Randomization of Arithmetic over Polynomial Modular Number System

Laurent-Stéphane Didier, Fangan-Yssouf Dosso, Nadia El Mrabet, Jérémy Marrez, Pascal Véron

Randomization of Arithmetic over Polynomial Modular Number System

Laurent-Stéphane Didier*, Fangan Yssouf Dosso*, Nadia El Mrabet†, Jérémy Marrez ‡ and Pascal Véron *
*Laboratoire IMATH, Université de Toulon, France, *didier,dosso,veron@univ-tln.fr*
†Ecole des Mines de St Etienne, France, *nadia.el-mrabet@emse.fr*
‡LIP6, Sorbonne Université, France, *jeremy.marrez@lip6.fr*

***Abstract*—The Polynomial Modular Number System (PMNS) is an integer number system designed to speed up arithmetic operations modulo a prime p. Such a system is defined by a tuple $\mathcal{B} = (p, n, \gamma, \rho, E)$ where $E \in \mathbb{Z}[X]$ and $E(\gamma) \equiv 0 \pmod{p}$. In a PMNS, an element a of $\mathbb{Z}/p\mathbb{Z}$ is represented by a polynomial A such that: $A(\gamma) \equiv a \pmod{p}$, $\deg A < n$ and $\|A\|_\infty < \rho$. In [6], the authors mentioned that PMNS can be highly redundant but they didn't really take advantage of this possibility. In this paper we use, for the first time, the redundancy of PMNS to protect algorithms against Side Channel Attacks (SCA). More precisely, we focus on elliptic curve cryptography. We show how to randomize the modular multiplication in order to be safe against existing SCA and we demonstrate the resistance of our construction. We describe the generation of a PMNS while guaranteeing, for all elements of $\mathbb{Z}/p\mathbb{Z}$, the minimum number of distinct representations we want. We also show how to reach all these representations.**

***Keywords*-Polynomial Modular Number System, Side Channel Countermeasure, Modular Artithmetic.**

I. Introduction

Most protocols in public key cryptography require modular arithmetic operations over large integers, like for instance RSA [28] or Elliptic Curve Cryptography (ECC) [21]. In practice, these operations must be fast and secure. In order to speed up modular arithmetic, specific representations of integers such as the Residue Number System (RNS [17], [3]) or the Polynomial Modular Number System (PMNS [5], [6], [25], [26]) have been studied. The security concerns the resistance to side channel analysis, especially when the implementation targets embedded devices.

Side channel attacks (SCA) use the leakage of information during the execution of a cryptographic protocol in order to fully or partially recover the secret [22]. The leakage of information can be the execution time, the power consumption or the electromagnetic emission of the implemented algorithm. SCA have proven to be efficient in ECC [1]. Countermeasures to those attacks should be included in the implementation of the scalar multiplication in ECC. This operation is the main and most critical operation in ECC. It consists in adding a point $\mathcal{P}$ on an elliptic curve E, k times. Existing countermeasures rely on the addition of randomness during the computation. The randomness could be included in the scalar k [13], [12], [29], [10], [9] or in the coordinates of the point $\mathcal{P}$ [13]. It is also possible to randomize the instructions flow of the field multiplications computed during the points addition [11]. Another strategy is to introduce randomization at the arithmetical level, which is the purpose of this paper.

Our goal is to protect elliptic curve scalar multiplication (ECSM) against SCA [22] using the PMNS to represent the coordinates of any curve points. All operations involved in PMNS representation use regular algorithms so they are intrinsically Simple Power Analysis (SPA) immune. Thus, it should be sufficient to use regular algorithm (like the Montgomery powering ladder [24]) to perform the ECSM in order to be safe against SPA attacks. To protect the classical scalar multiplication $k\mathcal{P}$ against Differential Power Analysis (DPA) attacks, we first show how the conversion process which maps an integer to a representative in PMNS can be easily modified to randomize the base point $\mathcal{P}$. Next, we show how to randomize all intermediate values involved in the scalar multiplication by adding some randomness in the PMNS modular multiplication primitive.

Randomization of the scalar k can be done using classical countermeasures [15].

The remaining of the paper is organised as follow. We recall the principal definition and properties of a PMNS representation in Section II. In Section III we present both the randomization of inputs (Sec. III-B) and the multiplication (Sec. III-C). In Section IV, we give costs of modular multiplications in PMNS and describe some specific advantages of PMNS regarding some attacks like Goubin's [18]. We conclude in Section V.

II. Background on PMNS

Modular arithmetic is one of the key point for efficient and secure cryptographic applications. A challenge is to obtain a number system which permits fast modular computations over large integers.

Bajard and al. [6] introduced the Modular Number System (MNS) which is a generalization of positional number systems. The main idea is that any integer $x = \sum_{i=0}^{n} x_i \beta^i$, can be seen as a polynomial evaluated in β. In MNS, the choice of β is free under certain conditions. This number system is defined by the tuple (p, n, γ, ρ) as follows.

