

Evaluation of Resilience of randomized RNS implementation

Jérôme Courtois¹, Lokman Abbas-Turki², Jean-Claude Bajard¹

¹Sorbonne Universités, UPMC, CNRS, LIP6, Paris, France

²Laboratoire de Probabilités et Modèles Aléatoires, UMR 7599, UPMC

Abstract. Randomized moduli in Residue Number System (RNS) generate effectively large noise and make quite difficult to attack a secret key K from only few observations of Hamming distances $H = (H_0, \dots, H_{d-1})$ that result from the changes on the state variable. Since Hamming distances have gaussian distribution and most of the statistic tests, like NIST's ones, evaluate discrete and uniform distribution, we choose to use side-channel attacks as a tool in order to evaluate randomisation of Hamming distances. This paper analyses the resilience against Correlation Power Analysis (CPA), Differential Power Analysis (DPA) when the cryptographic system is protected against Simple Power Analysis (SPA) by a Montgomery Powering Ladder (MPL). While both analysis use only information on the current state, DPA Square crosses the information of all the states. We emphasize that DPA Square performs better than DPA and CPA and we show that the number of observations S needed to perform an attack increases with respect to the number of moduli n . For Elliptic Curves Cryptography (ECC) and using a Monte Carlo simulation, we conjecture that $S = O((2n)!/(n!)^2)$.

Keywords: RNS, moduli randomization, Monte Carlo, ECC, side channel attack, DPA, CPA, DPA Square

1 Introduction

Side channel attacks or faults attacks are among the most effective attacks on all implementations of cryptographic protocols. As sophisticated as the mathematical model used can be for proving the robustness, unprotected implementation could leak out some fundamental information. Among the most usual leakage exploitations, we find those establishing correlations between leakage consumption or electromagnetic radiation recorded during successive runs with the same secret. Leaks are recorded mainly due to variations in the level of the memory points from one to zero in the calculation. The commonly accepted model is based on the differences of Hamming weight of the successive execution states usually assumed proportional to power consumption [21]. The attacker makes a guess on the secret; with respect to the input values and he/she can determine what could be the state transition of the observed leakages using usually DPA

or CPA [8, 18] or other more recent attacks like second-order DPA [25], template attacks [9] and Mutual Information Analysis (MIA) [6]. For a survey on the different attacks and countermeasures, we refer the reader to [11].

The state transitions depend clearly on the data representation which should be also an assumption made by the attacker. The randomization with respect to an arithmetic system ensures that the computations involved in the execution use different representation for the variables from one execution to another, reducing significantly any prediction on the state transitions for an attacker. We focus in this paper on RNS representation based on the Chinese Remainder Theorem where each value is coded by its residues over a set of co-prime numbers which represents the base of the system.

Beyond the fact that RNS allows natural randomization, the arithmetic is completely independent from the chosen field. Therefore, the methods presented in the paper will be suitable for any cryptosystem such as RSA, ECC, Euclidean Lattice or others. RNS is also adapted to an increase in key sizes induced by the progress of the cryptanalysis and RNS computations can be efficiently parallelized [2]. For further benefits of RNS, we refer to [13].

In [4], the authors introduce a Leak Resistant Arithmetic (LRA) using RNS with randomization of moduli. The RNS system is very easy to randomize, especially since there is a great diversity of moduli. Moreover, with Montgomery multiplication algorithm [26], the Montgomery factor strengthen the random behaviour. 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 that can be detected in the consumption or in any other leak. As sketched on Figure 1, we distinguish two randomness sources given by both the configuration of moduli C and the key K .

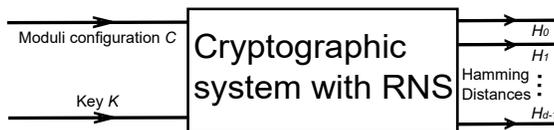


Fig. 1. Hamming distances with respect to randomness sources

$H = (H_0, \dots, H_{d-1})$ are the Hamming distances computed through the execution of the Montgomery Power Ladder (MPL) algorithm associated, in our examples of this paper, to ECC. When the implementation is public and is free of bugs, it is clear that the random vector $H = (H_0, \dots, H_{d-1})$ is completely measurable with respect to the couple of random variables (K, C) . Consequently, more the noise generated by C is important more the link between K and H will be difficult to establish. The perfect noise would be the one that mimics an independence between K and H . Without loss of generality, let us denote informally $L(H, K)$, $L(H|K)$, $L(H)$ and $L(K)$ respectively the probabilistic joint distribution of (H, K) , the conditional distribution of H knowing K and the

marginal distribution of H and K . The perfect noise must fulfill

$$L(H, K) := L(H|K)L(K) = L(H)L(K). \quad (1)$$

or equivalently $L(H|K) = L(H)$ meaning that K must not provide any information on H . Although 1 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 $H = (H_0, \dots, H_{d-1})$ 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 the number of moduli n randomized in the RNS representation, denoted also by $\text{RNS}n$. Nevertheless, because studying $L(H|K)$ for the whole vector H is computationally barely possible, we develop a strategy based on few Hamming distances ~ 10 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 DPA Square use a cross-information based on ~ 10 Hamming distances. This DPA Square notion extends naturally the DPA when we assume that $H = (H_0, \dots, H_{d-1})$ has a multivariate Normal distribution. DPA Square has the monotonicity benefit since the size S of the sample that makes an attack possible increases with respect to n . Therefore, our main contribution can be summarized in the following points:

- 1) We show that the cryptographic system has to be resilient with respect to cross-information attack and not only attacks that use marginal 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 present the DPA Square with an asymptotic error which can be used as a threshold for an attack.
- 5) Unlike DPA, the size S to perform an attack with DPA Square is monotonous.

