Residue systems efficiency for modular products summation: Application to Elliptic Curves Cryptography

Jean-Claude Bajard, Sylvain Duquesne, Milos Ercegovac, Nicolas Meloni

Residue systems efficiency for modular products summation: Application to Elliptic Curves Cryptography

JC Bajard[a], S. Duquesne[b], M Ercegovac[c] and N Meloni[ab]

[a] ARITH-LIRMM, CNRS Université Montpellier2, France;
[b] I3M, CNRS Université Montpellier2, France;
[c]UCLA, Computer Science Dep.Los Angeles, US

ABSTRACT

Residue systems of representation, like Residue Number Systems (RNS) for primary field($GF(p)$) or Trinomial Residue Arithmetic for binary field ($GF(2^k)$), are characterized by efficient multiplication and costly modular reduction. On the other hand, conventional representations allow in some cases very efficient reductions but require costly multiplications.

The main purpose of this paper is to analyze the complexity of those two different approaches in the summations of products. As a matter of fact, the complexities of the reduction in residue systems and of the multiplication in classical representations are similar. One of the main features of this reduction is that it doesn't depend on the field. Moreover, the cost of multiplication in residue systems is equivalent to the cost of reduction in classical representations for special well-chosen fields.

Taking those properties into account, we remark that an expression like $A * B + C * D$, which requires two products, one addition and one reduction, evaluates faster in a residue system than in a classical one. So we propose to study types of expressions to offer a guide for choosing a most appropriate representation.

One of the best domain of application is the Elliptic Curves Cryptography where addition and doubling points formulas are composed of products summation. The different kinds of coordinates like affine, projective, and Jacobean, offer a good choice of expressions for our study.

Keywords: Elliptic Curve Cryptography (ECC), modular addition, modular multiplication, modular reduction, Residue Number System (RNS), hardware implementation

1. MULTIPLICATION AND REDUCTION PROBLEMS

In the expressions using modular arithmetic in domains like cryptography we find combinations of multiplications and additions. The modular reductions are not necessarily linked to the operations, i.e., we can perform several operations before modular reduction. This is particularly true for a series of additions, where the growth of the numbers involved is slow.

In multiplications, however, the growth of precision is fast since the number of digits in the product is the sum of those of the operands. Most of the time multiplications are followed by additions, allowing minimizing the number of reductions. This is true for any number representations, but it is clear that the faster the operations are, the better the expected gain would be, even if the reduction is costly.

1.1. Cost of Multiplication

Since we are considering Elliptic Curve Cryptography (ECC), we assume that the operands don't need more than 512 bits. Hence, we analyze only one multiplication algorithm: the Basic (paper-and-pencil) Method. This is based on the GMP recommendations[12]. Toom-Cook method[14] is only applicable for huge numbers not used in the ECC, and Karatsuba method[14] is more appropriate for software implementation (like in GMP) than for dedicated architectures we are considering.

Architecture	word size	Karatsuba	Toom-Cook
AMD K7	32	26 (832)	202 (6464)
Pentium 4	32	18 (576)	139 (4448)
PowerPC 32	32	20 (640)	226 (7232)
SPARC v7 32	32	8 (256)	466 (14912)
PowerPC 64	64	8 (512)	57 (3648)
ultrasparc-II	64	22 (1408)	98 (6272)
IA64	64	47 (3008)	288 (18432)

Table 1. Thresholds of method usefulness given in number of words (bits)

To summarize, depending on the architecture of the processors, the Basic Method is preferable up to an average of 16 words (512 bits, for a 32 bits architecture), see Table 1.

In this paper, we consider a k-bit word architecture, with $k = 16, 32$ or 64. Let $\beta = 2^k$. Then, a large integer is represented with radix β as $A = \sum_{i=0}^{n-1} a_i \beta^i$, having a length of kn bits.