Definition II.1. A modular number system (MNS) $\mathcal{B}$ is defined by a tuple (p, n, γ, ρ), such that for every integer $0 \leqslant x < p$, there exists a vector $V = (v_0, \ldots, v_{n-1})$ such that: $x = \sum_{i=0}^{n-1} v_i \gamma^i \mod p$, with $|v_i| < \rho$ and $0 < \rho, \gamma < p$. In that case, we say that V (or equivalently the polynomial $V(X) = v_0 + v_1.X + \cdots + v_{n-1}.X^{n-1}$) is a representation of x in $\mathcal{B}$ and we notate $V \equiv x_{\mathcal{B}}$, which means that $V(\gamma) \equiv x \pmod p$. In this system, arithmetic operations are performed on polynomials.
An example of MNS can be found in [6].

The product T of two MNS numbers $V \equiv x_{\mathcal{B}}$ and $W \equiv y_{\mathcal{B}}$ satisfies $T(\gamma) \equiv xy \pmod p$. However, T might not be in $\mathcal{B}$ because its degree could be greater or equal to n. The Polynomial Modular Number System is an extension of MNS which keep the degree of the product bound by n.

Definition II.2. A Polynomial Modular Number System (PMNS) [6] is a MNS where γ is a root modulo p of an unitary polynomial $E(X) \in \mathbb{Z}[X]$, such that $\deg(E) = n$ and $\|E\|_\infty$ is "small". The polynomial E is named *external reduction polynomial*. The PMNS $\mathcal{B}$ is defined by the tuple (p, n, γ, ρ, E). In this system, arithmetic operations are performed on polynomials modulo E.

Several methods exists for computing the PMNS parameters. Plantard [26] give a building method for very efficient PMNS. Its main drawback is that the parameter p cannot be set. It is computed through the process. In [7], [14] the authors show that it is always possible to build many PMNS for a given modulus p.

In PMNS, the multiplication $T = VW \pmod E$, with $V \equiv x_{\mathcal{B}}$ and $W \equiv y_{\mathcal{B}}$, satisfies $T(\gamma) \equiv xy \pmod p$ because $E(\gamma) \equiv 0 \pmod p$. However, even if $\deg(T) < n$, T might not be a representation of $xy \pmod p$ in $\mathcal{B}$, because its coefficients could be greater or equal to ρ. In order to get this representation in $\mathcal{B}$, a special primitive called the *internal reduction* has to be applied. The operation that reduces the size of the polynomial coefficients is described in Section II-A.

In [26], [6], the authors show how to use PMNS to speed up modular arithmetic. The main PMNS primitives are recalled in Section II-B, II-C and II-D.

A. Internal reduction

The goal of the internal reduction is to maintain small enough coefficients of polynomials in PMNS. Let $\mathcal{B} = (p, n, \gamma, \rho, E)$ be a PMNS. The internal reduction process maps a polynomial $V(X) \in \mathbb{Z}[X]$ to a polynomial $\tilde{V}(X) \in \mathbb{Z}[X]$ such that $\tilde{V}(\gamma) \equiv V(\gamma) \pmod p$, $\|\tilde{V}\|_\infty \leqslant \|V\|_\infty$ and $\deg \tilde{V} = \deg V$. We describe two ways to perform this operation.

1) Internal reduction via a Mongtomery-like method: In [25], the authors give a reduction procedure similar to the Mongtomery algorithm (Alg. 1). It outputs a polynomial $\tilde{V}$ such that $\tilde{V} \equiv V(\gamma) r^{-1} \pmod p$. Using this version of the internal reduction requires to convert the elements of $\mathbb{Z}/p\mathbb{Z}$ in the Montgomery domain during the conversion process into the PMNS. An element $a \in \mathbb{Z}/p\mathbb{Z}$ is represented by a polynomial $A \in \mathcal{B}$ such that $A(\gamma) \equiv a.r \pmod p$. This way, we ensure the consistency of operations in the PMNS while using this method for coefficients reduction.

The choice of the polynomial M to ensure the existence of M' is discussed in [14]. The

Algorithm 1 RedCoeff - Montgomery like[25]

Require: $V \in \mathbb{Z}[X]$ such that $\deg(V) < n$; $\mathcal{B} = (p, n, \gamma, \rho, E)$; $M \in \mathcal{B}$, such that $M(\gamma) \equiv 0 \pmod{p}$; an integer r and $M' = -M^{-1} \bmod (E, r)$.
Ensure: $R(\gamma) \equiv V(\gamma) r^{-1} \pmod{p}$
1: $Q \leftarrow V \times M' \bmod (E, r)$
2: $R' \leftarrow V + (Q \times M) \bmod E$
3: $R \leftarrow R'/r$
4: return R

Algorithm 2 RedCoeff - Babaï like [16]

Require: $V \in \mathbb{Z}[X]$ with $\deg(V) < n$, $\mathcal{B} = (p, n, \gamma, \rho, E)$
Ensure: $R(\gamma) \equiv V(\gamma) \mod p$
Data: $D = \{D_i, 1 \le i \le n\}$ the LLL-reduced base of the lattice associated to $\mathcal{B}$
$\widetilde{D} = \{\widetilde{D_i}, 1 \le i \le n\}$ the Gram-Schmidt base obtained from D
1: $R \leftarrow V$
2: **for** $i = 1$ **to** n **do**
3: $c \leftarrow \lfloor < R, \widetilde{D}_{n-i+1} > / \|\widetilde{D}_{n-i+1}\|^2 \rceil$
4: $R \leftarrow R - c \times D_{n-i+1}$
5: **end for**
6: **return** R

Proposition II.1, from [25], gives a bound on the coefficients of the polynomial computed with the Algorithm 1. Here, $E(X) = X^n - \lambda$, with $\lambda \in \mathbb{Z}\backslash\{0\}$.