The contributions 1) to 5) are general and can be applied for other cryptographic protections. We make the strong assumption that the attacker has access to the Hamming distances. The noise created by the random moduli would add to the hardware noise in real applications. Our work studies the impact of moduli randomization on Hamming distances independent from hardware aspects. 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 Edward curves of 255 bits [7]. We focus on ECC as it is well suited for the evaluation of moduli randomization countermeasure because the arithmetic is dominant and the DPA is efficient. Indeed, all the probability tools like: Total variation, covariance matrix, asymptotic error of Monte Carlo can be reused in the same fashion for other systems. We point out also that our work is different from [27] 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.

The rest of this paper is arranged as follows. In Section 2, we briefly describe the moduli randomization of the RNS representation in the Montgomery algorithm applied for ECC. Section 3 explains the main reasons why a resilience of a system should rather focus on ~ 10 successive Hamming distances. Section 4 details our attack based on DPA Square and studies the size S of observations needed to achieve it.

There is not material implementation in this study because it is not useful or even counterproductive because we want to know the added noise by the randomization of moduli.

2 MPL using RNS representation applied to ECC

In Section 2.1, we explain briefly the randomization technic based on RNS representation for Montgomery multiplication. Then, Section 2.2 clarifies the way the Hamming distances are computed through the successive steps of the MPL.

2.1 Montgomery for RNS modular multiplication

In [26], 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 [1]. We summarize this method with a presentation that is quite similar to the one in [3].

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

$$X = \left(\sum_{i=1}^n x_i |M_i|_{m_i}^{-1} M_i \right) \bmod M \quad (2)$$

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

Let $\tilde{\mathcal{B}}_n = \{\tilde{m}_1, \dots, \tilde{m}_n\}$ be another RNS basis of pairwise coprime moduli that are also coprime with \mathcal{B}_n i.e. m_i and \tilde{m}_j are coprime for each $i \in \llbracket 1, n \rrbracket$ and $j \in \llbracket 1, n \rrbracket$. For a number X that is strictly smaller than $\tilde{M} = \prod_{i=1}^n \tilde{m}_i$, we use the notation $\{\tilde{x}_1, \dots, \tilde{x}_n\}$ for the decomposition of X on $\tilde{\mathcal{B}}_n$. Using these notations as well as the standard definition of the usual RNS operations (addition $+_{RNS}$, multiplication \times_{RNS} , opposite $(-X)_{RNS}$ and the division $/_{RNS}$ explained in [3]), Algorithm 1 presents the modular multiplication.

Algorithm 1 RNS n modular multiplication

Input A residue base $\mathcal{B}_n = \{m_1, \dots, m_n\}$ where $M = \prod_{i=1}^n m_i$

A residue base $\tilde{\mathcal{B}}_n = \{\tilde{m}_1, \dots, \tilde{m}_n\}$ where $\tilde{M} = \prod_{i=1}^n \tilde{m}_i$ with $\gcd(M, \tilde{M}) = 1$

A modulus N expressed in \mathcal{B}_n and $\tilde{\mathcal{B}}_n$ with $\gcd(N, M) = 1$, $\gcd(N, \tilde{M}) = 1$,
 $0 < (n+2)^2 N < M$ and $0 < (n+2)^2 N < \tilde{M}$

An Integer A expressed in \mathcal{B}_n and $\tilde{\mathcal{B}}_n$

An Integer B expressed in \mathcal{B}_n and $\tilde{\mathcal{B}}_n$ with $AB < NM$

Output An integer R expressed in \mathcal{B}_n and $\tilde{\mathcal{B}}_n$ such that $R \bmod N = ABM^{-1} \bmod N$

Procedure

$Q \leftarrow ((-(A \times_{RNS} B))_{RNS}) /_{RNS} N$ in base \mathcal{B}_n

Extension 1 Of Q from \mathcal{B}_n to $\tilde{\mathcal{B}}_n$

$R \leftarrow (A \times_{RNS} B +_{RNS} Q \times_{RNS} N) /_{RNS} M$ in base $\tilde{\mathcal{B}}_n$

Extension 2 of R from $\tilde{\mathcal{B}}_n$ to \mathcal{B}_n

end Procedure

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 $AM \bmod N$. Once $MM \bmod N$ is known, this form can be obtained with Algorithm 1 applied to A and $MM \bmod N$ provided that we exchange \mathcal{B}_n and $\tilde{\mathcal{B}}_n$ since

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

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

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

We point out that precomputing $|M\tilde{M}|_N$ instead of $|M^2|_N$, as proposed in [3], is justified by the randomization procedure explained latter in this section.

For extension 1 in Algorithm 1, we use a raw method to prevent heavy computations due to mixed radix systems [5]. Thus, to extend in base $\tilde{\mathcal{B}}_n = \{\tilde{m}_1, \dots, \tilde{m}_n\}$, we calculate

$$\tilde{q}_j = \left| \sum_{i=1}^n q_i |M_i^{-1}|_{m_i} \Big|_{m_i} M_i \Big|_{\tilde{m}_j} \right| \quad \text{for each } j \in \llbracket 1, n \rrbracket \quad (3)$$

we obtain $\tilde{Q} = \sum_{i=1}^n q_i |M_i^{-1}|_{m_i} \Big|_{m_i} M_i = Q + \alpha \times M$ with $\alpha \in \llbracket 1, n \rrbracket$ (4)

To evaluate α , we use Shenoy-Kumaresan method [28, 3] for extension 2 in Algorithm 1. In contrast to Kawamura Extension [17], Shenoy-Kumaresan allows a larger choice of moduli it also involves an extra modulo m_x since

$$\alpha = \left| \left| \tilde{M}^{-1} \right|_{m_x} \left(\sum_{j=1}^n \tilde{x}_j \left| \tilde{M}_j^{-1} \right|_{\tilde{m}_j} \tilde{M}_j \right) \Big|_{m_x} - |X|_{m_x} \right|_{m_x}. \quad (5)$$

Subsequently, $x_i = |X|_{m_i}$ can be computed using

$$x_i = \left| \sum_{j=1}^n \left| \tilde{x}_j \left| \widetilde{M}_j^{-1} \right|_{\tilde{m}_j} \widetilde{M}_j \right|_{\tilde{m}_j} - |\alpha M|_{\tilde{m}_j} \right|_{m_i}. \quad (6)$$

The Shenoy-Kumaresan extension requires that N has to fulfill $(n+2)^2N < M$. The latter inequality with $AB < NM$ makes $R < (n+2)N$. To obtain $R < 2N$ at the very end of an ECC, we use mixed radix representation [5].