1.1.1. The Basic Method

The Basic Method is best performed with the Horner scheme :

Algorithm 1: Basic(A, B)

Data: $A = \sum_{i=0}^{n-1} a_i \beta^i$ and $B = \sum_{i=0}^{n-1} b_i \beta^i$;
Result: P such that $P = A \times B$;
$P \leftarrow 0$;
for *i=n-1* **to** *0* **do**
1 $\quad (\underline{P}, \overline{P}) \leftarrow a_i \times B$;
2 $\quad P \leftarrow (P + \overline{P})\beta + \underline{P}$
end
return P

Line 1, the digit-by-digit product $a_i \times b_j$ consists two k-bit words, the upper and the lower one. The product $a_i \times B$ result in upper and lower parts, stored in two n-words numbers $\overline{P}$ and $\underline{P}$ which are added to the partial result P. In this form we have a better idea of the number of additions needed.

The cost of the Basic Method is n^2 products of two k-bits words giving a result on two words, and $2n^2 - n$ additions (with a carry) of two k-bits words. We don't take into account the carry propagation.

1.1.2. About parallel implementation

In fact, in the literature, only parallel implementations of the Basic Method are proposed. In this algorithm when we have at our disposal n^2 multipliers, all the products (n times line 1 of algorithm 1) can be done in parallel. Then with a tree of adders, all the additions can be computed in a logarithmic time. With some tricks the area can be reduced in $O(n^2/\log^2(n))$[8].

In a configuration with n arithmetic units, we can perform in parallel the n products of each call to line 1 of the Basic algorithm. Then, to avoid the carry propagation, we propose to accumulate the carries of the previous in a n-words variable R initialized to 0 (a kind of carry save approach). Thus the line 2 of Algorithm 1, becomes:

$$(P, R) \leftarrow (P + \overline{P})\beta + \underline{P} \tag{1}$$

Further author information: JC Bajard: E-mail: bajard@lirmm.fr.

At the end a carry-propagate addition produces $P \leftarrow P + R$.

The time complexity of this kind of implementation is n word-products and $3n$ word-additions (taking into account the accumulation of carries) and a final carry-propagate addition of two numbers of $2n$ words.

1.2. Modular reduction algorithms

1.2.1. Pseudo-Mersenne moduli

The NIST recommendations propose to use moduli which satisfy the Pseudo-Mersenne property:

$$N = \beta^n - c, \text{ with } c < \beta^{\frac{n}{2}}, \text{ and } w_H(c) < t$$

where w_H represents the Hamming weight and t is a small number. We remark that c can be written using sign digits. In this case w_H represents the non-zero digits.

The reduction can be done with three additions and two (half) products by c. If c is composed of $w_H(c)$ non-zero digits then those operations can be done by $w_H(c)$ additions.

If $X < \beta^{2n}$ then

$$X = X_1\beta^n + X_0 \equiv c \times X_1 + X_0 = X' \tag{2}$$

Now $X' < \beta^{\frac{3}{2}n}$, we reiterate the previous process

$$X' = X'_1\beta^n + X'_0 \equiv c \times X'_1 + X'_0 = X" \tag{3}$$

.

Thus $X" < 2\beta^n$, and a last reduction could be useful. In this case we just have to consider the bit r of weight β^n:

If $r = 1$ then $X \pmod N = (X" + c) \bmod \beta^n$
else $\widehat{X} = (X" + c) \bmod \beta^n$ and $r' = (X" + c) \text{ div } \beta^n$;
if $r' = 1$ then $X \pmod N = \widehat{X}$ else $X \pmod N = X"$.

To summarize: the cost of this reduction is two multiplications of $\frac{n}{2} \times n$ words integers and three n-words additions , or if t is small, lower than $3+2t$ n-words additions. Many approaches around this class of numbers have been considered such as in[9, 21].

1.2.2. Montgomery approach

This reduction is available when we use the Montgomery representation: An integer X is represented by the value $\widetilde{X} = X \times \beta^n \bmod N$.

Hence when a reduction occurs after a multiplication we get $\widetilde{X} \times \widetilde{Y} \equiv XY \times \beta^{2n} \times \beta^{-n} \bmod N \equiv XY \times \beta^n \pmod N$, which is a Montgomery representation of the product XY.

Algorithm 2: Montgomery$_N(R)$

Data: $R = \widetilde{X} \times \widetilde{Y} < N^2 < \beta^{2n}$ and $\beta^{n-1} \leq N < \beta^n$
and a precomputed value $(-N^{-1} \bmod \beta^n)$;
Result: $R' = R\beta^{-n} \pmod N < 2N$;
$Q \leftarrow R \times (-N^{-1}) \bmod \beta^n$;
$R' \leftarrow (R + QN)/\beta^n$;

We can also consider a digit version of this algorithm:

Algorithm 3: MontgomeryD$_N(R)$

Data: $R = \widetilde{X} \times \widetilde{Y} < N^2 < \beta^{2n}$ and $\beta^{n-1} \leq N < \beta^n$
and a precomputed value $(-n_0^{-1} \bmod \beta)$;
Result: $R' = R\beta^{-n} \pmod N < 2N$;
$R' \leftarrow R$;
for *i = 0* **to** *n-1* **do**
1 $\quad q \leftarrow r'_0 \times n_0^{-1} \bmod \beta$;
2 $\quad R' \leftarrow (R' + qN)/\beta$;
end

In line 1, we perform a multiplication of two words where only the lower word of the result is considered. Then in line 2, we multiply a n words value by a word, as described before, the result can be stored in two n words values, one for the lower parts and one for the upper one. That means that we have to perform n words products. Then, those results have to be added to R, that represents $2(n-1)$ words additions. Here we don't take into account the carry propagation. Hence, the total cost is $n^2 + n$ words products and $2n(n-1)$ words additions.

Now, for a parallel implementation on n arithmetic units we can store the carries in a variable C, we only have to evaluate exactly r_0 using c_0 (C is shifted like R). Thus, line 2 is reduced to one step of n words products and two steps of $(n-1)$ words additions. At the end, C must be added to R.

Montgomery algorithm is actually the most often implemented method available for any modulo[7, 15, 19]. It is particularly interesting for general architecture not dedicated to only one modulo, unlike the pseudo-Mersenne ones. This is the reason why it is so popular.

2. RNS PROPERTIES

2.1. A short introduction

The Residue Number System (RNS)[11, 24] is discussed in detail in[22] and[14]. The RNS is based on the Chinese remainder theorem which allows to represent an integer $X < M$ by the set $(x_1, \ldots, x_n)$ where $M = \prod_{i=1}^{n} m_i$ with $(m_i, m_j) = 1$ for $i \neq j$, and the residues $x_i = X \bmod m_i$.

The use of RNS is well-known in signal processing[13, 23] and cryptography[4, 10]. The main advantage of those systems is due to the fact that additions and multiplications are done independently on the residues. In other words, with a parallel architecture with n arithmetic units, the time needed to perform an addition or a multiplication is bounded by one modular operation on the largest residue. Therefore the choice of the set $(m_1, \ldots, m_n)$, called RNS basis, is very important, as we have to reduce modulo m_i. In[6], the authors propose some criteria for selecting a "good" RNS bases.

2.2. Choice of Moduli

To have better performance, the moduli must be pseudo-Mersenne numbers, namely of the form $2^k - c$ with c less than $2^{k/2}$ and it is preferable that the maximal weight of c in signed digit representation (denoted by a) is as small as possible. In this section, we are interested in finding sufficiently many such integers which are pairwise coprime.

To take into account the improvement of the change of basis presented in[6] , one wants to have consecutive moduli. We will denote by δ the amplitude of the set of moduli constructed. In general, such a set is split in order to construct two basis. Then, the interesting value is $\delta_{\max}$ which is the maximum of the magnitudes of the two basis.

We will deal with two sizes of numbers. On one hand we want numbers less than 2^{16} to use RNS arithmetic on 16-bit architectures and, on the other hand, we want numbers less than 2^{32} to use RNS arithmetic on 32-bit architectures.