Proposition II.1. *Let $m = \|M\|_\infty$. If V, ρ and m are such that: $\|V\|_\infty \leqslant n|\lambda|\rho^2$, $\rho \geqslant 2|\lambda|nm$ and $r \geqslant 2|\lambda|n\rho$ then the Algorithm 1 computes R such that $\|R\|_\infty < \rho$ (i.e $R \in \mathcal{B}$).*

We will generalise this bound in Section II-B for $E(X) = X^n - \Delta(X)$.

2) Internal reduction via a Babaï-like method: The Babaï-like algorithm (Alg. 2) does not require to maintain the elements of $\mathbb{Z}/p\mathbb{Z}$ in another domain and does not attempt to compute the result by using a polynomial M with specific properties. In this approach, we consider the Euclidean lattice $\mathfrak{L}_\mathcal{B}$ associated with the PMNS $\mathcal{B}$. Here, $\mathfrak{L}_\mathcal{B}$ is the set of all polynomials having γ as root modulo p and which degree is at most $n-1$: $\mathfrak{L}_\mathcal{B} = \{A(X) \in \mathbb{Z}[X]$, such that: $deg(A) < n$ and $A(\gamma) \equiv 0 \pmod{p}\}$.

The lattice $\mathfrak{L}_\mathcal{B}$ is defined from one of its bases, whose elements are $A_1(X) = p$ and $A_{i+1}(X) = X^i - \gamma^i$, for $1 \le i < n$, from which we compute D, the LLL-reduced base of $\mathfrak{L}_\mathcal{B}$, and the Gram-Schmidt base $\widetilde{D}$ obtained from D, whose elements are orthogonal but not necessarily in $\mathfrak{L}_\mathcal{B}$. [27].
We can deduce from [16] and [27] that:

Proposition II.2. *If ρ is such that $\rho \geqslant \frac{1}{2} 2^{\frac{3n-1}{2}} p^{1/n}$, then the Algorithm 2 computes R such that $\|R\|_\infty < \rho$ (i.e $R \in \mathcal{B}$).*

B. Multiplication

The elements in PMNS are polynomials of degree at most $n-1$. Their multiplication is done using classical polynomial multiplication. However, the result (of degree at most $2n-2$) must be reduced modulo the external reduction polynomial E. Then, an internal reduction is performed to obtain a result in $\mathcal{B}$.

The polynomial E is such that: $E(X) = X^n - \Delta(X) \in \mathbb{Z}[X]$. In order to reduce a polynomial V modulo E, we proceed step by step by replacing the term X^n in V by $\Delta(X)$ until $\deg(V) < n$. We consider here PMNS with reduction polynomials E of degree $n \ge 2$, with $\Delta(X) = \delta_k X^k + \cdots + \delta_1 X + \delta_0$ and $k \le \frac{n}{2}$.

From this technique, we compute polynomials of degree lower than n to represent the powers X^i modulo E, for $0 \le i \le 2n-2$, with $X^i \mod E = \sum_{l=i-n}^{k+i-n} \delta_{l-i+n} X^l$ when $n \le i < 2n-k$ and $X^i \mod E = \sum_{l=i-n}^{n-1} \delta_{l+n-i} X^l + \sum_{l=0}^{2k-2n+i} (\sum_{j=0}^{l} \delta_j \, \delta_{l+2n-2j}) X^l$ for $2n-k \le i \le 2n-2$.

Then, any polynomial V can be reduced modulo E by multiplying the vector of the coefficients of V by the $(2n-1)\times n$ matrix $\mathbf{S}$ whose rows represent the coefficients of each power X^i modulo E, for $0 \le i \le 2n-2$.

We denote s the 1-norm $\|\mathbf{S}\|_1$ of the matrix $\mathbf{S}$, and deduce the following proposition.

Proposition II.3. *Let A, $B \in \mathbb{Z}[X]$ be two polynomials, such that: $\deg(A) < n$ and $\deg(B) < n$. We have: $\|A \times B \mod E\|_\infty < n\, s\, \|A\|_\infty \|B\|_\infty$.*

From this proposition the bound given in Proposition II.1 can be rewritten by substituting $|\lambda|$ by s.

C. Addition

Addition procedure is done using classical polynomial addition in $\mathbb{Z}[X]$ followed by an internal reduction in order to obtain a result in $\mathcal{B}$.

D. Conversion procedures

The conversion procedures maps an integer to a representation in $\mathcal{B}$ and vice-versa. They depend on internal reduction step that can be achieved using any one of the two procedure previously described. The corresponding algorithms are descibed in section 4.3 of [6].