2.2 Measuring Hamming distances in our implementation

ECC is usually implemented as an asymmetric cryptographic algorithm in particular for the Integrated Encryption and Decryption Scheme [22, 30]. Alice encrypts a text with the public key of Bob. The cyphertext contains a point G of the elliptic curve. With his private key K , Bob calculates $[K]G$ to decrypt the cyphertext. ECC is more commonly implemented for the Diffie-Hellman protocol to exchange key via the network.

Before each ECC execution, we perform a random pick of n moduli $\{m_1, \dots, m_n\}$ among $\{\mu_1, \dots, \mu_{2n}\}$ for base \mathcal{B}_n and the remaining moduli set the base $\tilde{\mathcal{B}}_n$. This random choice is based on a standard drawing without replacement.

To compute $[K]G$ on an elliptic curve and protect against Simple Power Analysis (SPA)[15], 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.

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 $E : y^2 \equiv x^3 + ax + b \pmod{N}$, $G(x_G, y_G)$ 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 [30] employing Jacobian coordinates that avoid the division and reduce computations [24, 10, 14]. Each point is defined by three Jacobian coordinates $(X; Y; Z)$ with the affine representation $(X/Z^2; Y/Z^3)$. Although there is no uniqueness of the Jacobian representation, computing the Hamming distances on $(X; Y; Z)$ produce more information than computing them on $(X/Z^2; Y/Z^3)$.

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 [24].

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 .

Algorithm 2 Montgomery Powering Ladder for ECC in RNSn

Input A point $G = (X; Y; 1)$ in Jacobian coordinates written in RNS representation

A key K with a binary representation $K = 2^{d-1}b_0 + 2^{d-2}b_1 + \dots + 2b_{d-2} + b_{d-1}$

Output

$A_0 = [K]G$ in Jacobian coordinates

$(H_i)_{i \in \{0, \dots, d-1\}}$, the Hamming distances

Procedure

Choose a random base permutation

$A_0 = (|XM|_N, |YM|_N, |M|_N)$, *Montgomery form of G*

$A_1 = [2]A_0$

$H_0 = \mathbf{Hamming\ Weight\ of\ } (A_0, A_1)$

for $i=1$ to $d-1$ **do**

$A'_0 = A_0$ et $A'_1 = A_1$

$A_{b_i} = A_{b_i} + A_{b_i}$

$A_{b_i} = [2]A_{b_i}$

$H_i = \mathbf{Hamming\ distance\ between\ } (A_0, A_1) \mathbf{\ and\ } (A'_0, A'_1)$

end for

Result $A_0 = (|X'M|_N, |Y'M|_N, |Z'M|_N)$ *in Montgomery form*

Return to the Non-Montgomery form with mixed radix [5]

$A_0 = (|X'|_N, |Y'|_N, |Z'|_N)$

end Procedure

3 A conditional attack strategy and limitations of CPA, DPA, second-order DPA and MIA

Since Hamming distances have gaussian distribution (see figure 3 b) and most of the statistic tests, like NIST's ones [31], evaluate discrete and uniform distribution, we choose to use side-channel attacks as a tool in order to evaluate randomisation of Hamming distances.

The randomization of moduli generates effectively noisy data. Because of the lack of structure in Hamming distances, it is quite difficult for an attacker to develop a denoising procedure. Consequently, an attack should target the Hamming distances that provide the most 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 attack is not monotonous with respect to the number of moduli. Section 3.3 discusses the adaptation of more recent attacks to RNS randomization.

From now on, we denote S the size of simulations.

3.1 Sufficient information and conditional attack

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 lag that separates each couple.
- γ) Under randomization of moduli, each Hamming distance H_i has a normal distribution. This remark shall not make completely absurd the assumption that the whole vector H is Gaussian.

These are important properties as they allow to put in place an efficient attack based on the first few Hamming distances. Once the first few bits associated to the first Hamming distances are known, we attack only few following bits conditionally on the fact that we found the first and so on till we find the whole secret K .

As for property α), at each step i of MPL detailed in Algorithm 2, we study the dependence between the random variables K and H_i that take respectively their values on integers in $[0, 2^p[$ and $[\min(H_i), \max(H_i)]$ (Maximum/minimum taken on the realizations of these bounded random variables). In order to reduce the complexity of computations and increase the Monte Carlo accuracy, we use appropriate subdivisions (appropriate parameters p' , q and λ) of:

$$\text{a) } [0, 2^p[= \bigcup_{k=0}^{2^{p'}-1} I_k = \bigcup_{k=0}^{2^{p'}-1} [k2^{p-p'}, (k+1)2^{p-p'}[, (p' < p)$$

and

$$\text{b) } [\min(H_i), \max(H_i)] = \bigcup_{j=0}^{q-1} \mathcal{H}_j^i$$

with

$$\mathcal{H}_0^i = [\min(H_i), \min(H_i) + \lambda[, \mathcal{H}_{q-1}^i = [\max(H_i) - \lambda, \max(H_i)],$$

$$\mathcal{H}_j^i = [\min(H_i) + \lambda + j\epsilon, \min(H_i) + \lambda + (j+1)\epsilon[$$

for

$$j = 1, \dots, q-2 \text{ where } \epsilon = \frac{\max(H_i) - \min(H_i) - 2\lambda}{q-2}$$

and the choice of λ is closely linked to the mean and the standard deviation of H_i which has a Gaussian distribution. The accuracy of Monte Carlo was quantified thanks to the 95% confidence interval; with 95% chance we have at most a 10% relative error.