2.2.1. Case of 16 bits words

It is of course not easy to find a set of such pairwise coprime numbers. If $a = 3$, it is not possible to find a set with more than 21 elements. We give here only the value of c.

{8, 15, 17, 27, 29, 39, 47, 57, 59, 63, 95, 113, 119, 123, 125, 127, 129, 131, 135, 137, 143}.

If $a = 4$ one can find a set of 40 such numbers (and no more).

{15, 17, 19, 21, 23, 27, 29, 33, 35, 39, 45, 47, 53, 57, 59, 63, 77, 83, 87, 89, 99, 105, 113, 117, 123, 125, 129, 143, 147, 153, 155, 165, 167, 183, 189, 192, 195, 197, 209, 225 }.

Without any condition on a, the largest possible set has 48 elements. Since in a n-words RNS arithmetic it is necessary to have a set with $2n$ elements, it is not possible to perform RNS arithmetic with more than 24 16-bits words.

{15, 17, 19, 21, 23, 27, 29, 33, 35, 38, 39, 45, 47, 53, 57, 59, 63, 77, 83, 87, 89, 99, 105, 113, 117, 123, 125, 129, 143, 147, 153, 155, 165, 167, 173, 179, 183, 189, 195, 197, 209, 213, 215, 225, 227, 243, 249, 255}.

Finally, the difference between two numbers of the same set is always less than 2^8 by construction so that the improvements on the RNS change of basis given in[6] can be done. Moreover, it is possible to split the sets above in two parts such that the difference between two numbers in each subset is even smaller. For instance, if one wants to perform ECC 160, one can choose {8, 15, 17, 27, 29, 39, 47, 57, 59, 63 } and {113, 119, 123, 125, 127, 129, 131, 135, 137, 143} so that $\delta_{\max} < 2^6$.

2.2.2. Case of 32 bits words

There are more choices in this case. If $a = 3$ the maximal size of a suitable set is 87 and if $a = 4$ one can find 448 numbers which are pairwise coprime. This is more than necessary to deal with very large numbers. We do not give the sets here since they are easy to find, using for instance the program available at `http://www.math.univ-montp2.fr/~duquesne`.

However the improvement given in[6] cannot be used since we only have by construction that $\delta \leq 2^{16}$. We will now take into account the constraint that the difference between two numbers of the set must be less than δ.

A good value for δ is 2^8. If $a = 4$, one can find 47 such numbers

{3, 5, 9, 13, 15, 17, 21, 23, 27, 29, 33, 39, 45, 47, 57, 63, 65, 68 69, 75, 77, 83, 89, 93, 99, 105, 107, 117, 119, 129, 135, 143, 149, 153, 155, 159, 185, 189, 195, 197, 209, 225, 227, 233, 237, 245, 255 }.

This is more than required by elliptic curve cryptography since 32 numbers are sufficient. It is easy to extract from this set two bases of 16 moduli such that $\delta_{\max} < 2^7$, two bases of 8 moduli such that $\delta_{\max} < 2^5$ or two basis of 6 moduli such that $\delta_{\max} < 2^4$. Hence elliptic curve cryptography can be very efficiently implemented with this kind of moduli.

In order to obtain better performance, one can choose $a = 3$. But, in this case, one can find only 25 numbers which are pairwise coprime and satisfy the constraint on δ. It is not enough. If one wants to perform RNS arithmetic with large numbers one must relax the constraint. If we take $\delta = 2^9$ one can find 32 numbers with $a = 3$

{3, 5, 7, 9, 17, 21, 23, 27, 29, 33, 39, 47, 57, 59, 63, 65, 68, 69, 119, 129, 135, 143, 159, 225, 255, 383, 449, 479, 497, 507, 509, 513}.

Of course, it is not possible to have $\delta_{\max} < \delta$ for two bases of 16 moduli, but one can again obtain $\delta_{\max} < 2^5$ for two bases of 8 moduli and $\delta_{\max} \leq 2^4$ for two bases of 6 moduli. We summarize in the table 2 the possibilities for performing RNS arithmetic.

16 bits word	words	bits
$a = 3$	10	160
$a = 3$, $\delta_{\max} < 2^6$	10	160
$a = 4$	20	320
$a = 4$, $\delta_{\max} < 2^6$	12	192
$a = 4$, $\delta_{\max} < 2^7$	20	320
a arbitrary	24	384

32 bits words	words	bits
$a = 3$	43	1376
$a = 3$, $\delta_{\max} < 2^9$	16	512
$a = 3$, $\delta_{\max} < 2^8$	12	384
$a = 3$, $\delta_{\max} < 2^5$	8	256
$a = 3$, $\delta_{\max} < 2^4$	6	192
$a = 4$	224	7168
$a = 4$, $\delta_{\max} < 2^8$	23	736
$a = 4$, $\delta_{\max} < 2^7$	16	512
$a = 4$, $\delta_{\max} < 2^5$	8	256
$a = 4$, $\delta_{\max} < 2^4$	6	192

Table 2. number of words for which RNS is possible

2.3. The RNS Montgomery Reduction

An efficient RNS modular reduction was proposed in[2]. This approach is a translation in RNS of the algorithm 2, where operations are performed in RNS.

Algorithm 4: RNS_Montgomery$_N(R, \widetilde{R})$

Data: Two RNS bases $\mathcal{B}_n$, $\{m_1, \ldots, m_n\}$, where $M = \prod_{i=1}^{n} m_i$;
and $\widetilde{\mathcal{B}}_{\tilde{n}}$, $\{\widetilde{m}_1, \ldots, \widetilde{m}_{\tilde{n}}\}$, where $\widetilde{M} = \prod_{i=1}^{\tilde{n}} \widetilde{m}_i$ where $\gcd(M, \widetilde{M}) = 1$ and $M < \widetilde{M}$;
N known in the two bases, with $\gcd(N, M) = 1$, and $0 < (n+2)N < M$
R given in RNS in the two RNS bases with $R < M * N$, we note $\widetilde{R}$
Result: $R < (n+2)N$ expressed in the two RNS bases with $R \equiv R \times M^{-1} (\mathrm{mod} N)$

1 $Q \leftarrow R \times_{RNS} (-N)^{-1}$ in $\mathcal{B}_n$;
2 Extension 1: $Q \rightarrow \widehat{Q}$ from $\mathcal{B}_n$ to $\widetilde{\mathcal{B}}_{\tilde{n}}$;
3 $\widetilde{R} \leftarrow (\widetilde{R} \ +_{RNS} \widehat{Q} \times_{RNS} N) \times_{RNS} M^{-1}$ in $\widetilde{\mathcal{B}}_{\tilde{n}}$;
4 Extension 2: $\widetilde{R} \rightarrow R$ from $\widetilde{\mathcal{B}}_{\tilde{n}}$ to $\mathcal{B}_n$

The drawback of this algorithm is due to the inversion of M, which is not possible in its basis, so a bases extension is needed. We analyze those extensions in the next section.

2.4. Bases extensions

The first extension takes the Lagrange formulation:

$$\widehat{Q} = \sum_{i=1}^{n} \left| q_i \left| M_i \right|_{m_i}^{-1} \right|_{m_i} M_i = Q + \alpha M \tag{4}$$

for some value of α where $0 \leq \alpha < n$. Thus, this computing is done for each modulo $\widetilde{m}_j$ of the basis $\widetilde{\mathcal{B}}_{\tilde{n}}$. Hence, the first extension does not really give the representation of Q in $\widetilde{\mathcal{B}}_{\tilde{n}}$, but the one of $\widehat{Q}$. That gives the condition $0 < (n+2)N < M$.

