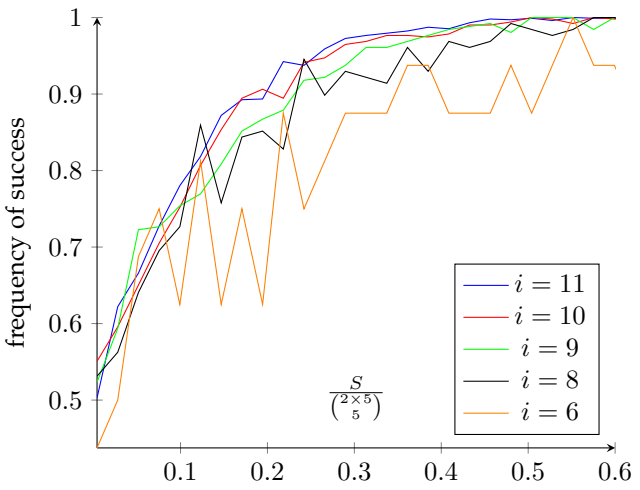

Fig. 8. Frequency of success to find the first bit of the key with MLE on ECC 112 in RNS5 ($n = 5$) for different size $(m^{k,i}, \Gamma_{k,i})$ of the template

n	5	6	7	8	9	10	11
β	2.37	2.38	2.30	2.40	2.31	2.24	2.33

TABLE 1
slope of the linear regression

- We quantify the value of S that makes the information leakage exploitable.

We point out that this MLE analysis was possible largely to our GPU (Graphics Processing Unit) implementation that provides sufficient throughput to perform this study, especially for Figure 9.

Figure 8 shows that there is a remarkable information leak in the first $\sim$ 10 successive Hamming distances. In particular, we see that there is not substantial amelioration between $i = 9$ and $i = 11$. This confirms what was already explained with TVI in Section 3.1.

According to Figure 9, with $S = \frac{(2n)!}{n!^2}$ we get almost 100% of success for ten bits. It is quite remarkable to see that S should be of the order of $(2n)!/(n!)^2$ that is the number of combinations in a RNSn. We point out that we obtain similar results with other curves including: Edwards 25519 [20] and ECCsecp256r1 [21].

Denoting $H_K^i = (H_{0,K}, ..., H_{i,K})$, a sequence of Hamming distances given by a key K, MLE shows that distribution of each H_K^i is completely distinguishable with its couple of mean and covariance matrix. Moreover, our study shows that the distribution differences reduce with an increase of the number of moduli till reaching the order of $\binom{2n}{n} = (2n)!/(n!)^2$.

Let us denote f the frequency of success. Figure 9 shows that f depends on $\frac{S}{\binom{2n}{n}}$ and is almost linear with respect to $x = \frac{S}{\binom{2n}{n}}$ for $0 < x < 0.3$. Moreover, as we see in Table 1, the value of the slope barely changes for $n \in \{5, 6, 7, 8, 9, 10, 11\}$. Consequently, one can set $f \approx \beta \frac{S}{\binom{2n}{n}}$ with $2.2 < \beta < 2.4$ and $0 \le \frac{S}{\binom{2n}{n}} < 0.3$.

To protect the system, the frequency of success f should

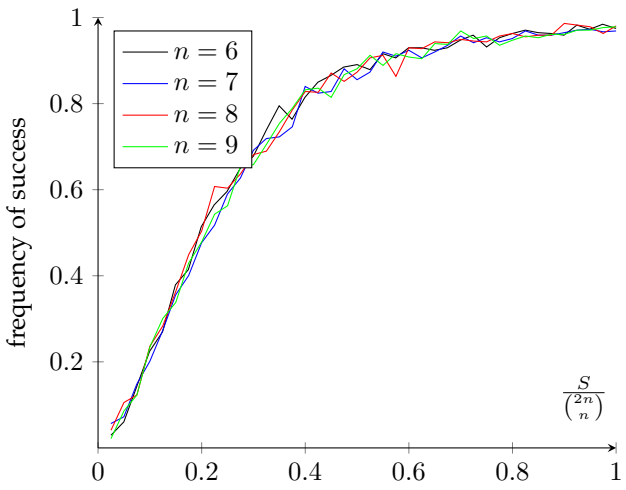

Fig. 9. Frequency of success to find a 10-bit key with MLE for different RNSn on ECC 112 Montgomery in Jacobian coordinates.