As introduced informally in (1), the dependence is quantified through the distance between the probability of the product $P(H_i \in \mathcal{H}_j^i, K \in I_k) = P(K \in I_k)P(H_i \in \mathcal{H}_j^i|K \in I_k)$ and the the product of probabilities $P(H_i \in \mathcal{H}_j^i)P(K \in I_k)$. For the subdivisions a) and b), we compute this distance using the Total Variation to Independence (TVI) [20] given by

$$\text{TVI}_i = \frac{1}{2} \sum_{k=0}^{2^{p'}-1} \sum_{j=0}^{q-1} P(K \in I_k) \left| P(H_i \in \mathcal{H}_j^i) - P(H_i \in \mathcal{H}_j^i|K \in I_k) \right| \quad (7)$$

The value of $P(K \in I_k)$ is known since we draw uniformly an integer value on $[0, 2^p[$. However, the value $P(H_i \in \mathcal{H}_j^i | K \in I_k)$, and subsequently $P(H_i \in \mathcal{H}_j^i)$, is approximated using Monte Carlo simulation. For more mathematical details on Monte Carlo simulation we refer the reader to [16].

In Figure 2, we calculate TVI_i for each step in MPL either for a fixed choice of moduli or when they are randomized. Consequently, when the moduli configuration is fixed we draw only independent keys $\{K^l\}_{1 \leq l \leq S}$ and 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

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

or by

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

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 [19] that is appropriate computationally and statistically for Monte Carlo simulation.

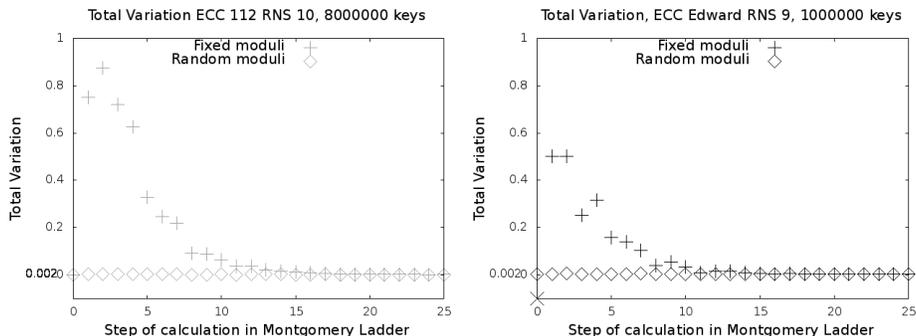


Fig. 2. Total variation as a function of the calculation step.

According to Figure 2, randomizing moduli reduces effectively TVI for all Hamming distances. Also, according to Figure 2 when conditioning on the choice of moduli, 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

$$\text{Cov}(H_i, H_j) \approx \frac{1}{S} \sum_{l=1}^S H_i(K^l) H_j(K^l) - \frac{1}{S} \sum_{l=1}^S H_i(K^l) \frac{1}{S} \sum_{k=1}^S H_j(K^k).$$

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

It is property γ) that makes the covariance very important. Indeed, in a multivariate Gaussian vector, each two coordinates are independent if and only if their covariance is equal to zero. When it is not obvious to show numerically the multivariate Normal distribution, 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 3 (b) a histogram associated to H_{10} that shows a bell-shaped distribution. For more mathematical details on multivariate Normal distribution, we refer the reader to [16].

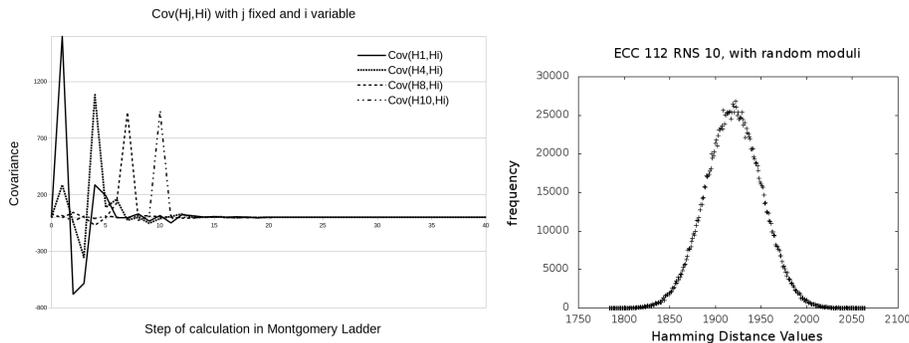


Fig. 3. (a) RNS10, $\text{Cov}(H_j, H_i)_{j=1,4,8,10}$. (b) Frequency of H_{10} , $2e6$ computations.

3.2 Unreliable CPA and inconsistent DPA

The essential result of Section 3.1 is that an effective attack 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 an attack 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.

Unfortunately, CPA and DPA attacks are not conceived to take advantage of this cross-information. Although CPA and DPA are attacks on power consumption, we apply them directly on Hamming distances. We focus rather on the pure software information without hardware noise which can be justified by leakage models presented in [21] between the power consumption and the Hamming distances.

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

$$\xi_i = \frac{\frac{1}{S} \sum_{l=1}^S [H_i(K, C^l) - \bar{H}_i(K, C)] [H_i(K', C^{l+S}) - \bar{H}_i(K', C)]}{\sqrt{\frac{1}{S} \sum_{l_1=1}^S [H_i(K, C^{l_1}) - \bar{H}_i(K, C)]^2 \frac{1}{S} \sum_{l_2=1}^S [H_i(K', C^{l_2+S}) - \bar{H}_i(K', C)]^2}} \quad (8)$$

where

$$\bar{H}_i(K, C) = \frac{1}{S} \sum_{j=1}^S H_i(K, C^j) \quad \text{and} \quad \bar{H}_i(K', C) = \frac{1}{S} \sum_{k=1}^S H_i(K', C^{k+S}). \quad (9)$$

The lag $+S$ is not usual in the expression (8) of ξ_i , but it is natural since the moduli configurations $\{C^l\}_{1 \leq l \leq S}$ used by the system attacked is supposed to be independent from the moduli configuration $\{C^l\}_{S+1 \leq l \leq 2S}$ used by the attacker. The independence of the sequence $\{C^l\}_{1 \leq l \leq 2S}$ makes the use of CPA completely irrelevant as shown in Figure 4. We use `0xddeefbf7` as the key under attack and `0xffffffff` is used as a distinguisher.