III. Randomization of the scalar multiplication in elliptic curve cryptography

Our motivation is to present algorithms that will ensure the resistance of the scalar multiplication against existing SCA. The scalar multiplication in ECC takes a point $\mathcal{P}$ over a public elliptic curve, a private integer k and computes $k\mathcal{P}$. This operation can be implemented using the classical double and add method, but is not resistant to SCA [22]. The Montgomery ladder [24] and its variant [8], [23], [19], [20] are more resistant but are still attackable [1]. The recent survey [1] presents existing side channel attacks and countermeasures for the scalar multiplication over ECC. Existing countermeasures rely either on the randomization of the scalar k [13], [12], [29], [10], [9], or on the randomization of the point $\mathcal{P}$. The randomization of point $\mathcal{P}$ can be calculated using the randomization of projective coordinates [13] or the addition of a random point $\mathcal{R}$ [13]. Only one countermeasure is designed on the field multiplication using a random permutation [11].

We provide here two possibilities to randomize the scalar multiplication:

- randomization of each initial coordinate of $\mathcal{P}$ each time this point is used during the scalar multiplication algorithm. This protects against SCA, where the attacker tries to find out the secret using the knowledge of the point $\mathcal{P}$ [22] and ensures the resistance to specific point attacks [18], [2]. We use either Algorithm 3 or Algorithm 4 as a conversion procedure depending on the `RedCoeff` method that will be used later for internal reduction;
- randomization of each multiplication $a \times b$ for a and b in PMNS representation to be more resistant to SCA, using either Algorithm 5 or Algorithm 6.

A. Random polynomial generation

In order to randomize data in the PMNS, we generate a random polynomial $Z \in \mathbb{Z}[X]$ such that: $\|Z\|_\infty \leq z$ and $\deg Z < n$ with $z \in \mathbb{N}$. The value z is chosen during the PMNS generation process. This integer z defines the minimum number of distinct representations of any element of $\mathbb{Z}/p\mathbb{Z}$ in $\mathcal{B}$. Once z is fixed, there are exactly $(2z+1)^n$ polynomials Z. We will show how to use Z in order to guarantee that there are at least $(2z+1)^n$ distinct representations of any element of $\mathbb{Z}/p\mathbb{Z}$.

Hereafter we use a function `randPoly(z)` for generating random polynomials which coefficients are in the set $\{-z, \ldots, z\}$. We consider this function to be safe (see [4] for example).

B. Randomization of the input data

Here, we show how to randomize the conversion procedure from binary to the PMNS in order to obtain randomized input data. Let $\mathcal{B} = (p, n, \gamma, \rho, E)$ be a PMNS. Let $a \in \mathbb{Z}/p\mathbb{Z}$ be an integer. We want to compute a randomized representation of a in $\mathcal{B}$.

1) Conversion randomization using the Mongtomery-like method: This algorithm is a slight modification of the initial conversion algorithm described in [6]. The randomized conversion method introduces a polynomial multiplication (line 4) before the internal reduction. Doing so, we ensure that with the same input and different random polynomials this algorithm produces different representations.

Theorem III.1. *Let $\mathcal{B} = (p, n, \gamma, \rho, E)$ a PMNS. Let $a \in \mathbb{Z}/p\mathbb{Z}$ be an integer and $r = 2^j$, $j \geq 1$. Let $M \in \mathcal{B}$ be a polynomial such that: $M(\gamma) \equiv 0 \pmod p$ and gcd(r, resultant(E, M)) = 1. Let z be the input of the `randPoly` procedure. We consider $\mathcal{B}$, a, r, M and z as the inputs and data of Algorithm 3. Let $m = \|M\|_\infty$. If ρ and r satisfy $\rho \geq 2\,n\,s\,m\left(1 + z + \frac{z}{r}\right)$ and $r \geq 2\,n\,s\,\rho$,, then Algorithm 3 can generate $(2z+1)^n$ distinct outputs, all representing a and belonging to the PMNS $\mathcal{B} = (p, n, \gamma, \rho, E)$.*

Proof: We give a sketch of the proof. First, it can be easily checked that $V(\gamma) \equiv U(\gamma)$

Algorithm 3 Randomized conversion to PMNS via Montgomery

Require: $a \in \mathbb{Z}/p\mathbb{Z}$ and $\mathcal{B} = (p, n, \gamma, \rho, E)$
Ensure: $A(\gamma) \equiv a\,r \pmod p$
Data: $P_i \equiv (\rho^i)_{\mathcal{B}}$, for $i = 0, \ldots, n-1$, $z \in \mathbb{N}$, $r = 2^j$, $j \geq 1$ and $M \in \mathcal{B}$ with $M(\gamma) \equiv 0 \pmod p$ and $\gcd(r, \text{resultant}(E, M)) = 1$.

