Efficient modular arithmetic in Adapted Modular Number System using Lagrange representation

Christophe Negre[1] and Thomas Plantard[2]

[1] Team DALI, University of Perpignan, France.
[2] Centre for Information Security Research
School of Computer Science and Software Engineering
University of Wollongong, Australia.

Abstract. In 2004, Bajard, Imbert and Plantard introduced a new system of representation to perform arithmetic modulo a prime integer p, the Adapted Modular Number System (AMNS). In this system, the elements are seen as polynomial of degree $n-1$ with the coefficients of size $p^{1/n}$. The best method for multiplication in AMNS works only for some specific moduli p. In this paper, we propose a novel algorithm to perform the modular multiplication in the AMNS. This method works for any AMNS, and does not use a special form of the modulo p. We also present a version of this algorithm in *Lagrange Representation* which performs the polynomial multiplication part of the first algorithm efficiently using Fast Fourier Transform.

Keywords : Prime Field, Modular Multiplication, Modular Number System, Lagrange Representation.

1 Introduction

Several cryptographic applications like the Diffie-Hellman key exchange protocol [12], ECC [16,14], RSA [19] or pairing based protocol require efficient modular integer arithmetic. Specifically, for Diffie-Hellman key exchange the main operation is an exponentiation modulo a prime integer p: this operation is generally done using a chain of squaring and multiplication modulo p. For ECC, the main operation is the scalar multiplication which requires also a chain of additions and multiplications modulo a prime integer p.

The multiplication modulo p consists to multiply two integers A and B and after that to compute the remainder modulo p. The methods to perform this operation differ if the integer p has a special form or not. If p is arbitrary, the most used methods are the method of Montgomery [18] and the method of Barrett [9]. But the cost of these two methods is roughly equal to the cost of three integer multiplications.

When the integer p has a sparse binary representation [23] the reduction modulo p can be done really efficiently. This last case is, for now, the most

efficient, consequently standards recommends these types of prime integer [1]. On the other hand these types of prime are rare, and it thus interesting have efficient modular arithmetic modulo any prime.

Recently Bajard, Imbert and Plantard [6] proposed a new method to perform modular arithmetic by using a new representation of the elements. An integer A modulo p is expressed as $A = \sum_{i=0}^{n-1} a_i \gamma^i$ with $\gamma^n \equiv \lambda \mod p$ with λ a very small constant. The coefficients a_i are small relatively to p and γ (roughly $|a_i| \leq \rho \cong p^{1/n}$ and $\gamma \cong p$).

In this representation the multiplication of A and B is done in two steps: the first step consists to multiply the polynomials A and B in γ modulo $\gamma^n - \lambda$, the second step consists to reduce the coefficients.

In this paper, we will present a modified version of the multiplier of [6]. The initial proposition in [6] use lookup table which can't be used for big size modulus. Our approach is similar to Montgomery's [18,3] to perform the reduction of the coefficients. We add a multiple of the moduli p to kill the lower part of the coefficients of the polynomial product $C = A \times B \mod (\gamma^n - \lambda)$.

To use Fast Fourier Transform, to perform the polynomial multiplication, we slightly modify the first algorithm, and use a Lagrange approach to perform arithmetic modulo $(\gamma^n - \lambda)$. We then obtain an algorithm with a sub-quadratic complexity.

This article is organized as follows: in the first section we will briefly recall the AMNS representation, we will present our new multiplication in AMNS representation, and we will study the construction of the *shortest polynomial* which is required in the multiplication. After that, we will recall the Lagrange representation (LR) approach [5,4] to perform polynomial modular arithmetic, and present the Lagrange form of our algorithm. We conclude by a study of its cost and by a presentation of an implementation.

2 Modular Number System

2.1 Definition

Efficient arithmetic modulo a prime integer p is generally deeply related to the system of representation used to represent the elements. Generally integers are expressed as a sum $A = \sum_{i=0}^{n} a_i \beta^i$ where $0 \leq a_i < \beta$ (in practice β is often chosen as a power of 2). Here we are interested in integer multiplication modulo a prime integer p, and specifically for p of cryptographic size $2^{160} \leq p$.

We will use a modified version of this classical representation: the *Modular Number System* [6] to represent the elements modulo p.

Definition 1 (MNS [6]). *A Modular Number System (MNS) $\mathcal{B}$, is a quadruple (p, n, γ, ρ), such that for all positive integers $0 \leq a < p$ there exists a polynomial $A(X) = \sum_{=0}^{n-1} a_i X^i$ such that*

$$\begin{aligned} &A(\gamma) = a \bmod p, \\ &\deg(A(X)) < n, \\ &\|A\|_\infty < \rho. \end{aligned} \tag{1}$$

The polynomial $A(X)$ is a representation of a in $\mathcal{B}$.

The Modular Number System is a system of representation which includes the modulo p used in the modular arithmetic. Generally the MNS have a basis $\gamma \cong p$ and small coefficients $|a_i| < \rho \cong p^{1/n}$.

Example 1. In the table 1, we prove that the quadruplet $(17, 3, 7, 2)$ is a MNS.

Table 1. The elements of $\mathbb{Z}_{17}$ in $\mathcal{B} = MNS(17, 3, 7, 2)$

0	1	2	3	4	5
0	1	$-X^2$	$1 - X^2$	$-1 + X + X^2$	$X + X^2$

6	7	8	9	10	11
$-1 + X$	X	$1 + X$	$-X - 1$	$-X$	$-X + 1$

12	13	14	15	16
$-X - X^2$	$1 - X - X^2$	$-1 + X^2$	X^2	-1

In particular, we can verify that if we evaluate $(-1 + X + X^2)$ in γ, we have $-1 + \gamma + \gamma^2 = -1 + 7 + 49 = 55 \equiv 4 \bmod 17$. We have also $\deg(-1 + X + X^2) = 2 < 3$ and $\| - 1 + X + X^2\|_\infty = 1 < 2$.