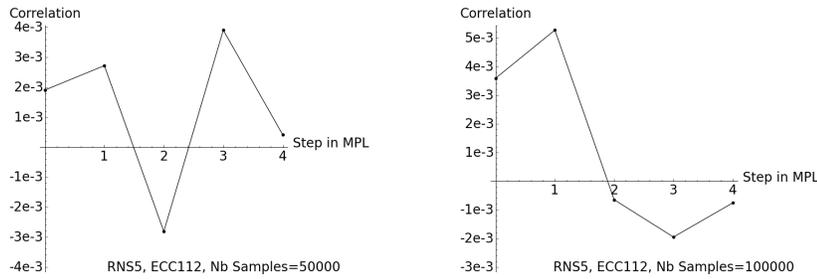


Fig. 4. RNS5, Correlation between `0xffffffff` and `0xddeefbf7`, 50000 and 100000 samples. Nothing appears at bit 2 and the correlations are too small.

Regarding the DPA based on Hamming distances, its value is given at each step i by

$$DPA_i = \overline{H}_i(K, C) - \overline{H}_i(K', C), \quad (10)$$

where $\overline{H}_i(K, C)$ and $\overline{H}_i(K', C)$ are defined in (9). Unlike for CPA attack, the lag $+S$ involved in the expression of DPA_i is less disturbing because, by the law of large numbers, $\frac{1}{S} \sum_{k=1}^S H_i(K', C^{k+S})$ and $\frac{1}{S} \sum_{k=1}^S H_i(K', C^k)$ converge to the same value as $S \rightarrow \infty$. As a consequence, although not perfect, the DPA can be used for an attack 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 5, sometimes we even need a bigger S for an RNS with less randomized moduli!

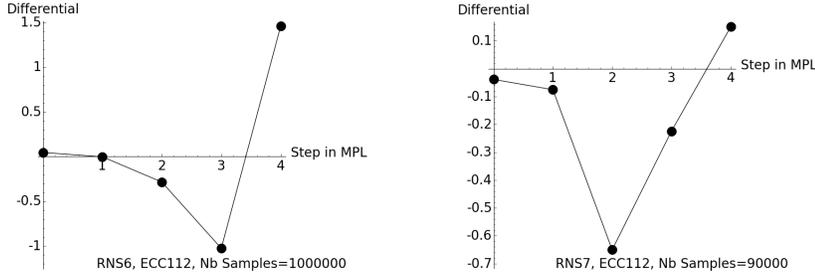


Fig. 5. RNS6 and RNS7: DPA between $0 \times ffffffff$ and $0 \times deeebf7$ with respectively 1000000 and 90000 samples. A jump appears for the bit 2 for RNS7 as we expected but the jump is not obvious for RNS6 and we needed more samples to have this little jump

3.3 Further attacks: Second order DPA, MIA and template attack

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

We make a simulation of second order DPA (2ODPA) as follow:

$$2ODPA_0 = DPA_0 \quad (11)$$

$$2ODPA_i = DPA_{i+1} - DPA_i \text{ if } i > 0 \quad (12)$$

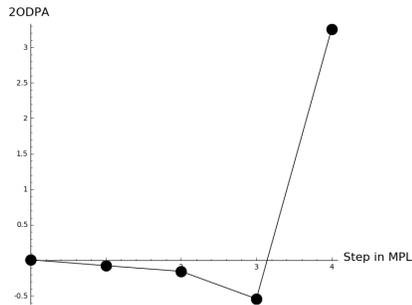


Fig. 6. RNS6: Second order DPA between $0 \times ffffffff$ and $0 \times deeebf7$ with 1000000 samples.

According to Figure 6, we see that the second order DPA on Hamming distances does not improve the results of DPA presented in Figure 5 (left part). 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 [25] (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.

Applying template attack [9] on Hamming distances should provide better results than DPA. Indeed, template attack is based on a maximum likelihood approach with a learning phase. The performance of this method is however limited by equation (2) of [9] related, in our context, to $\sum_{i=0}^p |DPA_i|$ with DPA_i expressed in (10). As a future work, we would like to compare DPA Square with template attack on RNS randomized systems.

Regarding MIA [6] applied to Hamming distances, we saw clearly in Figure 2 that the TVI is almost equal to zero when we perform randomization. Consequently, it is computationally very complex to develop an attack based on the mutual information for RNS randomized systems. MIA implementation involves the approximation of the logarithm of a probability which is much more computationally involving than the probability itself.

4 A conditional attack strategy with DPA Square,

Thanks to the marginal Gaussian behaviour of Hamming distances announced in property γ) of Section 3.1, it is not absurd to assume that the vector H of Hamming distances has a multivariate Normal distribution. With the latter assumption, H is completely specified by its mean vector and its covariance matrix. Section 4.1 presents the mathematical tools for DPA Square and Section 4.2 provides the numerical results.

Since the attack is performed by parts using ~ 10 successive Hamming distances, without a loss of generality, we will denote them by (H_0, \dots, H_{d-1}) . Con-

sequently, from now on, H_0 and H_{d-1} designate respectively the first and the last Hamming distance involved in each part of the attack.

4.1 Theoretical basis of DPA Square

When the DPA is an attack on the mean vector of H , the DPA Square is an attack on its covariance matrix. The DPA Square has then a big advantage on the DPA because it uses the cross-information given by the different Hamming distances. We have chosen the name DPA Square instead of second-order DPA since the latter was already used for another DPA concept in [25]. The second-order DPA uses differences of differences of Hamming distances whereas the DPA square uses the difference of elements of the covariance matrix.