S	p_t 0.1	0.2	0.3	0.4	0.5	0.6
2^{10}	9	8	8	8	8	8
2^{15}	12	11	11	11	10	10
2^{20}	14	14	13	13	13	13
2^{25}	17	16	16	16	16	15
2^{30}	19	19	18	18	18	18
2^{35}	22	21	21	21	21	20
2^{40}	24	24	24	23	23	23
2^{45}	27	26	26	26	26	26
2^{50}	29	29	29	28	28	28

TABLE 2
Minimum n to protect 10 first bits of the key till S traces with a success frequency smaller than p_t.

not be greater than a frequency threshold p_t:

$$\beta \frac{S}{\binom{2n}{n}} < p_t. \tag{9}$$

If we extrapolate the result for $n > 10$, we obtain, in Table 2, the minimum number of 32-bit-moduli to fulfill the previous condition for the first 10 bits of the key. If we want to protect the whole key from a conditional strategy (cf. Section 3.1), we must consider another condition:

$$\beta \frac{S}{\binom{2n}{n}} < p_t^{\frac{9}{\#ECC-1}}, \tag{10}$$

knowing the value of the first bit and $\#ECC$ is the bit length of the cardinality of the ECC. Thus, Table 3 gives the minimum n to protect the whole key. This minimum is necessarily bigger than the smallest n required to implement an ECC of $\#ECC$ bits based on 32-bit moduli.

4.2 Additional considerations to choose the number of moduli

4.2.1 *From which level we loose the random behaviour?*

Let us denote the null hypothesis
$\mathbf{Hyp_0}$: "We obtain 10 bits of the key with a probability equal to 2^{-9}"

$S \times \frac{\#ECC-1}{9}$	#ECC 112	256	384	521
2^{10}	6	9	13	18
2^{15}	8	9	13	18
2^{20}	11	10	13	18
2^{25}	13	13	13	18
2^{30}	16	15	15	18
2^{35}	19	18	18	18
2^{40}	21	20	20	20
2^{45}	24	23	23	22
2^{50}	26	26	25	25

TABLE 3
Minimum n to protect the whole key till $S \times \frac{\#ECC-1}{9}$ traces from the target key: $p_t = 0.1$.

We calculate the 95% prediction interval with $p = 2^{-9}$:

$$\mathcal{I}_p = \left[p - 1.96\sqrt{\frac{p(1-p)}{SE}}; p + 1.96\sqrt{\frac{p(1-p)}{SE}} \right].$$

SE is a sample size. If $f \in \mathcal{I}_p$, we do not reject $\mathbf{Hyp_0}$ otherwise we reject $\mathbf{Hyp_0}$.

We can notice in Table 4 that we have to use $n > 7$ to avoid an attack with a single trace. This confirms the suggestion of [19].

n	5	6	7	8	9	10	11
S	1	1	1	5	7	16	130

TABLE 4
Minimum size to reject $\mathbf{Hyp_0}$ with a sample size $SE = 32256$ ($error < 0.1\%$ for a 95% prediction interval)

4.2.2 *The learning phase costs more than the estimation phase even with Monte Carlo.*

We used an exact value of $(m^{k,10}, \Gamma_{k,10})$ to set the template, so we needed $2^9 \times \binom{2n}{n}$ traces. Comparing this value to Table 3, we notice that the learning phase costs more than the estimation phase. Thus, the former determines the number of moduli needed for protection. Even with Monte Carlo, the success decreases by half when we use 80% of $2^9 \times \binom{2n}{n}$ traces to set the template (Fig. 10). If the attacker chooses to reduce computations via Monte Carlo, he decreases significantly his chance to find the secret.

5 Conclusion and Future Work

In this work, we used MLE that takes advantage of the cross-information in the Hamming distances. This provides an efficient evaluation of the information leakage even for cryptographic systems protected by RNS randomization. We showed however that this efficiency decreases as the number of needed observations is of the order of $(2n)!/(n!)^2$. With this evaluation, we gave an estimation of the number of moduli to protect efficiently a system against a template attack. For example, since RNS12 requires $2^{30.36}$ computations to construct a good template, it provides a good protection for the actual standard of ECC like Edwards curve 25519 [20] and ECCsecp256r1 [21]. A RNS15 protects also against the estimation phase at the same level and in any case, it is better to have $n > 7$ to have a correct random behaviour against a single trace attack.

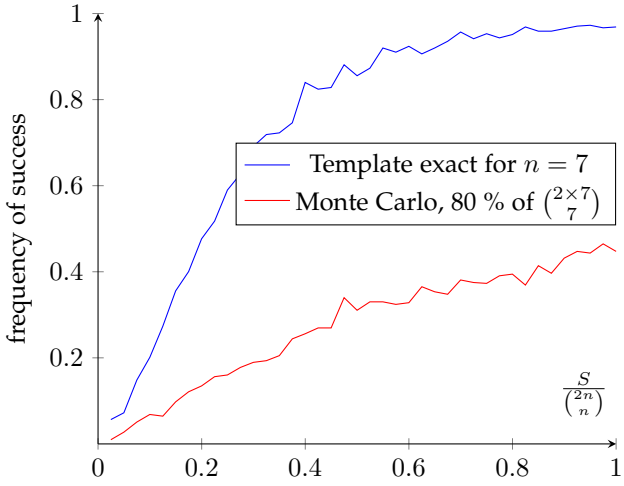

Fig. 10. Frequency of success to find a 10-bits key with MLE on ECC 112. Comparison between Template exact and Monte Carlo.

As a future work, we would like to explore if there exists a method to decrease the number of moduli for a given level of randomness in order to reduce calculations. Furthermore, it would be good to prove theoretically the behaviour obtained in Figure 9 or at least establish an upper bound curve. It could be also interesting to do this same study when the template phase is evaluated with a Monte Carlo method and/or investigate variance reduction techniques.

Acknowledgements

This work was funded by project ARRAND (ANR-15-CE39-0002-01).

References

[1] S. Mangard, E. Oswald, and T. Popp, Power analysis attacks: Revealing the secrets of smart cards. Springer Science & Business Media, 2008, vol. 31.

[2] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman, "Mining your ps and qs: Detection of widespread weak keys in network devices," in Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12), 2012, pp. 205–220.

[3] F. Benhamouda, C. Chevalier, A. Thillard, and D. Vergnaud, "Easing coppersmith methods using analytic combinatorics: applications to public-key cryptography with weak pseudorandomness," in IACR International Workshop on Public Key Cryptography. Springer, 2016, pp. 36–66.

[4] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2004, pp. 16–29.

[5] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, "Introduction to differential power analysis," Journal of Cryptographic Engineering, vol. 1, no. 1, pp. 5–27, 2011.

[6] T. S. Messerges, "Using second-order power analysis to attack dpa resistant software," in International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2000, pp. 238–251.

[7] S. Chari, J. R. Rao, and P. Rohatgi, "Template attacks," in International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2002, pp. 13–28.

[8] L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon, "Mutual information analysis: a comprehensive study," Journal of Cryptology, vol. 24, no. 2, pp. 269–291, 2011.

[9] J. Fan and I. Verbauwhede, "An updated survey on secure ecc implementations: Attacks, countermeasures and cost," in Cryptography and Security: From Theory to Applications. Springer, 2012, pp. 265–282.

[10] L. Batina, Ł. Chmielewski, L. Papachristodoulou, P. Schwabe, and M. Tunstall, "Online template attacks," in International Conference in Cryptology in India. Springer, 2014, pp. 21–36.

[11] M. Joye, "Smart-card implementation of elliptic curve cryptography and dpa-type attacks," in Smart card research and advanced applications VI. Springer, 2004, pp. 115–125.

[12] M. Dugardin, L. Papachristodoulou, Z. Najm, L. Batina, J.-L. Danger, and S. Guilley, "Dismantling real-world ecc with horizontal and vertical template attacks," in International Workshop on Constructive Side-Channel Analysis and Secure Design. Springer, 2016, pp. 88–108.

[13] S. Antão, J.-C. Bajard, and L. Sousa, "Rns-based elliptic curve point multiplication for massive parallel architectures," The Computer Journal, vol. 55, no. 5, pp. 629–647, 2011.

[14] N. Guillermin, "A high speed coprocessor for elliptic curve scalar multiplications over $\mathbb{F}_p$," in International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2010, pp. 48–64.

[15] J.-C. Bajard, L. Imbert, P.-Y. Liardet, and Y. Teglia, "Leak resistant arithmetic," in International Workshop on Cryptographic Hardware and Embedded Systems. Springer, 2004, pp. 62–75.

[16] G. Perin, L. Imbert, L. Torres, and P. Maurine, "Attacking randomized exponentiations using unsupervised learning," in International Workshop on Constructive Side-Channel Analysis and Secure Design. Springer, 2014, pp. 144–160.

[17] A. Lesavourey, C. Negre, and T. Plantard, "Efficient leak resistant modular exponentiation in rns," in 2017 IEEE 24th Symposium on Computer Arithmetic (ARITH). IEEE, 2017, pp. 156–163.

[18] A. P. Fournaris, L. Papachristodoulou, and N. Sklavos, "Secure and efficient rns software implementation for elliptic curve cryptography," in 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, 2017, pp. 86–93.

[19] G. X. Yao, M. Stöttinger, R. C. Cheung, and S. A. Huss, "Side channel attacks and their low overhead countermeasures on residue number system multipliers," in Emerging Technology and Architecture for Big-data Analytics. Springer, 2017, pp. 137–158.

[20] D. J. Bernstein and T. Lange, "Faster addition and doubling on elliptic curves," in International Conference on the Theory and Application of Cryptology and Information Security. Springer, 2007, pp. 29–50.