The second definition of this section corresponds to a sub-family of the Modular Number System. We use the possibility to choose freely the basis γ to have advantageous properties for the modular arithmetic. That's why Bajard *et al.* said that these systems are adapted to the modular arithmetic: this is the *Adapted Modular Number System.*

Definition 2 (AMNS [6]). *A Modular Number System $\mathcal{B} = (p, n, \gamma, \rho)$ is called Adapted (AMNS) if there exists a small integer λ such that $\gamma^n = \lambda \bmod p$. We call E the polynomial $X^n - \lambda$. γ is a root of the polynomial E in $\mathbb{Z}/p\mathbb{Z}$: $E(\gamma) \equiv 0 \pmod p$. We also note $(p, n, \gamma, \rho)_E$ the Modular Number System (p, n, γ, ρ) which is adapted to the polynomial E.*

The difficulty in the construction of AMNS is to find an n-th roots of a fixed element λ in $\mathbb{Z}/p\mathbb{Z}$. Since p is prime the problem can be easily solved [11] (when such root exists) and in this paper we will focus on AMNS associated to p prime. If p were a composite number, for example an RSA number, the problem could be solved using the factorization of p. This means that the method presented in this paper, could be extended to multiply two integers modulo an RSA number which admits such n-th roots.

2.2 Multiplication in AMNS

As described in [6] the multiplication of two elements A and B in AMNS is done through the three following steps

1. Polynomial multiplication $C(X) = A(X) \times B(X)$.
2. Polynomial reduction $C'(X) = C(X) \mod E(X)$.
3. Coefficient reduction $R = CoeffRed(C')$: the coefficients of C' lie in the interval $]-n\rho^2\lambda, n\rho^2\lambda[$, they must be reduced such that they have absolute value smaller than ρ.

The first step can be done using usual methods: polynomial school-book , Karatsuba, or FFT methods. The second step is quite easy because of the form of E: we have only to add the lower part of C with λ times the high part of C to get C'. The last part, is for now the most complicated: in [7] Bajard, Imbert and Plantard proposed a method using look up table, the performance of such algorithm is not easy to evaluate, it depends on the size of the table, and the memory access delay.

Consequently some improvements need to be done to have efficient coefficient reduction and thus efficient multiplication in AMNS.

3 Novel AMNS Multiplication

In this section, we will present a new AMNS-multiplication algorithm. Let us fix an AMNS (p, n, γ, ρ) and $M(X)$ a polynomial such that $M(\gamma) = 0 \mod p$ and $\gcd(M, E) = 1$. As we will see later, M must be chosen in practice with small coefficients.

To perform the multiplication in the AMNS, we use a trick similar to Montgomery's method [18]. We will use the polynomial M to kill the lower part of the coefficients of the product $C = A \times B \mod E$. This method work as follows.

Algorithm 1: AMNS Multiplication (Polynomial version)

Input : $A, B \in \mathcal{B} = AMNS(p, n, \gamma, \rho)_E$ with $E = X^n - \lambda$
Data : M such that $M(\gamma) \equiv 0 \pmod p$
an integer m and $M' = -M^{-1} \bmod (E, m)$
Output: R such that $R(\gamma) = A(\gamma)B(\gamma)m^{-1} \bmod p$
begin
  $C \leftarrow A \times B \bmod E$;
  $Q \leftarrow C \times M' \bmod (E, m)$;
  $R \leftarrow (C + Q \times M \bmod E)/m$;
end

We remark that if we take $m = 2^k$, in the third step we add some multiple of the modulo p (i.e. $Q \times M$ is a multiple of p since $Q(\gamma)M(\gamma) \equiv 0 \mod p$) to annihilate the lest significant bit of the coefficients of C in the same way as in classical Montgomery Multiplication.

Let us check that Algorithm 1 is exact: we have to verify that $R(\gamma) = A(\gamma)B(\gamma)m^{-1} \bmod p$. We know that $E(\gamma) \equiv 0 \pmod p$ (See Definition 2), thus we have $C(\gamma) \equiv A(\gamma)B(\gamma) \bmod p$. We know also that $M(\gamma) \equiv 0 \pmod p$ thus we have

$$C(\gamma) + Q(\gamma)M(\gamma) \equiv C(\gamma) \equiv A(\gamma)B(\gamma) \bmod p$$

We now prove that the division by m is exact. This is equivalent to prove that $(C+Q\times M \bmod E) \equiv 0 \bmod m$. We have by definition that $Q \equiv Q \mod m$ and also that $Q = C \times P \bmod E$ and that $P = -M^{-1} \bmod E$. We obtain that

$$\begin{aligned} C + Q \times M \bmod E &\equiv (C + C \times (-M^{-1} \times M) \bmod E) \bmod m \\ &\equiv (C - C \bmod E) \bmod m \\ &\equiv 0 \bmod m \end{aligned}$$

as required. At the end, we have $R(\gamma) \equiv A(\gamma)B(\gamma)m^{-1} \bmod p$ since an exact division (the division by m) is equal to the multiplication by an inverse modulo p. □

At this step we know that the resulting polynomial R of the previous algorithm satisfies $R(\gamma) = A(\gamma)B(\gamma)m^{-1} \bmod p$, but we do not know whether it is expressed in the AMNS, i.e., when the coefficients of R are smaller than ρ. This is the goal of the following theorem.