Unlike the DPA, which is unpredictable (see figure 5), the number of samples to succeed the DPA square is increasing with respect to the number of moduli as we can see on figure 9. Among the original contributions of this paper is the asymptotic quantification of the number of observations needed for an attack. This contribution was possible thanks to DPA square in contrast to DPA which is much less conclusive.

Assume $H = (H_0, \dots, H_{d-1})$ Gaussian, its mean value is $\mathbb{E}(H)$ and let $\sigma^2(H_i)$ be the variance of each coordinate. For a fixed key K , studying the covariance matrix of $H(K)$ is equivalent to study the covariance matrix of $Y(K) = (Y_0(K), \dots, Y_{d-1}(K))$ with $Y_i(K) = \frac{H_i(K) - \mathbb{E}(H_i(K))}{\sigma(H_i(K))}$ which will be denoted by $\Gamma^K = \mathbb{E}({}^tY(K)Y(K))$ which provides

$$\Gamma^K = \begin{pmatrix} \text{var}(Y_0(K)) & \text{cov}(Y_0(K), Y_1(K)) & \dots & \text{cov}(Y_0(K), Y_{d-1}(K)) \\ \text{cov}(Y_1(K), Y_0(K)) & \text{var}(Y_1(K)) & \dots & \text{cov}(Y_1(K), Y_{d-1}(K)) \\ \vdots & \vdots & \ddots & \vdots \\ \text{cov}(Y_{d-1}(K), Y_0(K)) & \text{cov}(Y_{d-1}(K), Y_1(K)) & \dots & \text{var}(Y_{d-1}(K)) \end{pmatrix}$$

Using a sample of size S , Γ^K is approximated by $\tilde{\Gamma}^K$ and we compute $\tilde{\Gamma}^{K'}$ associated to a guessed key K'

$$\tilde{\Gamma}^K = \frac{1}{S} \sum_{j=1}^S {}^tY(K, C^j)Y(K, C^j), \quad \tilde{\Gamma}^{K'} = \frac{1}{S} \sum_{j=1}^S {}^tY(K', C^{j+S})Y(K', C^{j+S})$$

then compute the distance using Frobenius norm $\|\cdot\|$.

Thanks to the Central Limit Theorem, we can assume that $\tilde{\Gamma}^K = \Gamma^K + \delta^K$ and $\tilde{\Gamma}^{K'} = \Gamma^{K'} + \delta^{K'}$ where $\sqrt{S}\delta^K$ and $\sqrt{S}\delta^{K'}$ are asymptotically matrices of centred Gaussian variables with a variances smaller or equal to 1. We define then the DPA Square

$$\text{DPA2} = 2\|\tilde{\Gamma}^K - \tilde{\Gamma}^{K'}\|^2. \quad (13)$$

Because

$$\begin{aligned}\|\tilde{\Gamma}^K - \tilde{\Gamma}^{K'}\|^2 &= \sum_{0 \leq i, j \leq d-1} (k_{i,j} - k'_{i,j} + \delta_{i,j} - \delta'_{i,j})^2 \\ &\leq 2 \sum_{0 \leq i, j \leq d-1} (k_{i,j} - k'_{i,j})^2 + (\delta_{i,j} - \delta'_{i,j})^2\end{aligned}$$

and $\sqrt{S}(\delta_{i,j} - \delta'_{i,j})$ has asymptotically a normal distribution $G_{i,j}$ with a variance smaller or equal to 2 thus

$$\begin{aligned}\mathbb{E} \left[\|\tilde{\Gamma}^K - \tilde{\Gamma}^{K'}\|^2 \right] &\leq 2 \sum_{0 \leq i, j \leq d-1} (k_{i,j} - k'_{i,j})^2 + 2\mathbb{E} \left[\sum_{0 \leq i, j \leq d-1} \left(\frac{\sqrt{2}}{\sqrt{S}} G_{i,j} \right)^2 \right] \\ &\leq 2 \sum_{0 \leq i, j \leq d-1} (k_{i,j} - k'_{i,j})^2 + \frac{4}{S} \sum_{0 \leq i, j \leq d-1} \mathbb{E} [G_{i,j}^2]\end{aligned}$$

$$\mathbb{E} \left[\|\tilde{\Gamma}^K - \tilde{\Gamma}^{K'}\|^2 \right] \leq 2 \sum_{0 \leq i, j \leq d-1} (k_{i,j} - k'_{i,j})^2 + \frac{4d^2}{S} = 2\|\Gamma^K - \Gamma^{K'}\|^2 + \frac{4d^2}{S}. \quad (14)$$

This latter expression tells that one has to use S big enough to do an attack and decrease the asymptotic error term $\frac{4d^2}{S}$ when compared to $2\|\Gamma^K - \Gamma^{K'}\|^2$.

4.2 Numerical results of DPA Square

In order to have an efficient attack based on DPA Square, $2\|\Gamma^K - \Gamma^{K'}\|^2$ has to be bigger than $\frac{4d^2}{S}$. Because we do not know the value of $2\|\Gamma^K - \Gamma^{K'}\|^2$, we replace it in our attacks by $\text{DPA2} = 2\|\tilde{\Gamma}^K - \tilde{\Gamma}^{K'}\|^2$.

As showed in the following figures, attacking our RNS5 requires $S \geq 4000$ to have a jump of DPA2 above the estimated error (using two independent Monte Carlo) and the asymptotic one $\frac{4d^2}{S}$. These figures show an attack on the key 0xdeefbf7 with the bit 2, 7, 11, 15, 21 and 28 at zero.