1: $Z \leftarrow$ **randPoly**(z)
2: $a \leftarrow a\,r^2 \pmod p$
3: $b \leftarrow (a_{n-1}, ..., a_0)_\rho$ # radix-ρ decomposition
4: $U \leftarrow \sum_{i=0}^{n-1} b_i\,P_i$
5: $V \leftarrow U + ((r+1)Z \times M) \mod E$
6: $A \leftarrow$ **RedCoeff**(V)
7: **return** A

(mod p). Next, in order to ensure $\|A\|_\infty < \rho$ we choose r and ρ such that $r \geq 2\,n\,s\,\rho$ and $\rho \geq 2\,n\,s\,m\left(1 + z + \frac{z}{r}\right)$. Thus, A is a representation of $a\,r \pmod p$ in the PMNS. Finally, we prove by contradiction that, for the same entry $a \in \mathbb{Z}/p\mathbb{Z}$ and two distinct random polynomials, this algorithm returns two distinct outputs A_1 and A_2. ■

2) Randomization using the Babaï-like method: This algorithm is a slight modification of Algorithm 2 using the Babai-like method for coefficients reduction. Here, the representations in the PMNS system are seen as vectors, the i-th coordinate corresponding to the i-th coefficient of the polynomial form of the representation. We consider a PMNS $\mathcal{B}$ and its associated lattice $\mathfrak{L}_{\mathcal{B}}$ with a LLL-reduced base, denoted D. The randomization of the Babaï-like method is based on two random vectors:

- A *mask* vector V used to generate a linear combination of the elements of D to randomize computations without affecting the result.
- A *shift* vector Z used to randomize the output.

Theorem III.2. *Let $\mathcal{B} = (p, n, \gamma, \rho, E)$ a PMNS. Let $a \in \mathbb{Z}/p\mathbb{Z}$ be an integer. Let v and z be the inputs of the* `randPoly` *procedure respectively for the mask and the shift polynomial. We consider $\mathcal{B}$, a, v and z as the inputs and data of Algorithm 4. If ρ satisfies $\rho \geq \left(\frac{1}{2} + z\right)\left(2^{\frac{3n-1}{2}}\,p^{1/n}\right)$, then Algorithm 4 can generate $(2\,z+1)^n$ distinct outputs, all representing a and belonging to the PMNS $\mathcal{B} = (p, n, \gamma, \rho, E)$.*

Algorithm 4 Randomized conversion to PMNS via Babaï

Require: $a \in \mathbb{Z}/p\mathbb{Z}$ and $\mathcal{B} = (p, n, \gamma, \rho, E)$
Ensure: $A(\gamma) \equiv a \pmod p$
Data: $P_i \equiv (\rho^i)_{\mathcal{B}}$, for $i = 0, \ldots, n-1$, $D = \{D_i, 1 \leq i \leq n\}$, $\widetilde{D} = \{\widetilde{D_i}, 1 \leq i \leq n\}$, $v \in \mathbb{N}$ such that $\|V\|_\infty \leq v$ with $V \in \mathbb{Z}^n$ the mask vector, $z \in \mathbb{N}$ such that $\|Z\|_\infty \leq z$ with $Z \in \mathbb{Z}^n$ the shift vector.

1: $V \leftarrow$ **randPoly**(v), $Z \leftarrow$ **randPoly**(z)
2: $b \leftarrow (a_{n-1}, ..., a_0)_\rho$ # radix-ρ decomposition
3: $T \leftarrow \sum_{i=0}^{n-1} b_i\,P_i$, $A \leftarrow T + \sum_{i=0}^{n-1} v_i\,D_{i+1}$
4: **for** $i = 1$ **to** n **do**
5: $\quad c \leftarrow \lfloor < A, \widetilde{D}_{n-i+1} > / \|\widetilde{D}_{n-i+1}\|^2 \rceil + z_{n-i}$
6: $\quad A \leftarrow A - c \times D_{n-i+1}$
7: **end for**
8: **return** A

Proof: We only give a sketch of the proof. First, it is obvious that the polynomial T, in Algorithm 4, is such that: $T(\gamma) \equiv a \pmod p$. Moreover, as D is a LLL-reduced base of the lattice associated to $\mathcal{B}$, it is always true that $A(\gamma) \equiv T(\gamma) \pmod p$. Thus, $A(\gamma) \equiv a \pmod p$. Next condition on ρ guarantees that A will be in the PMNS after the internal reduction process. Finally, for two distinct random polynomials Z_1 and Z_2 with the same entry $a \in \mathbb{Z}/p\mathbb{Z}$, it can be proved that the algorithm returns two distinct outputs A_1 and A_2. ■

C. Randomization of the multiplication

In PMNS the randomization of the multiplication provides an additional level of security when used with the randomized conversion. However, combining these two randomizations leads to stronger constraints on some parameters of the PMNS.

In both versions, we randomize one input in order to randomize all the intermediate results. Moreover, for the same input and different random polynomials (shift vectors for Babaï), the results computed by these algorithms are different representations of the same integer in $\mathbb{Z}/p\mathbb{Z}$.

1) Randomization via Montgomery: This algorithm is a variation of the Montgomery modular

multiplication.

