Modular Number Systems: Beyond the Mersenne Family

Jean-Claude Bajard, Laurent Imbert, Thomas Plantard

Modular Number Systems: Beyond the Mersenne Family

Jean-Claude Bajard[1], Laurent Imbert[2], and Thomas Plantard[1]

[1] LIRMM, CNRS UMR 5506
161 rue Ada, 34392 Montpellier cedex 5, France
{bajard,plantard}@lirmm.fr

[2] CNRS, France and ATIPS, CISaC, University of Calgary
2500 University drive N.W, Calgary, T2N 1C2, Canada
Laurent.Imbert@lirmm.fr

Abstract. In SAC 2003, J. Chung and A. Hasan introduced a new class of specific moduli for cryptography, called the more generalized Mersenne numbers, in reference to J. Solinas' generalized Mersenne numbers proposed in 1999. This paper pursues the quest. The main idea is a new representation, called Modular Number System (MNS), which allows efficient implementation of the modular arithmetic operations required in cryptography. We propose a modular multiplication which only requires n^2 multiplications and $3(2n^2 - n + 1)$ additions. Our solution is thus more efficient than Montgomery for a very large class of numbers that do not belong to the large Mersenne family.

Keywords: Generalized Mersenne numbers, Montgomery multiplication, Elliptic curve cryptography

1 Introduction

Efficient implementation of modular arithmetic is an important prerequisite in today's public-key cryptography [6]. In the case of elliptic curves defined over prime fields, operations are performed modulo prime numbers whose size range from 160 to 500 bits [4].

For moduli p that are not of special form, Montgomery [7] or Barrett [1] algorithms are widely used. However, modular multiplication and reduction can be accelerated considerably when the modulus p has a special form. Mersenne numbers of the form $2^m - 1$ are well known examples, but they are not useful

for cryptography because there are only a few primes (the first Mersenne primes are $3, 7, 31, 127, 8191, 131071, 524287, 2147483647$, etc). Pseudo-Mersenne of the form $2^m - c$, introduced by R. Crandall in [3], allow for very efficient modular reduction if c is a small integer. In 1999, J. Solinas [8] introduced the family of generalized Mersenne numbers. They are expressed as $p = f(t)$, where t is a power of 2 and lead to very fast modular reduction using only a few number of additions and subtractions. For example, the five NIST primes listed bellow, recommended in the FIPS 186-2 standard for defining elliptic curves over primes fields, belong to this class[3].

$$
\begin{aligned}
p_{192} &= 2^{192} - 2^{64} - 1 \\
p_{224} &= 2^{224} - 2^{96} + 1 \\
p_{256} &= 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1 \\
p_{384} &= 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1 \\
p_{521} &= 2^{521} - 1
\end{aligned}
$$

In 2003, J. Chung and A. Hasan, in a paper entitled "more generalized Mersenne numbers" [2], extended J. Solinas' concept, by allowing any integer for t.

In this paper we further extend the idea of defining new classes of numbers (possibly prime), suitable for cryptography. However, the resemblance with the previous works ends here. Instead of considering numbers of special form, we represent the integers modulo p in the so-called Modular Number System (MNS). By a careful choice of the parameters which define our MNS, we introduce the concept of Adapted Modular Number System (AMNS). We propose a modular multiplication algorithm which is more efficient than Montgomery's algorithm, and we explain how to define suitable prime moduli for cryptography. We provide examples of such numbers at the end of the paper.

2 Modular Number Systems

In positional number systems, we represent any nonnegative integer X in base β as

$$X = \sum_{i=0}^{k-1} d_i \, \beta^i, \tag{1}$$

where the digits d_is belong to the set $\{0, \ldots, \beta - 1\}$. If $d_{k-1} \neq 0$, we call X a k-digit base-β number.

[3] p_{521} is also a Mersenne prime.

In cryptographic applications, computations have to be done over finite rings or fields. In these cases, we manipulate representatives of equivalence classes modulo P (for simplicity we use the set of positive integers $\{0, 1, 2, \ldots, P-1\}$), and the operations are performed modulo P.

In the next definition, we extend the notion of positional number system to represent the integers modulo P.