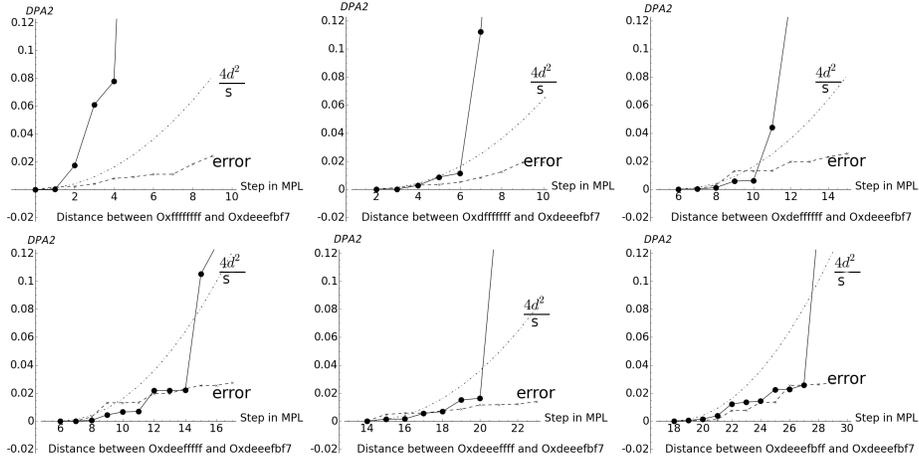


Fig. 7. DPA2 in RNS5 on ECC112 with $S = 4000$: Each new jump over $\frac{4d^2}{S}$ gives the index of the bit that is equal to zero.

Regarding RNS10, we needed $S \geq 2500000$ to do our attack illustrated for two bits in Figure 8.

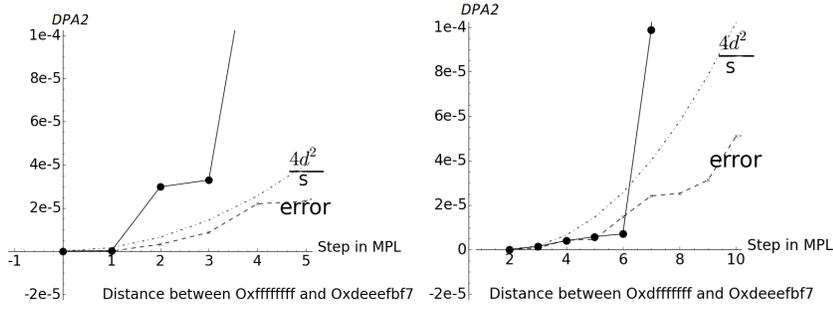


Fig. 8. DPA2 in RNS10 on ECC112 with $S = 2500000$: Each new jump over $\frac{4d^2}{S}$ gives the index of the bit that is equal to zero.

Figure 9 shows the evolution in average of the required number of observations S to perform an attack. It is quite remarkable to see that S should be of the order of $(2n)!/(n!)^2$ which represents the number of combinations in a an RNS n .

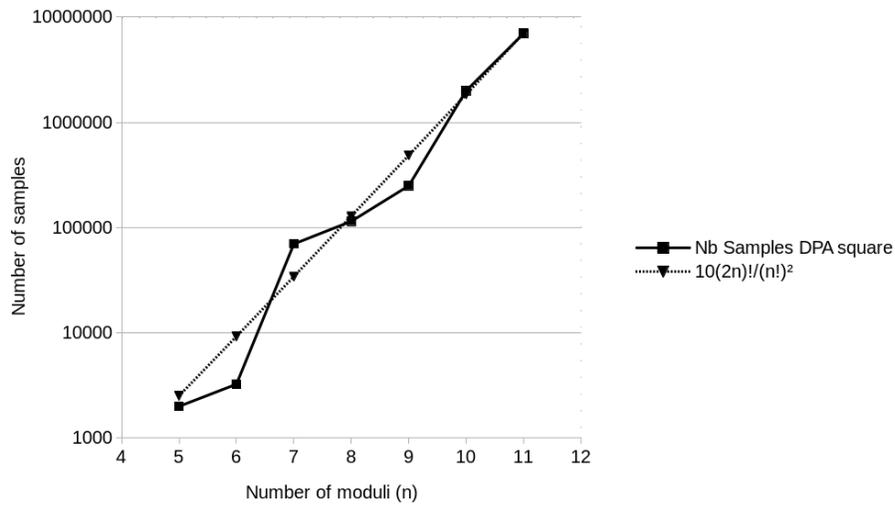


Fig. 9. Attack with DPA2: Size of observations S to attack the first 10 bits of 0xddeeffb7 with respects to the number of moduli

Finally we show some results of DPA Square implementation on Edward curve 25519 in Figure 10.

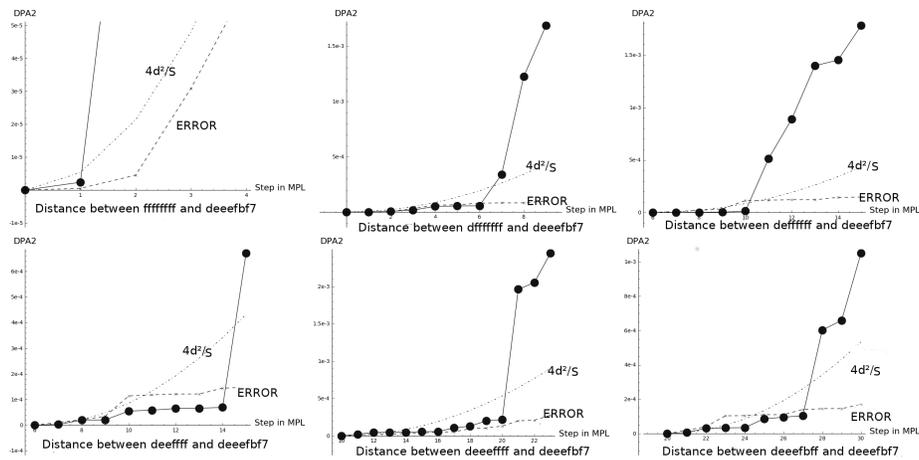


Fig. 10. DPA2 in RNS9 on ECC Edward 25519 of 255 bits with $S = 750000$: Each new jump over $\frac{4d^2}{S}$ gives the index of the bit that is equal to zero.