Theorem 1. *Let $\mathcal{B} = AMNS(p, n, \gamma, \rho)_E$ an Adapted Modular Number System, M a polynomial of $\mathcal{B}$ such that $M(\gamma) \equiv 0 \pmod p$ and $\sigma = \|M\|_\infty$, and A, B two elements of $\mathcal{B}$, if we have ρ and an integer m such that $\rho > 2|\lambda|n\sigma$ and $m > 2|\lambda|n\rho$ then the polynomial R output by the Algorithm 1 with input $\mathcal{B}$,M,m,A and B is in the Adapted Modular Number System $\mathcal{B}$.*

Proof. From the Definition 1, the polynomial R is in the Modular Number System $\mathcal{B} = (p, n, \gamma, \rho)_E$, if $\deg R < n$ and if $\|R\|_\infty < \rho$. The fact that $\deg R < n$ is easy to see since all the computation in the Algorithm 1 are done modulo $E = X^n - \lambda$.

Thus we have only to prove that $\|R\|_\infty < \rho$. We first have the following inequalities

$$\begin{aligned} \|R\|_\infty &= \|A \times B + Q \times M \bmod E\|_\infty / m \\ &\leq |\lambda| n (\|A\|_\infty \|B\|_\infty + \|Q\|_\infty \|M\|_\infty)/m \\ &\leq |\lambda| n (\rho^2 + m\sigma)/m = |\lambda| n (\tfrac{\rho^2}{m} + \sigma) \end{aligned}$$

using that $\|A\|_\infty, \|B\|_\infty \leq \rho$.

But, by hypothesis, we have $\rho > 2|\lambda|n\sigma, \quad m > 2|\lambda|n\rho$. Thus if we use the fact that $m > 2|\lambda|n\rho$, we obtain:

$$\|R\|_\infty < |\lambda| n (\tfrac{\rho^2}{2|\lambda|n\rho} + \sigma) \leq \tfrac{\rho}{2} + |\lambda| n \sigma.$$

And with $\rho > 2|\lambda|n\sigma$, i.e., $\sigma < \frac{\rho}{2|\lambda|n}$, we get the required result

$$\|R\|_\infty < \frac{\rho}{2} + |\lambda| n \frac{\rho}{2|\lambda|n} \leq \frac{\rho}{2} + \frac{\rho}{2} = \rho.$$

An important remark on the Theorem 1 is that the length of the coefficients of the representation depends on the length $\|M\|_\infty$ of the polynomial M, specifically if σ is small then ρ can be also taken small. So now we will focus on the construction of such *short polynomial* M.

3.1 The Shortest Polynomial

To construct such polynomial we will use technique provided by lattice theory. Indeed the Modular Number System has an interesting link with lattice theory. We recall the definition of Lattice.

Definition 3 (Lattice).
A lattice $\mathcal{L}$ is a discrete sub-group of $\mathbb{R}^n$, or equivalently the set of all the integral combinations of $d \leq n$ linearly independent vectors over $\mathbb{R}$.

$$\mathcal{L} = \mathbb{Z}\,\boldsymbol{b}_1 + \cdots + \mathbb{Z}\,\boldsymbol{b}_d = \{\lambda_1 \boldsymbol{b}_1 + \cdots + \lambda_d \boldsymbol{b}_d \,:\, \lambda_i \in \mathbb{Z}\}.$$

The set of vector $\boldsymbol{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d)$ is called a basis *of $\mathcal{L}$.*

The lattice associated to an MNS is a subset of the polynomials $\mathbb{Z}[X]$ of degree $n-1$

$$\mathcal{L} = \{A \in \mathbb{Z}[X] \text{ such that } \deg A \leq n-1 \text{ and } A(\gamma) \equiv 0 \mod p\}.$$

It is easy to check that such set form a subgroup of $\mathbb{Z}_n[X] = \{Q \in \mathbb{Z}[X] \text{ with } \deg Q \leq n-1\} \cong \mathbb{Z}^n$. Indeed let $A, B \in \mathcal{L}$, then $A \pm B \in \mathcal{L}$ since $(A \pm B)(\gamma) \equiv A(\gamma) \pm B(\gamma) \equiv 0 \mod p$.

If we associate each polynomial in $\mathbb{Z}_n[X]$ a vector with entries in $\mathbb{Z}$, we get the following set vectors of the lattice $\mathcal{L}$

$$\boldsymbol{B} = \begin{pmatrix} p & 0 & 0 & 0 & \ldots & 0 \\ -\gamma & 1 & 0 & 0 & \ldots & 0 \\ -\gamma^2 & 0 & 1 & 0 & \ldots & 0 \\ \vdots & & & \ddots & & \vdots \\ -\gamma^{n-2} & 0 & 0 & \ldots & 1 & 0 \\ -\gamma^{n-1} & 0 & 0 & \ldots & 0 & 1 \end{pmatrix} \begin{matrix} \leftarrow p \\ \leftarrow X - \gamma \\ \leftarrow X^2 - \gamma^2 \\ \vdots \\ \leftarrow X^{n-2} - \gamma^{n-2} \\ \leftarrow X^{n-1} - \gamma^{n-1} \end{matrix}.$$

If we define by $\mathcal{L}'$ the lattice spanned by these n vectors, we can easily note that the vectors $\boldsymbol{b} \in \boldsymbol{B}$ are clearly linearly independent and thus the dimension of $\mathcal{L}'$ (and thus of $\mathcal{L}$) is equal to n: $\mathcal{L}$ and $\mathcal{L}'$ are full dimensional lattices.