[21] C. Research, "Version 1.0 and 2.0: Standards for efficient cryptography recommended elliptic curve domain parameters," 2000.

[22] P. L. Montgomery, "Modular multiplication without trial division," Mathematics of computation, vol. 44, no. 170, pp. 519–521, 1985.

[23] J. A. Ambrose, H. Pettenghi, and L. Sousa, "Darns: a randomized multi-modulo rns architecture for double-and-add in ecc to prevent power analysis side channel attacks," in 2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2013, pp. 620–625.

[24] J.-C. Bajard, L.-S. Didier, and P. Kornerup, "Modular multiplication and base extensions in residue number systems," in Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001. IEEE, 2001, pp. 59–65.

[25] V. G. Martínez, L. H. Encinas, and C. S. Ávila, "A survey of the elliptic curve integrated encryption scheme," ratio, vol. 80, no. 1024, pp. 160–223, 2010.

[26] A. B. Association et al., "Public key cryptography for the financial services industry: The elliptic curve digital signature algorithm (ecdsa)," ANSI X9, pp. 62–1998, 2005. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf

[27] T. Izu and T. Takagi, "A fast parallel elliptic curve multiplication resistant against side channel attacks," in International Workshop on Public Key Cryptography. Springer, 2002, pp. 280–296.

[28] J.-C. Bajard and T. Plantard, "Rns bases and conversions," in Advanced Signal Processing Algorithms, Architectures, and Implementations XIV, vol. 5559. International Society for Optics and Photonics, 2004, pp. 60–70.

[29] N. Meloni, "New point addition formulae for ecc applications," in International Workshop on the Arithmetic of Finite Fields. Springer, 2007, pp. 189–201.

[30] H. Cohen, G. Frey, R. Avanzi, C. Doche, T. Lange, K. Nguyen, and F. Vercauteren, Handbook of elliptic and hyperelliptic curve cryptography. Chapman and Hall/CRC, 2005.
[31] D. Hankerson, A. J. Menezes, and S. Vanstone, "Guide to elliptic curve cryptography," Computing Reviews, vol. 46, no. 1, p. 13, 2005.
[32] A. R. . al., "A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications." NIST Special Publication 800-22rev1a, 2010.
[33] D. A. Levin and Y. Peres, Markov chains and mixing times. American Mathematical Soc., 2017, vol. 107.
[34] J. Jacod and P. Protter, Probability essentials. Springer Science & Business Media, 2012.
[35] P. L'ecuyer, R. Simard, E. J. Chen, and W. D. Kelton, "An object-oriented random-number package with many long streams and substreams," Operations research, vol. 50, no. 6, pp. 1073–1075, 2002.
[36] M. H. DeGroot and M. J. Schervish, Probability and statistics. Pearson Education, 2012.
[37] F. Scholz, "Maximum likelihood estimation," Encyclopedia of statistical sciences, 2004.
[38] S. Kawamura, M. Koike, F. Sano, and A. Shimbo, "Cox-rower architecture for fast parallel montgomery multiplication," in International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 2000, pp. 523–538.
[39] A. Shenoy and R. Kumaresan, "Fast base extension using a redundant modulus in rns," IEEE Transactions on Computers, vol. 38, no. 2, pp. 292–297, 1989.

Jérôme Courtois has a diploma from Centrale Nantes in 1992. After twenty years of teaching as "Agrégé" in mathematics , he decided to study computer science then received a Master degrees in computer science from Paris Pierre and Marie Curie University in 2016. He is preparing is Ph.D. with Jean-Claude Bajard and Lokman Abbas-Turki about Randomized Arithmetics.

Lokman Abbas-Turki is associate professor in mathematics at LPSM with a strong expertise in GPU programming and high performance computing. He is specialized in numerical probability and quantitative finance. He received his PhD degree in applied mathematics from Universit Paris-Est and a Master degree in signal processing from Suplec. Lokman won a CUDA Teaching Center label for his CUDA course taught in not less than 12 Master programs at different institutions including Sorbonne Université, ENSAE, TU Berlin, ICM Edinburgh, Polytech Sorbonne and EISTI.

Jean-Claude Bajard received the PhD degree in computer science from the Ecole Normale Superieure de Lyon (ENS), France, in 1993. He taught mathematics in high school from 1979 to 1990 and served as a research and teaching assistant at the ENS until 1993. From 1994 to 1999, he was an assistant professor at the Universite de Provence, Marseille, France. From 1999 to 2009, he was a professor at the University Montpellier 2, and member of the LIRMM. He is currently a professor at the Sorbonne Universite, Paris, France, and a member of the LIP 6. His research interests include computer arithmetic and cryptography.