5 Conclusion and future work

In this work, we presented the notion of DPA Square that takes advantage of the cross-information in the Hamming distances. This provides an efficient attack 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 $(2n)!/(n!)^2$.

We have started testing further the RNS randomization, especially we would like to provide t-test [12, 29] results in a future work. We are also projecting to implement template attack [9] on RNS randomization and compare it with DPA Square.

References

1. 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”, Asia and South Pacific Design Automation Conference, pp. 620–625, 2013.
2. S. Antao, J.C. Bajard and L. Sousa, “RNS-Based Elliptic Curve Point Multiplication for Massive Parallel Architectures”, The Comput. J. Oxford J., vol. 55(5), pp. 629–647, 2011.
3. J.C. Bajard, L.S. Didier and P. Kornerup, “Modular Multiplication and Base Extensions in Residue Number Systems”, IEEE symposium on computer arithmetic, pp. 59–65, 2001.
4. J.C. Bajard, L. Imbert, P.Y. Liardet, Y. Teglia, “Leak Resistant Arithmetic”, Cryptographic Hardware and Embedded Systems, Springer LNCS vol. 3156, pp. 62–75, 2004.
5. J.C. Bajard and T. Plantard, *RNS bases and conversions*. LIRMM UMR 5506, University of Montpellier 2, France.
6. L. Batina, B. Gierlich, E. Prouff, M. Rivain, F.X. Standaert and N. Veyrat-Charvillon, “Mutual Information Analysis: a Comprehensive Study”, J. Cryptol. 24, pp. 269–291, 2011.
7. D.J. Bernstein, T. Lange, ”Faster addition and doubling on elliptic curves” in: Asiacrypt 2007 vol. 19, pp. 29–50, 2007.
8. E. Brier, C. Clavier and F. Olivier, “Correlation Power Analysis with a Leakage Model”, Cryptographic Hardware and Embedded Systems, Springer LNCS vol. 3156, pp. 16–29, 2004.
9. S. Chari, J. R. Rao and P. Rohatgi, “Template Attacks”, Cryptographic Hardware and Embedded Systems, Springer LNCS vol. 2523, pp. 13–28, 2003.
10. H. Cohen and G. Frey, *Handbook of Elliptic and Hyperelliptic Cryptography*. Chapman & Hall, 2006.
11. J. Fan, I. Verbauwhede, “An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost”, in: Naccache, D. (ed.) Quisquater Festschrift, Springer LNCS vol. 6805, pp. 265–282, 2012.
12. G. Goodwill, B. Jun, J. Jaffe and P. Rohatgi, *A testing methodology for side channel resistance validation*. In NIST non-invasive attack testing workshop, 2011. http://csrc.nist.gov/news_events/non-invasive-attack-testing-workshop/papers/08_Goodwill.pdf .
13. N. Guillermin, *Implémentation matérielle de coprocesseurs haute performance pour la cryptographie asymétrique*. PhD thesis, Université Rennes 1, 2012.

14. D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Springer, 2004
15. T. Izu and T. Takagi, “A fast parallel elliptic curve multiplication resistant against side channel attacks”, International Workshop on Public Key Cryptography, Springer LNCS vol. 2274, pp. 280–296, 2002.
16. J. Jacod and P. Protter, *Probability Essentials*, second edition, Springer-Verlag, 2003.
17. H. Kawamura, M. Koike, F. Sano, and A. Shimbo, “Cox-Rower Architecture for Fast Parallel Montgomery Multiplications”, EUROCRYPT, pp. 523–538, 2000.
18. P. Kocher, J. Jaffe, B. Jun and P. Rohatgi, “Introduction to differential power analysis”, J. Cryptogr Eng, 1, pp. 5–27, 2011.
19. P. L’Ecuyer, R. Simard, E. J. Chen, and W. D. Kelton, “An Objected-Oriented Random-Number Package with Many Long Streams and Substreams”, Operations Research, vol. 50(6), pp. 1073–1075, 2002.
20. D.A. Levin, Y. Peres and E. L. Wilmer, “Markov Chains and Mixing Times”, American Mathematical Soc. ISBN 9780821886274
21. S. Mangard, E. Oswald and T. Popp, *Power Analysis Attacks*, Springer, 2007.
22. G. Martínez, H. Encinas, S. Ávila: A Survey of the Elliptic Curve Integrated Encryption Scheme, JCSE, 2, 2 (2010), 7-13.
23. M. Medwed and C. Herbst, “Randomizing the Montgomery Multiplication to Repel Template Attacks on Multiplicative Masking”, in COSADE, 2010.
24. N. Meloni, “New Point Addition Formulae for ECC Applications”, International Workshop on the Arithmetic of Finite Fields, vol. 4547, pp. 189–201, 2007.
25. T.S. Messerges, “Using second-order power analysis to attack DPA resistant software”, Cryptographic Hardware and Embedded Systems, Springer LNCS vol. 1965, pp. 238–251, 2000.
26. P. Montgomery, “Modular multiplication without trial division”, Mathematics of Computation, vol. 44(170), pp. 519–521, 1985.
27. G. Perin, L. Imbert, L. Torres and P. Maurine, “Attacking Randomized Exponentiations Using Unsupervised Learning”, COSADE, Springer LNCS vol. 4547, pp. 144–160, 2014.
28. A.P. Shenoy and R. Kumaresan, “Fast base extension using a redundant modulus in RNS”, IEEE Transactions on Computer, vol. 38(2), pp. 292–296, 1989.
29. T. Schneider and A. Moradi, “Leakage assessment methodology”, J. Cryptographic Engineering, vol. 6(2), pp. 85–99, 2016.
30. September 20, 2000 Version 1.0 and 2.0: STANDARDS FOR EFFICIENT CRYPTOGRAPHY Recommended Elliptic Curve Domain Parameters, Certicom Research.
31. NIST Special Publication 800-22rev1a (dated April 2010), A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications