PMNS FOR LARGE INTEGERS : REVISITING THE INTERNAL MONTGOMERY REDUCTION

Nicolas Méloni, François Palma, Pascal Véron

Laboratoire IMath, Université de Toulon
La Garde, France

Abstract. The efficiency of the Polynomial Modular Number System (PMNS) has been highlighted in [2, 4], in the context of Elliptic Curve Cryptography over $\mathbb{F}_p$, with prime sizes in the 256 to 512-bit range. Unfortunately the performances of the current implementation decreases as the size of p increases. Moreover, the generation process of a PMNS has a worst case complexity of $\mathcal{O}(2^n)$ where n is the number of symbols used to represent an integer modulo p in this representation system. In this paper, we focus on the use of PMNS for large integers. We first implemented a 128-bit version of a PMNS. Our experimentations show that such an implementation is well suited for large integers compared to its 64-bit counterpart. Next, we modify an important step of the modular reduction process (called internal reduction). As a consequence, the current $\mathcal{O}(2^n)$ step of the generation process can be bypassed.

1. Introduction

Modular arithmetic operations are central to Public Key Cryptography systems. Reducing a number modulo a prime p is still costly currently and as such, to ensure fast computation, several systems have been developed to replace the long division with another operation altogether. The Polynomial Modular Number System (PMNS) is one such system.

Definition 1.1. Let $p \geqslant 3$, $n \geqslant 2$, $\gamma \in [1, p-1]$ and $\rho \in [1, p-1]$. Let $E \in \mathbb{Z}[X]$ a monic polynomial of degree n, such that $E(\gamma) \equiv 0 \pmod{p}$. A PMNS is a set $\mathcal{B} \subset \mathbb{Z}[X]$ such that :

(1) $\forall A \in \mathcal{B}$, $\deg(A) < n$,
(2) $\forall A(X) = \sum_{i=0}^{n-1} a_i X^i \in \mathcal{B}$, $-\rho < a_i < \rho$ for all i,
(3) $\forall a \in \mathbb{Z}/p\mathbb{Z}$, $\exists A \in \mathcal{B}$ such that $A(\gamma) \equiv a \pmod{p}$.

So A is a **representation** of a in $\mathcal{B}$ and we denote $A \equiv a_{\mathcal{B}}$. The tuple (p, n, γ, ρ, E) characterizes the PMNS $\mathcal{B}$. When $E(X) = X^n - \lambda$, with $\lambda \in \mathbb{Z} \setminus \{0\}$, $\mathcal{B}$ is also called AMNS (Adapted Modular Number System)[1].

To compute $a \odot b$, for a and $b \in \mathbb{Z}/p\mathbb{Z}$, and $\odot = \times$ or $+$, we proceed in 3 steps:

(1) compute a representative $A(X)$ of a and a representative $B(X)$ of b in $\mathcal{B}$,
(2) compute $C(X) = A(X) \odot B(X) \bmod E(X)$ (this step is called *external reduction* and $E(X)$ is called the external reduction polynomial),
(3) find a polynomial $\tilde{C}(X)$ such that $\deg C(X) = \deg \tilde{C}(X)$, $\tilde{C}(\gamma) \equiv C(\gamma) \bmod p$ and $|\tilde{c}_i| < \rho$, $\forall i$ (this step is called *internal reduction*).

Key words and phrases. Polynomial Modular Number System, Finite field, Modular arithmetic.

The internal reduction process is a critical step. Let

$$\mathbb{Z}_{n-1}[X] = \{A(X) \in \mathbb{Z}[X] \text{ such that } \deg(A(X)) \leqslant n-1\},$$

the classical algorithm used to perform this step is called the Montgomery like reduction algorithm and is described in [5] (see Alg. 1 in this paper). This algorithm depends on a parameter ϕ which for practical reasons is chosen as the size of the hardware words used in implementation, typically 2^{64}. The algorithm also mainly depends on the existence of a polynomial $M(X)$ of degree at most $n-1$ such that $M(\gamma) = 0 \bmod p$. The existence and the effective construction of this polynomial has been widely studied in [3, 4]. As mentioned in [4], this polynomial is computed from the elements of a reduced basis of the lattice $\mathfrak{L}$ where :

$$\mathfrak{L} = \{A(X) \text{ such that } \deg A(X) \leqslant n-1 \quad \text{and} \quad A(\gamma) = 0 \bmod p\} \quad \text{and} \quad \mathrm{vol}(\mathfrak{L}) = p\,.$$

A classical result in lattice theory states that the infinity norm of a "short" vector in $\mathfrak{L}$ is about $p^{1/n}$. Now from [4, Corollary 1], the parameters ρ and ϕ must satisfy the following bounds to guarantee the existence of the PMNS and the consistency of the Montgomery internal reduction :

$$\phi \geqslant 2w\rho(\delta+1)^2 \quad \text{and} \quad \rho \geqslant 2\|\mathcal{M}\|_1\,,$$

where :

- $w \geqslant \frac{3n}{2}$ for the class of external reduction polynomials studied in [4, Table 1],
- δ designates the number of "free additions" one can perform within the system before a multiplication without needing to perform an internal reduction,
- $\mathcal{M}$ is the internal reduction matrix defined as follows :

$$\mathcal{M} = \begin{pmatrix} m_0 & m_1 & \dots & m_{n-1} \\ \dots & \dots & \dots & \dots \\ \vdots & \vdots & & \vdots \\ \dots & \dots & \dots & \dots \end{pmatrix} \begin{matrix} \leftarrow M \\ \leftarrow X.M \bmod E \\ \\ \leftarrow X^{n-1}.M \bmod E \end{matrix}$$

As $\|\mathcal{M}\|_1 \geqslant \|M\|_\infty$, then the polynomial M must satisfy :

$$\|M\|_\infty \leqslant \frac{\phi}{4w(\delta+1)^2}\,.$$

This bound, and the classical bound on the infinity norm of a "short" vector in a lattice, depend on the integer n which has a direct impact on the performances of all the operations done in the PMNS. Hence, the smaller n is, the better the performances are. The optimal value for n on a 64-bit architecture is $n_{\text{opt}} = \lfloor \frac{\log_2 p}{64} \rfloor + 1$. Unfortunately for large primes (at least 1024 bits), to compute a polynomial M which satisfies the two preceding bounds for n_{opt} won't be successful in practice. As an example, let us suppose that we aim to build a PMNS for a 2048-bit integer, then $n_{\text{opt}} = 33$. For the class of efficient PMNS described in [4], and for $\delta = 0$, the two bounds give that :

$$\|\text{shortest vector of } \mathfrak{L}\|_\infty \simeq 2^{62} \qquad \text{and} \qquad \|M\|_\infty \leqslant \frac{2^{64}}{4w}\,,$$

where $w \geqslant \frac{3n_{\text{opt}}}{2} = 49.5$. This gives

$$\|M\|_\infty \leqslant \frac{2^{64}}{198} \simeq 2^{56.37}.$$

Hence lattice theory tells us that the infinity norm of the shortest vector in $\mathfrak{L}$ is about 2^{62} while we are searching in $\mathfrak{L}$ for a vector whose norm is at most $2^{56.37}$. To satisfy the two bounds, we have to choose a larger n than n_{opt}, specifically $n = 40$. The global performances of the PMNS suffer in consequence.
Another problem when generating a PMNS is that an exhaustive search among a set of 2^n polynomials in the lattice $\mathfrak{L}$ is done to find $M(X)$ (see [4, Algorithm 8]). As an example, it means that for a 4096-bit integer, if we aim to build a PMNS with $n = n_{\text{opt}} = 65$, an exhaustive search among a set of 2^{65} polynomials has to be done. Note that, as previously mentioned , in practice we will have to choose $n > n_{\text{opt}}$ to build a PMNS for a 4096-bit integer. Thus, the exhaustive search will actually cost more than 2^{65} iterations.

Algorithm 1 Coefficients reduction [5]

Require: $\mathcal{B} = (p, n, \gamma, \rho, E)$ a PMNS, $V \in \mathbb{Z}_{n-1}[X]$, $M \in \mathbb{Z}_{n-1}[X]$ such that $M(\gamma) \equiv 0 \pmod p$, $\phi \in \mathbb{N} \setminus \{0\}$ and $M' = -M^{-1} \bmod (E, \phi)$.
Ensure: $S(\gamma) = V(\gamma)\phi^{-1} \pmod p$, with $S \in \mathbb{Z}_{n-1}[X]$
1: $Q \leftarrow V \times M' \bmod (E,\ \phi)$
2: $T \leftarrow Q \times M \bmod E$
3: $S \leftarrow (V + T)/\phi$
4: return S