Algorithm 5 Randomized Montgomery PMNS Multiplication

Require: $\mathcal{B} = (p, n, \gamma, \rho, E)$ and $A, B \in \mathcal{B}$
Ensure: $R(\gamma) = A(\gamma)B(\gamma)r^{-1} \mod p$
Data: $r = 2^j$, $j \geq 1$, $z \in \mathbb{N}$, $M \in \mathcal{B}$, such that: $M(\gamma) \equiv 0 \pmod p$ and $\gcd(r, \text{resultant}(E, M)) = 1$, $M' = -M^{-1} \bmod (E, r)$.
1: $Z \leftarrow$ **randPoly**(z)
2: $J \leftarrow Z \times M \mod E$, $B' \leftarrow B + J$
3: $C \leftarrow (A \times B') \mod E$
4: $Q \leftarrow (C \times M') \mod (E, r)$
5: $R' \leftarrow C + (Q \times M) \mod E$
6: $R \leftarrow R'/r + 2 \times J$
7: **return** R

Theorem III.3. *Let $\mathcal{B} = (p, n, \gamma, \rho, E)$ a PMNS. Let $z \in \mathbb{N}$ be the bound of the random polynomials and $r = 2^j$, $j \geq 1$. Let $M \in \mathcal{B}$ be a polynomial such that: $M(\gamma) \equiv 0 \pmod p$ and gcd(r, resultant(E, M)) = 1. Let A and B be two elements of $\mathcal{B}$. We consider $\mathcal{B}$, A, B, M, z and r as the inputs and data of Algorithm 5. Let $m = \|M\|_\infty$. If ρ and r satisfy $\rho \geq 2\,n\,s\,m\,(2z+1)$ and $r \geq 2\,n\,s\,\rho \times max\left(z, \frac{5}{4}\right)$, then Algorithm 5 can generate $(2\,z+1)^n$ distinct outputs, all representing $A(\gamma)B(\gamma)r^{-1} \pmod p$ and belonging to the PMNS $\mathcal{B} = (p, n, \gamma, \rho, E)$.*

Proof: We only give a sketch of the proof which is similar to the proof of Th. III.1. First, we have $M(\gamma) \equiv 0 \pmod p$, so $J(\gamma) \equiv 0 \pmod p$ and $Q(\gamma)M(\gamma) \equiv 0 \pmod p$. Thus, $R(\gamma) \equiv A(\gamma)B(\gamma)r^{-1} \pmod p$.
Next, $\|R\|_\infty < \left(\frac{5}{4}\,n\,s\,\rho^2\right)/r + n\,s\,m(2\,z+1)$. Hence, for $r \geq 2\,n\,s\,\rho \times max(z, \frac{5}{4})$, $\|R\|_\infty < \frac{\rho}{2} + n\,s\,m(2\,z+1)$. This way, condition on ρ guarantees that $\|R\|_\infty < \rho$. Finally, it can be proven that for two distinct random polynomials and the same entries A and B, the algorithm returns two distinct outputs. ∎

Instead of using the bounds on ρ and r from Theorem III.1 for conversion in the PMNS, the bounds of Theorem III.3 have to be used for conversion if multiplications are to be randomized as described above.

2) Randomization via Babaï: This conversion lies on the same principles than the randomized conversion described in Section III-B2. The randomness is introduced in the multiplication trough the mask V and the translation Z.

Algorithm 6 Randomized Babaï PMNS Multiplication

Require: $\mathcal{B} = (p, n, \gamma, \rho, E)$ and $A, B \in \mathcal{B}$
Ensure: $R \in \mathcal{B}$ and $R(\gamma) = A(\gamma)B(\gamma) \mod p$
Data: $P_i \equiv (\rho^i)_{\mathcal{B}}$, for $i = 0, \ldots, n-1$, $D = \{D_i, 1 \leq i \leq n\}$, $\widetilde{D} = \{\widetilde{D_i}, 1 \leq i \leq n\}$, $v \in \mathbb{N}$ such that $\|V\|_\infty \leq v$ with $V \in \mathbb{Z}^n$ the mask vector, $z \in \mathbb{N}$ such that $\|Z\|_\infty \leq z$ with $Z \in \mathbb{Z}^n$ the shift vector.
1: $V \leftarrow$ **randPoly**(v), $Z \leftarrow$ **randPoly**(z)
2: $J \leftarrow \sum_{i=0}^{n-1} v_i\, D_{i+1}$
3: $B' \leftarrow B + J$, $R \leftarrow A \times B' \mod E$
4: **for** $i := 1$ **to** n **do**
5: $\quad c \leftarrow \lfloor < R, \widetilde{D}_{n-i+1} > /\|\widetilde{D}_{n-i+1}\|^2 \rceil + z_{n-i}$
6: $\quad R \leftarrow R - c \times D_{n-i+1}$
7: **end for**
8: **return** R

Theorem III.2 applies equally to Algorithm 6, which can generate $(2\,z+1)^n$ distinct outputs representing $A(\gamma)B(\gamma) \mod p$ in the PMNS $\mathcal{B} = (p, n, \gamma, \rho, E)$ with $\rho \geq \left(\frac{1}{2} + z\right)\left(2^{\frac{3n-1}{2}}\,p^{1/n}\right)$ and no collision.

IV. Cost evaluation

We describe here the theoretical cost estimation of the multiplication algorithms presented in the previous section. They are expressed as a function of the number of w-bit size multiplications, additions, divisions and shifts.

We assume that the inputs of these algorithms belong to $\mathcal{B} = (p, n, \gamma, \rho, E)$, with $\rho = 2^w$, $E(X) = X^n - \lambda$ and $\lambda = \pm 2^u$ with $w, u \in \mathbb{N}$. It has been shown in [14] that it is always possible to build such PMNS. As an example, with such notations the addition of two c-bit integers requires $\lceil c/w \rceil$ w-bit additions.