In Algorithm 1 we need a polynomial M such that $M(\gamma) \equiv 0 \mod p$ and $\|M\|_\infty$ is small. This is related to the classical problem in lattice to find the shortest vector since, $M \in \mathcal{L}'$: the best choice for M is the shortest polynomial in $\mathcal{L}'$.

Definition 4 (Shortest Polynomial). *A polynomial M is called Shortest Polynomial of a MNS $\mathcal{B} = (p, n, \gamma, \rho)$ if we have*

$$\left.\begin{array}{l} M \neq 0 \\ M(\gamma) = 0 \bmod p \\ \deg(M) < n \end{array}\right\} \text{ and } \forall A \in \mathbb{Z}[X], \text{ if } \left\{\begin{array}{l} A \neq 0 \\ A(\gamma) = 0 \bmod p \\ \deg(A) < n \end{array}\right\} \text{ then } \|M\|_\infty \leq \|A\|_\infty \tag{2}$$

We note σ the length of M: $\sigma = \|M\|_\infty$.

In 1896 [17], Minkowski gave a bound for the length of the shortest vector of a lattice $\mathcal{L}$ for all norm, precisely in the case of the norm $\|\cdot\|_\infty$ the shortest vector v satisfies $\|v\|_\infty \leq |\det \mathcal{L}'|^{1/d}$ if $d = \dim \mathcal{L}$.

A straightforward consequence of the Theorem of Minkowski is the following corollary which gives an upper bound on $\sigma = \|M\|_\infty$ the length of the shortest polynomial.

Corollary 1. *If the polynomial M is the Shortest Polynomial of the MNS $\mathcal{B} = (p, n, \gamma, \rho)$, we have $\|M\|_\infty \leq p^{1/n}$.*

Proof. This is trivial if we note that $\det(\mathcal{L}') = p$.

For practical application we will need to compute efficiently an approximation of the shortest polynomial M of a given AMNS (only an approximation is sufficient since we only need an M with small $\|M\|_\infty$). There is several algorithm to compute such M (cf. [20,21,13,8]), but LLL [15] might be the most efficient in our case.

In practice, in actual computers, LLL could not compute an LLL basis (and thus the M) for lattices of dimension bigger than 250. This restrict the use of AMNS to small range of n, we will discuss the consequences of this fact in Section 6.

4 Improved AMNS Multiplication

The AMNS multiplication (Algorithm 1) requires several polynomial multiplications modulo $E = X^n - \lambda$. There is different strategies to perform this operation efficiently: the polynomial multiplication can be done with classical methods (schoolbook method, Karatsuba, Toom-Cook or FFT algorithm), followed by a reduction modulo E.

Here we will study a modified version of Algorithm 1 by using a Lagrange representation of the polynomials. Our method performs the polynomial multiplication and the reduction modulo E at the same time. We begin by a brief review on Lagrange representation [5].

4.1 Lagrange representation

The Lagrange representation represents a polynomial by its values at n points, the roots of $E = \prod_{i=1}^{n}(X - \alpha_i)$ modulo an integer m. In an arithmetic point of view, this is related to the Chinese Remainder Theorem which asserts that the following application is an isomorphism.

$$\begin{array}{ccc} \mathbb{Z}/m\mathbb{Z}[X]/(E) & \longrightarrow & \mathbb{Z}/m\mathbb{Z}[X]/(X-\alpha_1) \times \cdots \times \mathbb{Z}/m\mathbb{Z}[X]/(X-\alpha_n) \\ A & \longmapsto & (A \bmod (X-\alpha_1), \ldots, A \bmod (X-\alpha_n)) \,. \end{array} \tag{3}$$

We remark that the computation of $A \mod (X - \alpha_i)$ is simply the computation of $A(\alpha_i)$. In other words the image of $A(X)$ by the isomorphism (3) is nothing other than the multi-points evaluation of A at the roots of E.

Definition 5 (Lagrange representation). *Let $A \in \mathbb{Z}[X]$ with $\deg A < n$, and $\alpha_1, \ldots, \alpha_n$ be the n distinct roots modulo m of $E(X)$.*

$$E(X) = \prod_{i=1}^{r} (X - \alpha_i) \mod m$$

If $a_i = A(\alpha_i) \mod m$ for $1 \leq i \leq k$, the Lagrange representation (LR) of $A(X)$ modulo m is defined by $\mathrm{LR}(A(X), m) = (a_1, \ldots, a_n)$.

The advantage of the LR representation to perform operations modulo E is a consequence of the Chinese Remainder Theorem. Specifically the arithmetic modulo E in classical polynomial representation can be costly if E has a high degree, in LR representation this arithmetic is decomposed into n independent arithmetic units, each does arithmetic modulo a very simple polynomial $(X-\alpha_i)$. But arithmetic modulo $(X - \alpha_i)$ is the arithmetic modulo m since the product of two degree zero polynomials is just the product modulo m of the two constant coefficients.

4.2 Improved AMNS algorithm using Lagrange representation

Let us go back to the Algorithm 1 and let us see how to use Lagrange representation to perform polynomial arithmetic in each step of the algorithm.

In view to use Lagrange representation, we select two integers m_1 and m_2 such that the polynomial $E = (X^n - \lambda)$ splits in $\mathbb{Z}/m_i\mathbb{Z}[X]$

$$E = \prod_{i=1}^{n} (X - \alpha_i) \mod m_1, \quad E = \prod_{i=1}^{n} (X - \alpha'_i) \mod m_2.$$

We can then represent the polynomials A and B in Algorithm 1 in Lagrange representation modulo m_1 and m_2.

Notation 1 *We will use in the sequel the following notation : for a polynomial A of degree $n-1$ we will denote $\overline{A}$ the Lagrange representation in α_i modulo m_1 and $\overline{\overline{A}}$ the Lagrange representation in α'_i modulo m_2.*

In this situation we can do the following modification in the Algorithm 1:

- the computation of C in the Algorithm 1 can be done in Lagrange representation modulo m_1;
- the last step of the Algorithm 1 can be done in Lagrange representation modulo m_2, providing that $m_2 \geq 2\rho$.

We have to deal with some troubleshooting provided by this strategy. Indeed, at the end of the first step we only know $\overline{Q}$, but we do not know $\overline{\overline{Q}}$ which is required in the modified step 3 of the AMNS multiplication. So we must perform a *change of Lagrange representation* to compute $\overline{\overline{Q}}$ from $\overline{Q}$. Similarly, to get

a complete multiplication algorithm, we need to know the $\overline{R}$ at the end of the AMNS multiplication to get the Lagrange representation of R modulo m_1 and m_2.

Let us call $ChangeLR$ the routine which performs the change between two Lagrange representations. We will show later how this $ChangeLR$ works. For now we can set the Lagrange version of the Algorithm 1.

Algorithm 2: Lagrange-AMNS Multiplication

Input : $\overline{A}, \overline{\overline{A}}, \overline{B}, \overline{\overline{B}}$ the Lagrange representation modulo m_1 and m_2 of A and B
Data : $\overline{\overline{M}}$ the LR representation of the shortest polynomial M, $\overline{M'}$ the LR representation of $M' = -M^{-1} \mod E$.
Output: $\overline{R}, \overline{\overline{R}}$ such that $R \in \mathcal{B}$ and $R(\gamma) = A(\gamma)B(\gamma)m_1^{-1} \bmod p$
begin
  $\overline{Q} \leftarrow \overline{A} \times \overline{B} \times \overline{M'}$;
  $\overline{\overline{Q}} \leftarrow ChangeLR_{m_1 \to m_2}(\overline{Q}))$;
  $\overline{\overline{R}} \leftarrow (\overline{\overline{A}} \times \overline{\overline{B}}) + \overline{\overline{Q}} \times \overline{\overline{M}}) \times m_1^{-1}$;
  $\overline{R} \leftarrow ChangeLR_{m_2 \to m_1}(\overline{\overline{R}})$;
end

4.3 The change of Lagrange representation

Let us fix A a polynomial of degree $(n-1)$ and $\overline{A}$, $\overline{\overline{A}}$ its Lagrange representations modulo m_1 and m_2. The basic method to perform the change of representation from $\overline{A}$ to $\overline{\overline{A}}$ consists

1. to first reconstruct the polynomial form $A(X)$ from its Lagrange representation $\overline{A}$
2. secondly, to evaluate the polynomial $A(X)$ at the root of E modulo m_2.

- We first deal with the problem to compute the Lagrange representation $\overline{A}$ from the polynomial representation of A. Recall that $E = X^n - \lambda$ split totally modulo m, thus the roots α_j of E modulo m are of the form $\alpha_j = \mu\omega^j$ where μ is an arbitrary roots of E modulo m and ω is a primitive n-th roots.
  To compute $A(\mu\omega^j)$ for $j = 1, \ldots, n$ we first determine

$$\widetilde{A}(X) = A(\mu X) = \sum_{i=0}^{n-1} a_i \mu^i X^i.$$

  After that we get $\overline{A} = (\widetilde{A}(1), \widetilde{A}(\omega), \ldots, \widetilde{A}(\omega^{n-1})) = DFT(m, n, \widetilde{A}, \omega)$.
- For the reverse problem which consists to reconstruct the polynomial $A(X)$ from its Lagrange representation $\overline{A}$ we simply reverse the previous process:
  1. we first compute $\widetilde{A} = DFT^{-1}(m, n, \overline{A}, \omega)$,
  2. and after that $A(X) = \widetilde{A}(\mu^{-1}X) = \sum_{i=0}^{n-1} \widetilde{a}_i \mu^{-i} X^i$.

So now, by joining these two methods we get the overall algorithm to perform the change of Lagrange representation $\overline{A} \rightarrow \overline{\overline{A}}$.

Algorithm 3: ChangeLR

Input : $\overline{A}$
Output: $\overline{\overline{A}}$
$\widetilde{A} \leftarrow DFT^{-1}(m_1, n, \omega_1, \overline{A})$;
$A(X) \leftarrow A(\mu_1^{-1}X) \mod m_1$;
$\widetilde{A}(X) \leftarrow A(\mu_2 X) \mod m_2$;
$\overline{\overline{A}} \leftarrow DFT(m_2, n, \omega_2, \widetilde{A}(X))$;

Finally the change of the representation is mainly reduced to the computation of one DFT and one DFT^{-1}. This is really interesting when the integer n is a power of 2 since is in this case we can use the so-called Fast Fourier Transform which performs this efficiently. This algorithm compute the DFT using $\frac{n}{2}\log_2(n)$ multiplications modulo m and $n\log_2(n)$ additions modulo m. (see [24] for a complete presentation of this algorithm).

Example 2. In the table 2, we present an example of the Lagrange-AMNS multiplication for the prime $p = 247649$ and for the two elements A and B expressed in the AMNS

$$A = 236 + 176X - 66X^2 - 248X^3, \quad B = -199 + 122X + 73X^2 - 148X^3.$$

To verify that the result is exact, we have to build the polynomial form $R = -2 - 8X - 17X^2 + 9X^3$ and then we can easily check that $R(\gamma) = ABm_1^{-1} \mod p = 114760$.

See [7], for other needed operations in a AMNS.

5 Complexity evaluation and comparison

Let us now evaluate the cost of AMNS multiplication in Lagrange Representation. We evaluate the cost of the algorithm in term of the number of additions and multiplications modulo m_1 and m_2. We assume that that m_1 and m_2 have the same size (generally m_1 is bigger since $m_1 \geq 2\lambda\rho$ and $m_2 \geq 2\rho$). Consequently an operation modulo m_1 and m_2 is assumed to have the same cost. In the table below we give the cost of each step of the Lagrange AMNS multiplication and the cost of the overall algorithm, in the case n is a power of 2 and FFT is used in the *ChangeLR* routine.

Let us briefly compare our scheme with a strategy *Montgomery Multiplication using Schönage-Strassen for integer multiplication*, which seems to be the best strategy for large integer arithmetic. Recall that Montgomery algorithm has a cost of 3 integer multiplications of size $\cong p$.

Table 2. Example of AMNS Multiplication

AMNS/Lagrange System
$\mathcal{B} = (p = 247649, n = 4, \gamma = 106581, \rho = 2^8)$,
$m_2 = 2^8 + 1, m_1 = 2^{12} + 1,$
$E = X^4 + 1$
$E = \prod_{i=0}^{3}(X - \mu_1\omega_1^i) \mod m_2,$
$E = \prod_{i=0}^{3}(X - \mu_2\omega_2^i) \mod m_1.$
$M = -8 - 5X - 17X^2 + 11X^3$
$M' \bmod m_1 = 497 + 3175X + 338X^2 + 895X^3$

Entries		
	Lag.E, m_1	LagE, m_2
A	$(1548, 2454, 2767, 2369)$	$(203, 256, 213, 15)$
B	$(3419, 3148, 1430, 3498)$	$(209, 195, 187, 155)$
M		$(147, 245, 64, 26)$
M'	$(1838, 1504, 1450, 1293)$	

AMNS Multiplication	
Step 1.	$\overline{Q} = \overline{ABM'} = (2384, 2371, 1252, 1591)$
Step 2.	$\overline{\overline{Q}} = ChangeLR_{m_1 \to m_2}(\overline{Q}) = (23, 176, 248, 182)$
Step 3.	$\overline{\overline{R}} = (\overline{\overline{AB}} + \overline{\overline{QM}})m_1^{-1} = (13, 51, 210, 232)$
Step 4.	$\overline{\overline{Q}} = ChangeLR_{m_2 \to m_1}(\overline{\overline{R}}) = (3454, 1159, 2560, 1013)$

Table 3. Complexity of basic operations

Computation	# Multiplications	# Additions
$\overline{ABM'}$	n	0
$ChangeLR_{m_1 \to m_2}(\overline{Q})$	$n\log_2(n) + 2(n-1)$	$2n\log_2(n)$
$(\overline{AB} + \overline{\overline{QM}})m_1^{-1}$	$3n$	n
$ChangeLR_{m_2 \to m_1}(\overline{Q})$	$n\log_2(n) + 2(n-1)$	$2n\log_2(n)$
Total	$2n\log_2(n) + 6n - 2$	$4n\log_2(n) + n$

In Schönage-Strassen [22] integer a are expressed in the first step of the recursive algorithm as $a = \sum_{i=0}^{n-1} a_i X^i$ where $n \cong \log_2(p)/2$ and $a_i \leq p^{1/n}$.

Interpolation using FFT modulo an integer $m \cong p^{2/n}$ is done to compute $ab \mod X^{2n} - 1$. Thus each integer multiplication requires $3FFT$ (counting only the first step of the recursion) at $2n$ points with a modulo m with size $p^{2/n}$.

Consequently: we have $9FFT$ in $2n$ points computations at for the overall Montgomery algorithm, with coefficients size $p^{2/n}$ in FFT compared to $4FFT$ in n points with coefficient size $p^{1/n}$ for AMNS.

We must mention that in Schönage-Strassen products with roots of unity in the FFT has a cost of one addition because of the choice of m, but such strategy could also be also applied in AMNS.

6 Practical aspects and Implementation

Let us discuss some troubleshouting which can appear in the implementation of AMNS-Lagrange multiplication.

First of all, due to the discussion on the construction of M in Section 2, and the fact that n must be a power of 2 to have efficient *ChangeRep*, n must be taken be taken in the set $\{2, 4, 8, 16, 32, 64, 128\}$.

For these special values of n, we prove that we can always find for any prime p a integer γ which is a root of a polynomial $X^n - \lambda$ modulo p with λ not too big (see Lemma 1)

Lemma 1. *Let m be an odd integer, and n an integer such that there exits an integer $k > 0$ with $n = 2^k$, there exists a polynomial $X^n - \lambda$ such that:*

i) $X^n - \lambda$ is irreducible in $\mathbb{Z}$
ii) there exist a root γ of $X^n - \lambda$ in $\mathbb{Z}/m\mathbb{Z}$
iii) $|\lambda| \leq 2^{\frac{n}{2}}$

Proof. Let be g a generator of the group of the invertible of $\mathbb{Z}/p\mathbb{Z}$ and $\phi(p)$ be the length of this group.

We decompose $\phi(p)$ with a positive integer k_1 and an odd integer p_1 such that $\phi(p) = 2^{k_1} p_1$.

2 is invertible in $\mathbb{Z}/p\mathbb{Z}$, so there exist an integer i such that $g^{i \bmod \phi(p)} = 2 \bmod p$. We decompose i with an positive integer k_2 and an odd integer p_2 such that $i = 2^{k_2} p_2$.

p is odd, so we have $\phi(p)$ even ($k_1 \geq 1$). We also know that $g^{2^{k_1-1} p_1} = -1 \bmod p$.

We have now four case:

1. If $k < k_1$, we choose $\lambda = -1$ and $\gamma = g^x$ with $x = 2^{k_1-1-k} p_1$ and we have
   i) $X^n + 1$ is irreducible in $\mathbb{Z}$
   ii) $\gamma^n = g^{xn} = g^{2^{k_1-1-k} p_1 2^k} = g^{2^{k_1-1} p_1} = -1 \bmod p$
   iii) $|\lambda| = 1 < 2^{\frac{n}{2}}$

We can easily verify that x is an integer.

2. If $k \leq k_2$, we choose $\lambda = 2$ and $\gamma = g^x$ $x = 2^{k_2-k}p_2$, then we have
   i) $X^n - 2$ is irreducible in $\mathbb{Z}$
   ii) $\gamma^n = g^{xn} = g^{2^{k_2-k}p_2 2^k} = g^{2^{k_2}p_2} = \lambda \bmod p$
   iii) $|\lambda| = 2 \leq 2^{\frac{n}{2}}$

   We can easily verify that x is an integer.

3. If $k > k_2 \geq k_1$, we choose $\lambda = 2$ and $\gamma = g^x$ with $x = \frac{2^{k_2-k_1}p_2+yp_1}{2^{k_1-k}}$ with $y = -2^{k_2-k_1}p_2p_1^{-1} \bmod 2^{k-k_1}$
   i) $X^n - 2$ is irreducible in $\mathbb{Z}$
   ii) $\gamma^n = g^{xn} = g^{(2^{k_2-k_1}p_2+yp_1)2^{k_1-k}2^k} = g^{2^{k_2}p_2+yp_1 2^{k_1}} = 2$
   iii) $|\lambda| = 2 \leq 2^{\frac{n}{2}}$

   We verify that x is an integer, that s the case if $2^{k_2-k}p_2 + yp_1 2^{k_1-k}$ can be divide by 2^{k-k_1}.

$$\begin{aligned} 2^{k_2-k}p_2 + yp_1 2^{k_1-k} &\equiv (2^{k_2-k_1}p_2 + (-2^{k_2-k_1}p_2p_1^{-1} \bmod 2^{k-k_1})p_1) \bmod 2^{k-k_1} \\ &\equiv 2^{k_2-k_1}p_2 - 2^{k_2-k_1}p_2p_1^{-1}p_1) \bmod 2^{k-k_1} \\ &\equiv 2^{k_2-k_1}p_2 - 2^{k_2-k_1}p_2) \bmod 2^{k-k_1} \\ &\equiv 0 \bmod 2^{k-k_1} \end{aligned}$$

4. If $k \geq k_1 > k_2$ then we choose $\lambda = -2^{2^{k_1-k_2-1}}$ and $\gamma = g^x$ with $x = \frac{\frac{p_1+p_2}{2}+yp_1}{2^{k_1-k}}$ with $y = -\frac{p_1+p_2}{2}p_1^{-1} \bmod 2^{k-k_1}$
   i) $X^n + 2^{2^{k_1-k_2}}$ is irreducible in $\mathbb{Z}$
   ii) $\gamma^n = g^{xn} = g^{(\frac{p_1+p_2}{2}+yp_1)2^{k_1-k}2^k} = g^{(p_1+p_2)2^{k_1-1}+yp_1 2_1^k} = g^{(p_1 2^{k_1-1}+p_2 2^{k_1-1}+y\phi(p)}$
   $= -g^{p_2 2^{k_1-1}} = -g^{p_2 2^{k_2+k_1-k_2-1}} = -2^{2^{k_1-k_2-1}}$
   iii) $|\lambda| = 2^{2^{k_1-k_2-1}} \leq 2^{2^{k-1}} \leq 2^{\frac{n}{2}}$

   We verify that x is an integer, that's the case if $\frac{p_1+p_2}{2} + yp_1$ is divisible by 2^{k-k_1}

$$\begin{aligned} \tfrac{p_1+p_2}{2} + yp_1 &\equiv \tfrac{p_1+p_2}{2} + yp_1 \bmod 2^{k-k_1} \\ &\equiv \tfrac{p_1+p_2}{2} + (-\tfrac{p_1+p_2}{2}p_1^{-1})p_1 \bmod 2^{k-k_1} \\ &\equiv \tfrac{p_1+p_2}{2} - \tfrac{p_1+p_2}{2} \bmod 2^{k-k_1} \\ &\equiv 0 \bmod 2^{k-k_1} \end{aligned}$$

□

So, we can construct AMNS for all prime with good conditions on n and λ.

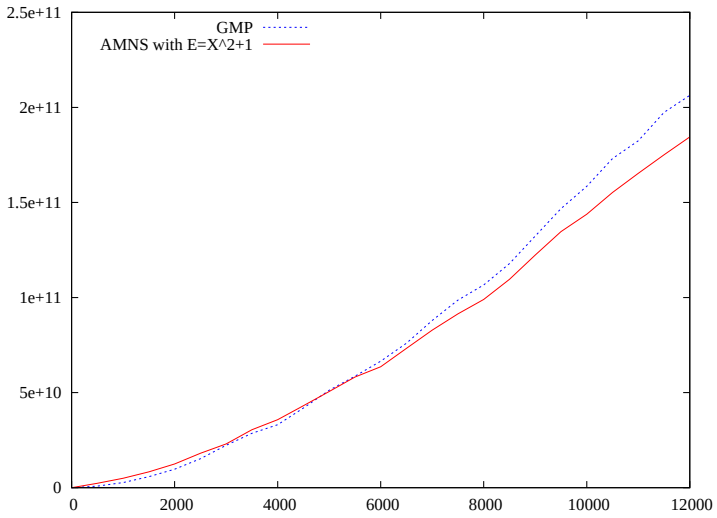
There is an alternative strategy on the drawback due to the restriction of the size of n : in AMNS-Lagrange multiplication if m_1 and m_2 are Fermat number, we can use Schönage-Strassen [10] method to perform arithmetic modulo m_1 and m_2, and keep the advantageous of the method.

Let us now present a result on the implementation of AMNS-LR multiplication. Figure 1 give the time in function of the modulus size of an implementation of Algorithm 2 in the special case $n = 2$ and $\lambda = -1$ on a Pentium 4, 2 GHz. The

case $n = 2$ and $\lambda = -1$ is an interesting case, since AMNS can be constructed for prime p when $p - 1$ is divisible by 4 (this is the case for 50% of prime p).

We compare this implementation of Algorithm 2 with GMP 4.2.1 [2] modular multiplication. For GMP, we use the modular multiplication of the modular exponentiation, to have its best one. Our implementation use also GMP [2] tools to construct AMNS-Lagrange system, and to perform the multiplication itself. Our code, could be highly optimized, for example, by using astuciously the fact that m_1 and m_2 are Fermat numbers.

Fig. 1. Comparison between GMP's modular multiplication and Algorithm 2 with $E = X^2 + 1$.



We can see that even if we don't use the advantageous form of Fermat moduli, Algorithm 2 begin to be faster when p have a size around 5000 bits. We expect that we could get better result with bigger n, since the complexity decrease with n.

7 Conclusion

In this paper we have presented a novel algorithm to perform integer modular arithmetic. Primarily, we gave a polynomial formulation of our algorithm which uses the AMNS [6] representation of integer and a Montgomery-like method to reduce the coefficients. Secondly we modify this algorithm in view to use a Lagrange representation to speed-up the polynomial multiplication part of the algorithm. From practical implementation, we expect that it should improve classical algorithm (Montgomery, Barrett) to perform modular multiplication modulo arbitrary for prime p of size several thousand bits.

References

1. *FIPS PUB 197: Advanced Encryption Standard (AES).* FIPS PUB. NIST 2001.
2. The GNU Multiple Precision arithmetic librairy, May 2006.
3. J.-C. Bajard, L.-S. Didier, and P. Kornerup. An RNS Montgomery modular multiplication algorithm. *IEEE Transactions on Computers*, 47:766–776, 1998.
4. J.-C. Bajard, L. Imbert, and C. Nègre. Arithmetic operations in finite fields of medium prime characteristic using the lagrange representation. *IEEE Trans. Computers*, 55(9):1167–1177, 2006.
5. J.-C. Bajard, L. Imbert, C. Negre, and T. Plantard. Efficient multiplication in gf(p^k) for elliptic curve cryptography. In *ARITH'16: IEEE Symposium on Computer Arithmetic*, pages 181–187, June 2003.
6. J.-C. Bajard, L. Imbert, and T. Plantard. Modular number systems: Beyond the Mersenne family. In *SAC 2004, LNCS 3357, Springer-Verlag*, pages 159–169, 2004.
7. J.-C. Bajard, L. Imbert, and T. Plantard. Arithmetic operations in the polynomial modular number system. In *ARITH'17: IEEE Symposium on Computer Arithmetic*, June 2005.
8. A. H. Banihashemi and A. K. Khandani. On the complexity of decoding lattices using the Korkin-Zolotarev reduced basis. *IEEE Transactions on Information Theory*, 44(1):162–171, 1998.
9. P. Barrett. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In *CRYPTO '86,LNCS* 263, 311–326. Springer-Verlag, 1986.
10. G. Brassard, S. Monet, and D. Zuffellato. Algorithms for very large integer arithmetic. *Tech. Sci. Inf.*, 5(2):89–102, 1986.
11. H. Cohen. *A course in computational algebraic number theory*, *Grad. Texts Math.* 138. Springer 1993.
12. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. on Inf. Theory*, IT-22(6):644–654, nov 1976.
13. R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.
14. N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, January 1987.
15. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen, Springer-Verlag*, 261:513–534, 1982.
16. V. Miller. Use of elliptic curves in cryptography. In *CRYPTO'85*, *LNCS* 218, 417-426. Springer, 1986.
17. H. Minkowski. *Geometrie der Zahlen.* B. G. Teubner, Leipzig, 1896.
18. P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, Apr 1985.
19. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Com. of the ACM*, 21(2):120–126, Feb 1978.
20. C.-P. Schnorr. Block Korkin-Zolotarev bases and successive minima, 1996.
21. C.-P. Schnorr. Fast LLL-type lattice reduction. *Information and Computation*, 204(1):1–25, 2006.
22. A. Schonhage and V. Strassen. Schnelle multiplikation grosser zahlen. *Computing*, 7:281–292, 1971.
23. J. Solinas. Generalized Mersenne numbers. Research Report CORR-99-39, Center for Applied Cryptographic Research, University of Waterloo, Canada, 1999.
24. J. von zur Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, 2003.