2. Adapting PMNS for large integers

2.1. **PMNS with $\phi = 2^{128}$.** To find a value n which is not too far from n_{opt} the value of ϕ can be increased so that the upperbound $\frac{\phi}{4w(\delta+1)^2}$ is greater than $p^{1/n}$. As the internal reduction algorithm involves a division by ϕ, a natural idea is to choose $\phi = 2^{128}$ instead of 2^{64}. As a consequence, the software implementation of the PMNS has to be upgraded in order to deal with 128 and 256-bit integer operations on polynomial coefficients. We developed a software implementation of those operations and did a comparison with the current 64-bit version of PMNS. The reference implementation with $\phi = 2^{64}$ makes use of internal compiler `__int128` registers which are 128-bit integer words handled directly by the compiler. This allows the algorithm to completely forgo any multiprecision considerations and leave it all to the compiler to be done internally in assembly. No such equivalent 256-bit register exists at the current time so multiprecision algorithms have to be used which slow down the calculations. Both Karatsuba and the Schoolbook version were evaluated and the Schoolbook version has been found to be slightly faster so far. This means that for any coefficient multiplication in our algorithm, we have to execute 4 distinct multiplication operations on 64-bit integers. With that said if we note n_{64} the "practical" n used for $\phi = 2^{64}$ for a given size of p and n_{128} the equivalent for $\phi = 2^{128}$, we find in practice that, for large integers, $n_{64} \geqslant 2n_{128}$ (see Table 1). In theory, this gives us complexities of $n_{64}^2 \geqslant 4n_{128}^2$ but in practice the 128-bit version has other operations dragging down the complexity function and giving us worse results until higher values of p widen the gap substantially between n_{64} and n_{128}. All the source code is available on https://github.com/francoispalma/PMNS.

size of p	256	512	1024	2048	4096	8192
n_{opt64}	5	9	17	33	65	129
n_{64}	5	10	19	40	84	188
n_{opt128}	3	5	9	17	33	65
n_{128}	3	5	9	18	36	73

TABLE 1. Optimal polynomial degrees vs practical ones used in implementation per size of prime considered.

size of p	256	512	1024	2048	4096	8192
$\phi = 2^{64}$	162	540	1947	8447	39665	229624
$\phi = 2^{128}$	340	1003	3236	12921	53056	207936

TABLE 2. Comparative table of performances for 64-bit vs 128-bit implementations in number of processor cycles for one modular multiplication on Intel processor i9-11900KF with gcc 11.3.0.

2.2. **Revisiting the internal Montgomery reduction.** Unfortunately, even if we are now able to find a suitable n for large integers, there is still a step in the generation process which involves an exhaustive search on a set of 2^n elements. For instance, for a 8192-bit integer, the value for n for $\phi = 2^{128}$ is 73.

We show in this section how we can bypass this exhaustive search by considering a modification on the internal reduction process. Let us first recall the main principle of the Montgomery modular reduction algorithm for integers. Let $c \in \mathbb{Z}$, $m \in \mathbb{N}$ (m odd), such that $2^{k-1} \leqslant m < 2^k (= \phi)$. To compute a representative of c in $\mathbb{Z}/m\mathbb{Z}$, the algorithm adds a multiple qm of m to c so that c and $c+qm$ are in the same equivalent class in $\mathbb{Z}/m\mathbb{Z}$. Now, q is chosen such that $c + qm$ is a multiple of ϕ. In other words, one computes q such that $c + qm = 0 \bmod \phi$. From this equality, one can deduce that $q = -m^{-1}c \bmod \phi$ and the algorithm outputs the value $(c+qm)/\phi$ which is a representative of $c\phi^{-1}$ in $\mathbb{Z}/m\mathbb{Z}$. C. Negre and T. Plantard generalized this idea in [5] to design an internal reduction process for the PMNS. Here, the main problem is that starting from two elements $A(X) \in \mathfrak{B}$ and $B(X) \in \mathfrak{B}$, the polynomial $C(X) = A(X)B(X) \bmod E(X)$ satisfies $C(\gamma) = A(\gamma)B(\gamma) \bmod p$, but there is no reason that $\|C\|_\infty < \rho$. To address this issue, the two authors add to the polynomial $C(X)$ a multiple $Q(X)M(X)$ of a polynomial $M(X)$ (modulo $E(X)$) where $M(X)$ satisfies $M(\gamma) \equiv 0 \bmod p$. Note that $C(X)$ and $C(X) + Q(X)M(X) \bmod E(X)$ have the same value when evaluated in γ modulo p. Now, $Q(X)$ is chosen such that all the coefficients of $C(X) + Q(M)M(X) \bmod E(X)$ are multiple of ϕ, hence $Q(X) = -M^{-1}(X)C(X) \bmod (E(X), \phi)$. The internal reduction algorithm outputs $(C(X) + Q(X)M(X) \bmod E(X))/\phi$ and conditions on ϕ are given so that $\|(C(X) + Q(X)M(X) \bmod E(X))/\phi\|_\infty < \rho$.

In the Montgomery multiplication algorithm, adding qm to c means adding a representative of 0 in $\mathbb{Z}/m\mathbb{Z}$. In the Montgomery internal reduction algorithm, adding $Q(X)M(X) \bmod E(X)$ is equivalent to adding a polynomial which vanishes in $\gamma \bmod p$ but with no guarantee that this element belongs to $\mathfrak{B}$. We propose to modify this step in order to add to $C(X)$ a representative $Z(X)$ of 0 in $\mathfrak{B}$ such that all coefficients of $C(X) + Z(X)$ are multiples of ϕ. Such a representative can easily be obtained from a reduced basis of the lattice $\mathfrak{L}$. Let $L_0(X), \ldots, L_{n-1}(X)$

be a reduced basis of $\mathfrak{L}$, and let $\mathcal{L}$ be the matrix where the i^{th} row contains the coefficients of $L_{i-1}(X)$. Let $C = (C_0, \ldots, C_{n-1})$ be the coefficients of $C(X)$. We aim to find a vector $q \in \mathbb{Z}^n$ such that $C + q\mathcal{L} = 0 \bmod \phi$. This equality gives $q = C(-\mathcal{L}^{-1}) \bmod \phi$. Note that, since $\text{vol}(\mathfrak{L})=p$, then $\det(\mathcal{L})=p$ which is coprime with ϕ for $\phi = 2^k$, hence $-\mathcal{L}^{-1} \bmod \phi$ always exists. Algorithm 2 sums up this new internal reduction. This algorithm does not need to precompute a specific polynomial $M(X)$ that might require an exhaustive search.

Algorithm 2 Coefficients reduction , new version

Require: $\mathcal{B} = (p, n, \gamma, \rho, E)$ a PMNS, $C \in \mathbb{Z}_{n-1}[X]$, $\mathcal{L}$ a reduced basis of $\mathfrak{L}$, $\phi \in \mathbb{N} \setminus \{0\}$ and $-\mathcal{L}^{-1} \bmod \phi$.
Ensure: $S(\gamma) = C(\gamma)\phi^{-1} \pmod p$, with $S \in \mathbb{Z}_{n-1}[X]$
1: $q \leftarrow C(-\mathcal{L}^{-1}) \bmod \phi$
2: $S \leftarrow (C + q\mathcal{L})/\phi$
3: return S

Proposition 2.1. *Let $C(X)$ be a polynomial of degree at most $n-1$, if*

$$\|C\|_\infty \leqslant \phi(\rho - \|\mathcal{L}\|_1)\,,$$

then the output S of Algorithm 2 (with C as input) is such that $\|S\|_\infty < \rho$ (i.e., $S \in \mathcal{B}$).

Proof. in full paper. □

References

1. J.-C. Bajard, Laurent Imbert, and Thomas Plantard, *Modular number systems: Beyond the mersenne family*, Selected Areas in Cryptography, 11th International Workshop, SAC 2004, Waterloo, Canada, 2004, pp. 159–169.
2. Titouan Coladon, Philippe Elbaz-Vincent, and Cyril Hugounenq, *MPHELL: A fast and robust library with unified and versatile arithmetics for elliptic curves cryptography*, ARITH 2021 (Torino, Italy), Transactions on Emerging Topics in Computing, June 2021.
3. Laurent-Stéphane Didier, Fangan Yssouf Dosso, and Pascal Véron, *Efficient modular operations using the Adapted Modular Number System*, Journal of Cryptographic Engineering (2020), 1–23.
4. Fangan Yssouf Dosso, Jean-Marc Robert, and Pascal Veron, *PMNS for Efficient Arithmetic and Small Memory Cost*, IEEE Transactions on Emerging Topics in Computing **10** (2022), no. 3, 1263 – 1277.
5. Christophe Negre and Thomas Plantard, *Efficient modular arithmetic in adapted modular number system using lagrange representation*, Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, 2008, pp. 463–477.

Université de Toulon, Institut de Mathématiques, Toulon, France
Email address: francois.palma@univ-tln.fr
Email address: nicolas.meloni@univ-tln.fr
Email address: pascal.veron@univ-tln.fr