Let $\mathcal{M}$ and $\mathcal{A}$ respectively denote the multiplication and the sum of two w-bits integers, $\mathcal{I}$ the division by an integer of $2\,w\,\lfloor \log_2(n) \rfloor$ bits and $\mathcal{R}$ the cost of one call to the `randPoly` function. We also respectively denote $\mathcal{S}_l^i$ and $\mathcal{S}_r^i$ a left shift and a right shift of i bits. In Table I, we compare the cost of the randomized multiplication in PMNS

with their non-randomized counterparts. The non-randomized version is a simple polynomial multiplication followed by a Babaï-like (Alg. 2) or Montgomery-like (Alg. 1) internal reduction.

In order to protect the ECSM $k\mathcal{P}$ against DPA attacks, classical countermeasures randomize beforehand the scalar k and the point $\mathcal{P}$ [15]. Then, during the computation process, the intermediate points that are computed are also randomized. The expectation of these approaches is to prevent an attacker to get information about k or to make useful assumption about the intermediate values of the ECSM.

In classical binary representation, these common countermeasures appear to be inefficient against Goubin's attack [18]. This attack can be done on curves that have at least one point with one coordinate equals to zero. On such a curve, using such a point, the attacker can find information about the secret key. It is possible to thwart this attack at the cost of an additional ECSM which must be done in addition to the common countermeasures [15].

A first advantage of our PMNS based randomized solutions is that regardless the type of curve, this attack cannot be performed. Indeed, there are at least $(2z+1)^n$ distinct representatives of $0 \in \mathbb{Z}/p\mathbb{Z}$ and these representatives do not have special shapes. Thus, for z big enough, the attacker should not be able to exploit any information to perform this attack. Consequently, the only randomization of the conversion process using Algorithms 3 or 4 suffices to counter the Goubin's attack even if non-randomized multiplications are used later.

Another advantage of our approach is that it operates at arithmetic level. Hence it can be combined with other classical countermeasures (point blinding, scalar blinding) to randomize $\mathcal{P}$ and the intermediate points.

V. Conclusion

In this paper, we show for the first time how to use the redundancy of the PMNS to define arithmetical protections against DPA attacks. We described how to randomize the inputs during the forward conversion to PMNS through two methods. We also gave two randomized modular multiplications in PMNS. These methods can be used to apply classical countermeasures on the elliptic curve scalar multiplication $k\mathcal{P}$. Moreover, we showed that randomizing only the conversion process suffices to protect against Goubin's attack. As a comparison, a safe countermeasure using the classical binary representation, requires an additional ECSM $k\mathcal{R}$ for some random point $\mathcal{R}$ [15]. These results are a first step in using randomization for arithmetic operations in PMNS. This work opens up new perspectives in the area of countermeasures for SCA attacks. A deeper study on practical efficiency and an exhaustive comparison with existing countermeasures will soon follow in order to establish the relevance of these methods ...

Acknowledgment

This work was supported by ARRAND ANR-15-CE39-0002-01. We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

References

[1] R. Abarzúa, C. Valencia, and J. López. Survey for performance and security problems of passive side-channel attacks countermeasures in ECC. Cryptology ePrint Archive, Report 2019/010, 2019.

[2] T. Akishita and T. Takagi. Zero-value point attacks on elliptic curve cryptosystem. In *Information Security, 6th International Conference, ISC 2003*, pages 218–233, 2003.

[3] S. Antão, J. C. Bajard, and L. Sousa. RNS based elliptic curve point multiplication for massive parallel architectures. *The Computer Journal*, 55(5):629–647, 2012.

[4] V. Bahadur, D. Selvakumar, and P. M. Sobha. Reconfigurable side channel attack resistant true random number generator. In *International Conference on VLSI Systems, Architectures, Technology and Applications*, pages 1–6, 2016.

[5] J-C. Bajard, L. Imbert, and T. Plantard. Modular number systems: Beyond the mersenne family. In *Selected Areas in Crytography: 11th International Workshop*, LNCS, pages 159–169. Springer, 2004.

[6] J. C. Bajard, L. Imbert, and T. Plantard. Arithmetic operations in the polynomial modular number system. In *17th IEEE Symposium on Computer Arithmetic*, pages 206–213, 2005.

[7] J. C. Bajard, J. Marrez, and T. Plantard. Polynomial modular number systems and the roots of their reduction polynomial in the field $\mathbb{Z}$/p$\mathbb{Z}$. submitted, 2019.

[8] E. Brier and M. Joye. Weierstraß elliptic curves and side-channel attacks. In *Public Key Cryptography*, volume 2274 of *LNCS*, pages 335–345. Springer, 2002.

[9] B. Chevallier-Mames. Self-randomized exponentiation algorithms. In *CT-RSA*, volume 2964 of *LNCS*, pages 236–249. Springer, 2004.

TABLE I
THEORETICAL COST OF OPERATIONS, WHERE $E(X) = X^n - \lambda$ WITH $\lambda = \pm 2^u$, $r = 2^j$.