Definition 1 (MNS). *A Modular Number System (MNS) $\mathcal{B}$ is defined according to four parameters (γ, ρ, n, P), such that all positive integers $0 \leq X < P$ can be written as*

$$X = \sum_{i=0}^{n-1} x_i \, \gamma^i \bmod P, \tag{2}$$

with $1 < \gamma < P$, and $x_i \in \{0, \ldots, \rho - 1\}$. The vector $(x_0, x_1, \ldots, x_{n-1})_{\mathcal{B}}$ denotes the representation of X in $\mathcal{B}$.

In the sequel of the paper, we shall omit the subscript $(.)_{\mathcal{B}}$ when it is clear from the context, and we shall consider X either as a vector or as a polynomial (in γ). In the later, the x_is correspond to the coefficients of the polynomial (note that we use a left-to-right notation; x_0 is the constant term).

Example 1. Let us consider the MNS defined with $\gamma = 7, \rho = 3, n = 4, P = 17$. Over this system, we represent the elements of $\mathbb{Z}_{17}$ as polynomials in γ of degree at most 3 with coefficients in $\{0, 1, 2\}$ (cf. table 1).

Table 1. The elements of $\mathbb{Z}_{17}$ in $\mathcal{B} = MNS(7, 3, 4, 17)$.

0	1	2	3	4	5	6	7	8
0	1	2	γ^3	$1+\gamma^3$	$2+\gamma^3$	$2\gamma^3$	γ	$1+\gamma$
9	10	11	12	13	14	15	16	
$2+\gamma$	$\gamma+\gamma^3$	$1+\gamma+\gamma^3$	$2+\gamma+\gamma^3$	$2\gamma^2$	$1+2\gamma^2$	γ^2	$1+\gamma^2$	

Definition 2 (AMNS). *A modular number system $\mathcal{B} = MNS(\gamma, \rho, n, P)$ is called Adapted Modular Number System (AMNS) if $\gamma^n \bmod P = c$ is a small integer. In this case we shall denote $\mathcal{B} = AMNS(\gamma, \rho, n, P, c)$.*

Although c is given by $\gamma^n \bmod P$, we introduce it in the AMNS definition to simplify the notations.

In the rest of the paper, we shall consider $\mathcal{B} = AMNS(\gamma, \rho, n, P, c)$, unless otherwise specified.

3 Modular multiplication

As in [2], modular multiplication is performed in three steps presented in algorithm 1.

Algorithm 1 – Modular Multiplication

Input : An AMNS $\mathcal{B} = (\gamma, \rho, n, P, c)$, and $A = (a_0, ..., a_{n-1})$, $B = (b_0, ..., b_{n-1})$

Output : $S = (s_0, ..., s_{n-1})$ such that $S = A\,B \bmod P$

1: Polynomial multiplication in $\mathbb{Z}[X]$: $U(X) \leftarrow A(X)\,B(X)$

2: Polynomial reduction: $V(X) \leftarrow U(X) \bmod (X^n - c)$

3: Coefficient reduction: $S \leftarrow CR(V)$, gives $S \equiv V(\gamma) \pmod P$

In order to evaluate the computational complexity of algorithm 1, let us get into more details. In the first step we evaluate

$$U(X) = \sum_{i=0}^{2n-2} u_i\,X^i, \text{ where } u_i = \sum_{j=0}^{i} a_i\,b_{i-j}, \tag{3}$$

where $a_t = b_t = 0$ for $t > n-1$. We have $u_0 = a_0 b_0 < \rho^2$, $u_1 = a_0 b_1 + a_1 b_0 < 2\rho^2$, etc. Clearly, the largest coefficient is $u_{n-1} < n\rho^2$. Then, for the coefficients of degree greater than $n-1$, we have $u_n < (n-1)\rho^2, \ldots, u_{2n-2} < \rho^2$.

The cost of the first step clearly depends on the size of ρ and n. It requires n^2 products of size $\log_2(\rho)$, and $(n-1)^2$ additions of size at most $\log_2(n\rho^2)$.

In step 2, we compute

$$V(X) = \sum_{i=0}^{n-1} v_i\,X^i, \text{ where } v_i = u_i + c\,u_{i+n}. \tag{4}$$

This yields

$$v_i < cn\rho^2, \quad \text{for } i = 0 \ldots n-1. \tag{5}$$

The cost of step 2 is $(n-1)$ products between the constant c and numbers of size $\log_2(n\rho^2)$, and $(n-1)$ additions of size $\log_2(cn\rho^2)$. When c is a small constant, for example a power of 2, the $(n-1)$ products can be implemented with only $(n-1)$ shifts and additions.

In order to get a valid AMNS representation we must reduce the coefficients such that all the v_is are less than ρ. This is the purpose of the coefficient reduction.

3.1 Coefficient reduction

For simplicity, we define $\rho = 2^{k+1}$. We reduce the elements of the vector V, obtained after step 2 of algorithm 1, by iteratively applying algorithm 2, presented bellow, which reduces numbers of size $\lceil \frac{3k}{2} \rceil$ bits to numbers of size $k+1$, i.e. less than ρ.

So, let us first consider a vector V with elements of size at most $\lceil \frac{3k}{2} \rceil$ bits. Our goal is to find a representation of V where the elements are less than ρ, i.e. of size at most $k+1$ bits.

If $V = (v_0, \ldots, v_{n-1})$, we define two vectors $\underline{V}$ and $\overline{V}$ such that

$$V = \underline{V} + \overline{V} \cdot 2^k I, \tag{6}$$

where the elements of $\underline{V}$ are less than 2^k and those of $\overline{V}$ are less than $2^{\lceil k/2 \rceil}$. In equation (6), I denotes the $n \times n$ identity matrix explicitly given by $I_{ij} = \delta_{ij}$ for $i, j = 0, \ldots, n-1$ and δ_{ij} is the Kronecker delta.

If we can express $\overline{V} \cdot 2^k I$ as a vector with elements less than 2^k, then the sum of the two vectors in (6) is less than 2^{k+1}, and gives a valid AMNS representation of V. The idea is to find a matrix M with small coefficients which satisfies

$$M \cdot (1, \gamma, \ldots, \gamma^{n-1})^T \equiv 2^k I \cdot (1, \gamma, \ldots, \gamma^{n-1})^T \pmod{P}. \tag{7}$$

Roughly speaking, the matrix M can be seen as a representation of $2^k I$ in the AMNS.

If $2^k = (\xi_0, \ldots, \xi_{n-1})_{\mathcal{B}}$, is a representation of 2^k in the AMNS, then by definition 1 we have

$$2^k \equiv \xi_0 + \xi_1 \gamma + \cdots + \xi_{n-1} \gamma^{n-1} \pmod{P}. \tag{8}$$

Similarly the following congruences hold:

$$\gamma\, 2^k \equiv c\xi_{n-1} + \xi_0 \gamma + \cdots + \xi_{n-2} \gamma^{n-1} \pmod{P} \tag{9}$$

$$\gamma^2\, 2^k \equiv c\xi_{n-2} + c\xi_{n-1}\gamma + \xi_0 \gamma^2 + \cdots + \xi_{n-3} \gamma^{n-1} \pmod{P} \tag{10}$$

$$\vdots$$

$$\gamma^{n-1}\, 2^k \equiv c\xi_1 + c\xi_2 \gamma + \cdots + c\xi_{n-1} \gamma^{n-2} + \xi_0 \gamma^{n-1} \pmod{P}. \tag{11}$$

Equations (8) to (11) allow us to define the matrix

$$M = \begin{pmatrix} \xi_0 & \xi_1 & \cdots & \xi_{n-1} \\ c\xi_{n-1} & \xi_0 & \cdots & \xi_{n-2} \\ \vdots & & & \\ c\xi_1 & c\xi_2 & \cdots c\xi_{n-1} & \xi_0 \end{pmatrix} \tag{12}$$

which satisfies equation (7). Thus $\overline{V} \cdot 2^k I \equiv \overline{V} \cdot M \pmod{P}$, and equation (6) becomes

$$V = \underline{V} + \overline{V} \cdot M. \tag{13}$$

If we impose $c\sum_{i=0}^{n-1} \xi_i < 2^{\lfloor k/2 \rfloor}$, then the elements of the vector $\overline{V} \cdot M$ are less than 2^k. Algorithm 2 implements equation (13) to reduce the elements of V to a valid AMNS representation, i.e. with $v_i < \rho = 2^{k+1}$ for $i = 0 \dots n-1$.

Algorithm 2 – Red$(V, \mathcal{B})$: reduction from $\lceil \frac{3k}{2} \rceil$ to $k+1$ bits

---

Input : $\mathcal{B} = (\gamma, \rho, n, P, c)$ an AMNS with $\rho = 2^{k+1}$; $2^k = (\xi_0, ..., \xi_{n-1})$ with $c\sum_{i=0}^{n-1} \xi_i < 2^{\lfloor k/2 \rfloor}$; a matrix M as defined in (12); a vector $V = (v_0, \dots, v_{n-1})$ with $v_i < 2^{\lceil 3k/2 \rceil}$ for $i = 0 \dots n-1$.

Output : $S = (s_0, ..., s_{n-1})$ with $s_i < \rho$ for all $i = 0 \dots n-1$.

1: Define vectors $\underline{V}$ and $\overline{V}$ such that $V = \underline{V} + \overline{V} \cdot 2^k I$

2: Compute $S \leftarrow \underline{V} + \overline{V} \cdot M$

The cost of algorithm 2 is n^2 multiplications of size $\frac{k}{2}$ and n additions of size k. However, since the M_{ij} in (12) are small constants, the n^2 products can be efficiently computed with only a small number of additions and shifts. For example, if c and the ξ_is are small powers of 2, then we can evaluate $\overline{V} \cdot M$ with $n(n-1)$ additions. In this case the total cost for Red is n^2 additions of size k.

In order to reduce polynomials with coefficients larger than $\frac{3k}{2}$ bits, we iteratively apply the previous algorithm until all the coefficients are less than ρ. The following theorem holds.

Theorem 1. *Let us define $\mathcal{B} = AMNS(\gamma, \rho, n, P, c)$, with $\rho = 2^{k+1}$. We denote $(\xi_0, \dots, \xi_{n-1})$ a representation of 2^k in $\mathcal{B}$, and we assume $V = (v_0, \dots, v_{n-1})$ with $v_i < cn\rho^2$.*

If $c\sum_{i=0}^{n-1} \xi_i < 2^{\lfloor k/2 \rfloor}$ then there exists an algorithm which reduces V into a valid AMNS representation, in $\frac{k+2+\lfloor \log_2(cn) \rfloor}{\lceil k/2 \rceil - 1}$ calls to Red (algorithm 2).

Proof. After step 2 of algorithm 1, and under the condition $\rho = 2^{k+1}$, the elements of V satisfy $v_i < 2^{2k+2}cn$. Thus $|v_i| < 2k + 3 + \lfloor \log_2(cn) \rfloor$, where $|v_i|$ denotes the size of v_i. Since each step of Red eliminates $\lceil \frac{k}{2} \rceil - 1$ bits of v_i, the number of iteration is given by the value t which satisfy the equation $2k + 3 + \lfloor \log_2(cn) \rfloor - t\left(\lceil \frac{k}{2} \rceil - 1\right) = k + 1$, i.e. $t = \frac{k+2+\lfloor \log_2(cn) \rfloor}{\lceil k/2 \rceil - 1}$. This gives $t = 2 + \frac{8+2\lfloor \log_2(cn) \rfloor}{k-2}$ is k is even, and $t = 2 + \frac{6+2\lfloor \log_2(cn) \rfloor}{k-1}$ if k is odd. □

Note that in practice, the number of iterations is very small. In the examples of section 5, the coefficient reduction step only requires 3 or 4 calls to algorithm Red. Algorithm 3 implements theorem 1.

Algorithm 3 – CR($V, \mathcal{B}$), Coefficient reduction

Input : $\mathcal{B} = (\gamma, \rho, n, P, c)$ an AMNS with $\rho = 2^{k+1}$; $2^k = (\xi_0, ..., \xi_{n-1})$ with $c\sum_{i=0}^{n-1} \xi_i < 2^{\lfloor k/2 \rfloor}$; a vector $V = (v_0, ..., v_{n-1})$.

Output : $S = (s_0, ..., s_{n-1})$ with $s_i < \rho$ for all $i = 0 \dots n-1$.

1: $l \leftarrow \max(\lfloor \log_2(v_i) \rfloor + 1)$
2: $U \leftarrow V$
3: **while** $l > \frac{3k}{2}$ **do**
4: Define $\underline{U}$ and $\overline{U}$ s.t. $U = \underline{U} + 2^{l-3k/2} \cdot \overline{U}$
5: $\overline{U} \leftarrow Red(\overline{U}, \mathcal{B})$
6: $U \leftarrow \underline{U} + 2^{l-3k/2} \cdot \overline{U}$
7: $l \leftarrow \max(\lfloor \log_2(U_i) \rfloor + 1)$
8: **end while**
9: $S \leftarrow Red(U, \mathcal{B})$

4 Complexity comparisons

In this section, we evaluate the number of elementary operations (word-length multiplications and additions) of our modular multiplication algorithm. Since the complexity of our algorithms clearly depends on many parameters we try to consider different interesting options. For simplicity, we assume $cn < \rho$ as this is the case in the examples presented in the next section.

As explained in section 3, our modular multiplication requires three steps: polynomial multiplication, polynomial reduction, and coefficient reduction.

The polynomial multiplication only depends on n and ρ. If $\rho = 2^{k+1}$ ($k + 1$ is the word-size), the cost of the first step is $n^2\, T_m$, where T_m is the delay of one

word-length multiplication, and $(n-1)^2$ additions involving two-word operands, i.e. of cost less than $3(n-1)^2\,T_a$, where T_a is the delay of one word-length addition. Thus, the cost of step 1 is

$$n^2\,T_m + 3(n-1)^2\,T_a.$$

The polynomial reduction depends on n, ρ, and c. In the general case, it requires $(n-1)$ multiplications of size $\log_2(n\rho^2)$. However, if $c = 1, 2, 4$ (resp. $c = 3, 5, 6$) it can be implemented in n shift-and-add of three-word numbers (resp. $2n$ shift-and-add). This yields a cost of $3n\,T_a$ (resp. $6n\,T_a$). Thus, for the second step, a careful choice of c can lead to

$$3n\,T_a.$$

The coefficient reduction depends on all the parameters. The cost of algorithm Red, for $\xi_i = 0, 1, 2$ and $c = 1, 2, 4$ is $n^2\,T_a$ (it becomes $n^2 + \frac{(n-1)^2}{2}\,T_a$ if $c = 3, 5, 6$). From theorem 1, algorithm CR requires 3 calls to Red if $cn < 2^{(k-10)/2}$ (4 calls if $cn < 2^{k-10}$). Finally, step 3 requires $3n^2\,T_a$ if $cn < 2^{(k-10)/2}$, $\xi_i = 0, 1, 2$, and $c = 1, 2, 4$ (we have $8n^2\,T_a$ if $cn < 2^{k-10}$, $\xi_i = 0, 1, 2$, $c = 3, 5, 6$). As for the previous step, a good choice of c and the ξ_is gives a complexity of

$$3n^2\,T_a.$$

The important point here is that we can perform the coefficient reduction without multiplications.

To summarize, our algorithm performs the modular multiplication, where the moduli do not belong to the Mersenne family, in

$$n^2\,T_m + 3\left(2n^2 - n + 1\right)\,T_a.$$

This is better than Montgomery which requires $2n^2\,T_m$ (cf. [7], [5]).

In the next section we explain how we define such modulus and we give examples that reach this complexity.

5 Construction of suitable moduli

In this section we explain how to find γ and P which allow fast modular arithmetic.

Let us first fix some of the parameters. Since we represent numbers as polynomials of coefficients less than $\rho = 2^{k+1}$, it is advantageous to define ρ according

to the word-size of the targeted architecture, i.e. by taking $k = 15, 31, 63$ for 16-bit, 32-bit, and 64-bit architectures respectively. We define n such that $(k+1)n$ roughly corresponds to the desired dynamic range. To get a very efficient reduction of the coefficients, we impose restrictions on the ξ_is, for example by only allowing values in $\{0, 1, 2\}$, and we choose very small values for c. Based on the previous choices, we now try to find suitable P and γ.

From equation (7), we deduce $V \cdot (2^k I - M) \equiv 0 \pmod{P}$, for all $V = (v_0, \ldots, v_{n-1})_{\mathcal{B}}$. Thus, it is clear that the determinant

$$d = \left|2^k I - M\right| \equiv 0 \pmod{P}. \tag{14}$$

All the divisors of d, including d itself, can be chosen for P. If we need P to be prime, we can either try to find a prime factor of the determinant which is large enough (this is easier than factorization since it suffices to eliminate the small prime factors up to an arbitrary bound), or consider only the cases where the determinant is already a prime.

We remark that γ is a root, modulo P, of both $\gamma^n - c$ and $2^k - \sum_{i=0}^{n-1} \xi_i \gamma^i$. Thus γ is also a root of $\gcd(\gamma^n - c, 2^k - \sum_{i=0}^{n-1} \xi_i \gamma^i) \bmod P$.

5.1 Generating primes for cryptographic applications

For elliptic curve defined over prime fields, P must be a prime of size at least 160 bits.

Let us assume a 16-bit architecture. we fix $\rho = 2^{16}$, and we see if we can generate good primes P with $n = 11$. Note that $nk = 176$ does not guaranty 176-bit primes for P. In practice, the candidates we obtain are slightly smaller. We impose strong restrictions on the other parameters, by allowing only $\xi_i \in \{0, 1\}$, and $2 \leq c \leq 6$.

As an example, we consider $c = 3$, and $2^k = (1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1)_{\mathcal{B}}$, which correspond to the polynomial $1 + x^5 + x^8 + x^9 + x^{10}$. Using (14), we compute

$$d = 467523550650744744856027134573563377101619107673 27,$$

which has

$$P = 792412797713126686196656160294175215426473063853$$

as a prime factor of size 160 bits.

Then, we compute a root of $\gcd(x^{11}-3, 2^{15}-1-x^5-x^8-x^9-x^{10})$ modulo P, and we obtain

$$\gamma = 4747967364968016271490925886337737240519368414 06.$$

We have investigated different set of parameters and applied the same technique to define suitable prime moduli. In table 2, we give the number of such primes and the corresponding parameters.

Table 2. Number of primes P greater than 2^{160} for use in elliptic curve cryptography, and the corresponding AMNS parameters.

$k+1$	n	c	ξ_i	Number of primes of size ≥ 160 bits
16	11	$\{2,3\}$	$\{0,1\}$	132
16	11	$\{2,3,4,5,6\}$	$\{0,1\}$	306
16	11	2	$\{0,1,2\}$	3106
16	11	$\{2,3,4,5,6\}$	$\{0,1,2\}$	≥ 7416 (a)
32	6	$\{2,3,4,5,6\}$	$\{0,1\}$	≥ 12 (b)
32	6	$\{2,3,4,5,6\}$	$\{0,1,2\}$	≥ 87 (b)
64	16	$\{2,3,4,5,6\}$	$\{0,1\}$	≥ 1053 (b)

(a): the determinant d was already a prime in 7416 cases. We did not try to factorize it in the other cases.
(b): computation interrupted.

6 Others operations in an AMNS

In this section we briefly describe the other basic operations in AMNS in order to provide a fully functional system for cryptographic applications. We present methods for converting numbers between binary and AMNS, as weel as solutions for addition and subtraction. It the context of elliptic curve cryptography, it is important to notice that, except for the inversion which can be performed only once at the very end of the computations if we use projective coordinates, all the operations can be computed within the AMNS. Thus conversions are only required at the beginning and at the end of the process.

6.1 Conversion from binary to AMNS

Theorem 2. *If X is an integer such that $0 \leq X < P$, given in classical binary representation, then a representation of X in the AMNS $\mathcal{B}$ is obtained with at most $2(n-1) + \frac{6(n-1)}{k-2}$ calls to Red.*

Proof. We simply remark that $P < 2^{n(k+1)}$ and that the size of the largest coefficient of X is reduced by $\lceil \frac{k}{2} \rceil - 1$ bits after each call to Red. Thus the reduction of $0 \leq X < P$ requires at least $\frac{(n-1)(k+1)}{\lceil \frac{k}{2} \rceil - 1}$ iterations, or more precisely $2(n-1) + \frac{6(n-1)}{k-2}$ is k is even, and $2(n-1) + \frac{4(n-1)}{k-1}$ if k is odd. □

We use theorem 2 by applying the coefficient reduction CR (algorithm 3) to the vector $(X, 0, \ldots, 0)$.

6.2 Conversion from AMNS to binary

Given $X = (x_0, \ldots, x_{n-1})_{\mathcal{B}}$, we have to evaluate $X = \sum_{i=0}^{n-1} x_i \gamma^i \bmod P$. The binary representation of X can be obtained with Horner's scheme

$$X = x_0 + \gamma\,(x_1 + \gamma\,(x_2 + \cdots + \gamma\,(x_{n-2} + \gamma\, x_{n-1}) \cdots)) \bmod P.$$

Since γ is of the same order of magnitude as P, the successive modular multiplications must be evaluated with Barrett or Montgomery algorithms. The cost of the conversion is thus at most $3n^3\, T_m$.

6.3 Addition, subtraction

Given $X = (x_0, \ldots, x_{n-1})_{\mathcal{B}}$ and $Y = (y_0, \ldots, y_{n-1})_{\mathcal{B}}$, the addition is simply given by $S = (x_0 + y_0, \ldots, x_{n-1} + y_{n-1})_{\mathcal{B}^+}$, where $\mathcal{B}^+$ denotes an extension of the AMNS $\mathcal{B}$ where the elements are not necessarily less than ρ. Since the input vectors of our modular multiplication algorithm do not need to have their elements less than ρ, this is a valid representation. However, if the reduction to $\mathcal{B}$ is required, it can be done thanks to algorithm CR.

Subtraction $X - Y$ is performed by adding X and the negative of Y. We use $Z = (z_0, \ldots, z_{n-1})_{\mathcal{B}+}$ as a representation of 0 in $\mathcal{B}^+$, i.e. with $z_i > \rho$ for $i = 0 \ldots n-1$. From (12) and (13) we have

$$Z = \sum_{i=0}^{n-1} z_i \gamma^i \equiv 0 \bmod P,$$

with $z_i = 3\left[2^k - \left(\sum_{j=0}^{i} \xi_j + c \sum_{j=i+1}^{n-1} \xi_j\right)\right]$.

The negative of Y is thus given in $\mathcal{B}^+$ by the vector $(z_0 - y_0, \ldots, z_{n-1} - y_{n-1})_{\mathcal{B}+}$. For the reduction in $\mathcal{B}$, the same remark as for the addition applies.

7 Conclusions

In this paper we defined a new family of moduli suitable for cryptography. In that sense, this research can be seen as an extension of the works by J. Solinas, and J. Chung and A. Hasan. We introduced a new system of representation for the integers modulo P, called Adapted Modular Number System (AMNS), and we proposed a modular multiplication in AMNS which is more efficient than Montgomery. We explained how to construct an AMNS which lead to good moduli, and we explicitly provided examples of primes for cryptographic sizes. Surprisingly, none of the primes we obtained with our computations belong to the large Mersenne family.

References

1. P. Barrett. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In A. M. Odlyzko, editor, *Advances in Cryptology - Crypto '86*, volume 263 of *LNCS*, pages 311–326. Springer-Verlag, 1986.
2. J. Chung and A. Hasan. More generalized mersenne numbers. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – SAC 2003*, volume 3006 of *LNCS*, Ottawa, Canada, August 2003. Springer-Verlag. (to appear).
3. R. Crandall. Method and apparatus for public key exchange in a cryptographic system. U.S. Patent number 5159632, 1992.
4. D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag, 2004.
5. Ç. K. Koç, T. Acar, and B. S. Kaliski Jr. Analyzing and comparing montgomery multiplication algorithms. *IEEE Micro*, 16(3):26–33, June 1996.
6. A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.
7. P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, April 1985.
8. J. Solinas. Generalized mersenne numbers. Research Report CORR-99-39, Center for Applied Cryptographic Research, University of Waterloo, Waterloo, ON, Canada, 1999.