Mult. Method	Montgomery-like	Babaï-like
Polynomial Mult.	$n^2\mathcal{M} + (2n^2 - 4n + 2)\mathcal{A}$	$n^2\mathcal{M} + (2n^2 - 4n + 2)\mathcal{A}$
External reduct.	$2(n-1)\mathcal{A} + (n-1)\mathcal{S}_l^u$	$2(n-1)\mathcal{A} + (n-1)\mathcal{S}_l^u$
Internal reduct.	$2n^2\mathcal{M} + (3n^2 - n)\mathcal{A} + n\mathcal{S}_r^j$	$3n^2\mathcal{M} + (3n^2 - 2n)\mathcal{A} + n\mathcal{I}$
Total	$3n^2\mathcal{M} + (5n^2 - 3n)\mathcal{A} + (n-1)\mathcal{S}_l^u + n\mathcal{S}_r^j$	$4n^2\mathcal{M} + (5n^2 - 4n)\mathcal{A} + n\mathcal{I} + (n-1)\mathcal{S}_l^u$
Mult. Method	**Montgomery randomized (Alg. 5)**	**Babaï randomized (Alg. 6)**
Polynomial Mult.	$2n^2\mathcal{M} + (3n^2 - 4n + 2)\mathcal{A} + \mathcal{R}$	$2n^2\mathcal{M} + (3n^2 - 4n + 2)\mathcal{A} + 2\mathcal{R}$
External reduct.	$2(n-1)\mathcal{A} + (n-1)\mathcal{S}_l^u$	$2(n-1)\mathcal{A} + (n-1)\mathcal{S}_l^u$
Internal reduct.	$2n^2\mathcal{M} + 3n^2\mathcal{A} + n(\mathcal{S}_r^j + \mathcal{S}_l^1)$	$3n^2\mathcal{M} + (3n^2 - n)\mathcal{A} + n\mathcal{I}$
Total	$4n^2\mathcal{M} + (6n^2 - 2n)\mathcal{A} + (n-1)\mathcal{S}_l^u + n(\mathcal{S}_l^1 + \mathcal{S}_r^j) + \mathcal{R}$	$5n^2\mathcal{M} + (6n^2 - 3n)\mathcal{A} + n\mathcal{I} + (n-1)\mathcal{S}_l^u + 2\mathcal{R}$

[10] M. Ciet and M. Joye. (virtually) Free randomization techniques for elliptic curve cryptography. In *ICICS*, volume 2836 of *LNCS*, pages 348–359. Springer, 2003.

[11] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil. Horizontal correlation analysis on exponentiation. In *ICICS*, volume 6476 of *LNCS*, pages 46–61. Springer, 2010.

[12] C. Clavier and M. Joye. Universal exponentiation algorithm. In *CHES*, volume 2162 of *LNCS*, pages 300–308. Springer, 2001.

[13] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *CHES*, volume 1717 of *LNCS*, pages 292–302, 1999.

[14] L.-S. Didier, F.-Y. Dosso, and P. Véron. Efficient and secure modular operations using the adapted modular number system. https://arxiv.org/abs/1901.11485, 2018.

[15] J. Fan and I. Verbauwhede. An updated survey on secure ECC implementations: Attacks, countermeasures and cost. In *Cryptography and Security: From Theory to Applications*, pages 265–282. Springer, 2012.

[16] Steven Galbraith. Algorithms for the closest and shortest vector problem. *Mathematics of Public Key Cryptography*, 2011.

[17] H. L. Garner. The residue number system. *IRE Transactions on Electronic Computers*, EL 8(6):140–147, 1959.

[18] L. Goubin. A refined power-analysis attack on elliptic curve cryptosystems. In *Public Key Cryptography*, volume 2567 of *LNCS*, pages 199–210. Springer, 2003.

[19] R. Goundar, M. Joye, A. Miyaji, M. Rivain, and A. Venelli. Scalar multiplication on weierstraß elliptic curves from co-Z arithmetic. *J. Cryptographic Engineering*, 1(2):161–176, 2011.

[20] M. Joye and S.-M. Yen. The montgomery powering ladder. In *CHES*, volume 2523 of *LNCS*, pages 291–302. Springer, 2002.

[21] N. Koblitz. A family of jacobians suitable for discrete log cryptosystems. In *CRYPTO*, volume 403 of *LNCS*, pages 94–99. Springer, 1988.

[22] P. Kocher. Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In *CRYPTO*, LNCS, pages 104–113. Springer, 1996.

[23] J. López and R. Dahab. Fast multiplication on elliptic curves over gf(2^{m}) without precomputation. In *CHES*, LNCS, pages 316–327. Springer, 1999.

[24] P. Montgomery. Speeding the Pollard and elliptic curve method of factorization. In *Mathematics of computation*, pages 243–264. Springer, 1987.

[25] C. Nègre and T. Plantard. Efficient modular arithmetic in adapted modular number system using Lagrange representation. In *13th Australasian conference on Information Security and Privacy*, pages 463–477. Springer, 2008.

[26] T. Plantard. *Modular arithmetic for cryptography(PhD in french)*. PhD thesis, LIRMM, Université Montpellier 2, 2005.

[27] T. Plantard, W. Susilo, and Z. Zhang. LLL for ideal lattices: re-evaluation of the security of gentry–halevi's fhe scheme. *Designs, Codes and Cryptography*, volume 76(n^o 2):325–344, 2015.

[28] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[29] E. Trichina and A. Bellezza. Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks. In *CHES*, LNCS, pages 98–113. Springer, 2002.