Now for the second extension, we use the same approach with $\widehat{R}$, which is such that: $\widehat{R} \equiv R \pmod{N}$.

$$\widehat{R} = \sum_{i=1}^{n} \left| \tilde{r}_i \left| \widetilde{M}_i \right|_{\widetilde{m}_i}^{-1} \right|_{\widetilde{m}_i} \widetilde{M}_i = \widetilde{R} + \alpha \widetilde{M} \tag{5}$$

But, using an extra modulus $m_x > n$[20] (most of the time, m_x is a small power of 2), we can evaluate α[20]:

$$\alpha = \left| |\widetilde{M}|_{m_x}^{-1} \left(\sum_{i=1}^{n} \left| \left| \tilde{r}_i \left| \widetilde{M}_i \right|_{\widetilde{m}_i}^{-1} \right|_{m_i} \widetilde{M}_i \right|_{m_x} - \left| \widetilde{R} \right|_{m_x} \right) \right|_{m_x} \tag{6}$$

Thus it is now possible to compute $r_j = |R|_{m_j}$ by

$$r_j = \left| \sum_{i=1}^{n} \left| \tilde{r}_i \left| \widetilde{M_i} \right|_{\widetilde{m}_i}^{-1} \right|_{\widetilde{m}_i} \widetilde{M_i} \right|_{m_j} - \left| \alpha \widetilde{M} \right|_{m_j} \Bigg|_{m_j} \tag{7}$$

2.5. Complexity

2.5.1. First extension

This extension is made using formula (4) for each $\widetilde{m}_j$ and for m_x. So, that needs n products mod m_i for the $\left| q_i \left| M_i \right|_{m_i}^{-1} \right|_{m_i}$, and $n^2 + n$ products by the M_i modulo $\widetilde{m}_j$, and $n^2 - 1$ additions.

If we consider an architecture with n (+1 but for small value m_x), that is done in a time equivalent to $n + 1$ modular products and n additions.

Now if we consider that a modular product costs 1 product and a additions, the complexity is $n^2 + 2n$ words products and $n^2(a + 1) + na - 1$ words additions. In parallel, the time complexity is equivalent to $n + 1$ words products and $n(a + 1) + a$ words additions.

2.5.2. Second extension

Now we use the formulas (4) and (6) which give the same complexity as for the first extension, plus one addition and one modular product for the evaluation of α and n modular products and n additions for the last reduction in equation (7).

Thus we obtain $n^2 + 3n$ words products and $n^2(a + 2) - 1$ words additions. In parallel implementation, the time complexity is equivalent to $n + 1$ words products and $n(a + 1) + a$ words additions.

2.5.3. Total cost of the RNS reduction

The complexity of lines 1 and 2 is due to RNS operations, thus we must perform $3n$ words modular products and n words modular additions. Since the cost of the reduction is a words additions, we get $3n$ words products and $(3a + 1)n$ words additions.

Hence, the total cost is $2n^2 + 8n + 1$ words modular products and $2n^2 + 2n - 1$ words modular additions. Thus the complexity is $2n^2 + 8n + 1$ words products and $2n^2 + 2n - 1 + a(2n^2 + 6n)$ words additions (m_x is a power of 2 and the reductions of the additions are included in those of the products).

3. STUDY OF DIFFERENT VARIETIES OF FORMULA

Here we propose to compare our approach in terms of multiplications which are more costly than additions. Furthermore, the a additions of the modular product reduction can be easily included in the product generation in dedicated architecture (i.e., on FPGAs[1]). We summarize in Tables 3 and 4 the different complexity found for the global cost of the methods and the time complexity for a parallel implementation with n arithmetic units.

Method	words-products	words-additions
Basic	n^2	$2n^2 - n$
Pseudo Mersenne	2B	$3n$
Pseudo Mersenne	0	$(3+2t)n$
Montgomery	$n^2 + n$	$2n(n-1)$
RNS multiplication	$2n$	$2an$
RNS Montgomery	$2n^2 + 8n + 1$	$2n^2 + 2n - 1 + a(2n^2 + 6n)$

Table 3. Number of words operations

Method	words-products	words-additions
Basic	n	$3n + (2n \text{ or } \log(n))$
Pseudo Mersenne	2B	$3(n \text{ or } \log(n))$
Pseudo Mersenne		$(3+2t)n$
Montgomery	$2n$	$2n + 1$
RNS multiplication	2	$2a$
RNS Montgomery	$2n + 7$	$2n + 1 + a(2n + 6)$

Table 4. Parallel implementation : Number of words operations

3.1. Formulas of the type $\sum_{i=1}^{s} A_i B_i \bmod N$

Whatever the representation of the values involved in the calculation of a such an expression, it is clear that the best is to compute reductions mod$N$ only when necessary. In this case, we consider that only one reduction is necessary.

Indeed, that modifies somewhat the conditions depending on the reduction method. For example, with the Montgomery method for summing $\log_2 s$ extra rounds could be necessary, or with the pseudo-Mersenne c must be smaller than $\beta^{n/2-\log_2 s}$.

Thus, the complexity of the evaluation of the kind of formulas is equivalent to s products and $s-1$ additions following by only one reduction. In Table 5 we summarize the cost as a function of s of the different combinations which could be used to compute our expression: Basic + Pseudo Mersenne (B+PM), Basic + Montgomery (B+Montg), and RNS product + RNS Montgomery (RNS) .

Combination	# products	parallel cost over n units
B+PM	$(s) \times n^2$	$s \times n$
B+Montg	$(s+1) \times n^2 + n$	$(s+2) \times n$
RNS	$(2s+8) \times n + 2n^2 + 1$	$2s + 2n + 8$

Table 5. Number of products in the evaluation of $\sum_{i=1}^{s} A_i B_i \bmod N$

If we consider the global cost of these different approaches, we could evaluate for which value of s RNS becomes more efficient. First we deal with B+PM, in which case we have:

$$(2s+8) \times n + 2n^2 + 1 < s \times n^2 \quad (8)$$

$$s > \frac{2n^2 + 8n + 1}{n^2 - 2n} \quad (9)$$

Thus, for $s = 3$ the RNS is better as long as $15 \leq n$, and $s = 4$is required if $9 \leq n \leq 14$. Now, we could

compare to B+Montg with the RNS approach:

$$(2s+8)\times n + 2n^2 + 1 < (s+1)\times n^2 + n \quad (10)$$

$$s > \frac{n^2+7n+1}{n^2-2n} \quad (11)$$

Here, RNS is most efficient for $s \geq 2$ and $n \geq 12$, $s \geq 3$ and $7 \leq n \leq 11$ or $s \geq 4$ with $n = 6$.

The RNS is remarkable when we consider a parallel implementation on n arithmetic units. Compared to B+PM, RNS begin to be efficient when $s \geq 3$ for $n \geq 13$ (the condition is $s \geq \frac{2n+7}{n-2}$). But if we deal with general architectures available for many fields, B+Montg is preferable. In this case RNS is better when $s > 0$ for $n \geq 9$ (the condition is $s \geq \frac{7}{n-2}$).

3.2. Application to ECC

Let p be a prime number and $E : y^2 = x^3+ax+b$ be an elliptic curve defined over $\mathbb{F}_p$. Let also $P = (X_p, Y_p, Z_p)$ and $Q = (X_q, Y_q, Z_q) \in E(\mathbb{F}_p)$ given in projective coordinates. Assume that the difference $P - Q = (x, y)$ is known in affine coordinates. Then one can obtain the X and Z-coordinates for $P + Q$ and $2P$ in terms of the X and Z-coordinates for P and Q by the following formulas :

$$\begin{aligned}
X_{p+q} &= -4bZ_pZ_q(X_pZ_q + X_qZ_p) + (X_pX_q - aZ_pZ_q)^2, \\
Z_{p+q} &= x\left((X_pZ_q + X_qZ_p)^2 - 4X_pX_qZ_pZ_q\right), \\
X_{2p} &= \left(X_p^2 - aZ_p^2\right)^2 - 8bX_pZ_p^3, \\
Z_{2p} &= 4X_pZ_p\left(X_p^2 + aZ_p^2\right) + 4bZ_p^4.
\end{aligned}$$

In[3] we proposed to compute X_{p+q} and Z_{p+q} using the following operations:

$$1.\ \alpha = Z_pZ_q \qquad 2.\ \beta = X_pZ_q + X_qZ_p \qquad 3.\ \gamma = X_pX_q \qquad 4.\ \delta = -4b\alpha$$

$$5.\ X_{p+q} = \beta\delta + (\gamma - a\alpha)^2 \qquad 6.\ \epsilon = \beta^2 - 4\alpha\gamma \qquad 7.\ Z_{p+q} = x\epsilon$$

To compute X_{2p} and Z_{2p}, the following operations must be done :

$$1.\ \alpha = Z_p^2 \qquad 2.\ \beta = 2X_pZ_p \qquad 3.\ \gamma = X_p^2 \qquad 4.\ \delta = -4b\alpha$$

$$5.\ X_{2p} = \beta\delta + (\gamma - a\alpha)^2 \qquad 6.\ Z_{2p} = 2\beta(\gamma + a\alpha) - \alpha\delta$$

The scalar point multiplication over E is performed using the Montgomery ladder[17][16] which involves an addition and a doubling at each step of the algorithm. With the previous formulae, this operation can be done in 17 multiplications and 13 modular reductions.

So, one step of the Montgomery exponentiation algorithm using the previous formulae requires $18(2n) + 13(2n^2 + 8n)$ operations in RNS, $17n^2 + 14(n^2 + n)$ with the Montgomery modular multiplication and $17n^2$ with Mersenne numbers.

We can see that in practice our approach is slower than both methods for standard length (160-192). We make two remarks: first, compared to the Montgomery modular multiplication, we are asymptotically better; second, the problem comes from the fact that those formulae remain simple and involve only 4 independent coordinates, which limits the possibility of obtaining long sums of products. This suggests that our approach may be more suited to hyperelliptic curves (which can involve 12 different parameters).

$\|p\|_2$	word	RNS	Montgomery	Mersenne
160	5	1350	845	425
192	6	1776	1200	612
256	8	2784	2096	1088
320	10	4000	3240	1700
512	16	8896	8160	4352

Table 6. Number of word size multiplications for one step of Montgomery exponentiation algorithm

4. CONCLUSIONS

In this study we have shown that the cost of the reduction in the RNS is compensated by fast evaluation of the product and the sum. Whenever more than two products have to be added, the RNS is a good alternative and if the number of terms of the sum increases, it becomes clearly the most efficient method.

One domain of application for this approach is the Elliptic Curve Cryptography where expressions of points addition formulae can be rewritten to take into account the RNS specificities[3]. These expressions can be improved and they could be extended to hyper-elliptic curves approaches.

Another point that could be analyzed is the number and the size of the RNS bases elements. It could be interesting to mix the basic approach with the RNS one. For example, we could consider m_i as pseudo-Mersenne numbers of 128 bits on which products and additions are made with the classical approach. In this cases the number of elements of the RNS bases should be small, for example 4, and we could perform the operations in the RNS and then convert into classical representation for the reduction. In fact, this could be used also for the modulo calculation for each m_i which will be part of our future research.

REFERENCES

1. Badrignans B., Mesquita D., et al: A Parallel and Secure Architecture for Asymmetric Cryptography ReCoSoC 2006, Montpellier France.
2. Bajard, J.C., Didier, L.S., Kornerup, P.: Modular multiplication and base extension in residue number systems. 15th IEEE Symposium on Computer Arithmetic, IEEE Computer Society Press (2001) 59–65
3. Bajard, J.C., Duquenne S., Meloni, N.: Combining Montgomery ladder for elliptic curves defined over $\mathbb{F}_p$ and RNS representation, Research Report LIRMM 06041 2006.
4. Bajard, J.C., Imbert, L.: A full RNS implementation of RSA. IEEE Transactions on Computers 53:6 (2004) 769–774
5. Bajard, J.C., Imbert, L., Liardet, P.Y., Teglia, Y.: Leak resistant arithmetic. CHES 2004, LNCS 3156 59–65
6. Bajard, J.C., Meloni, N., Plantard, T.: Efficient RNS bases for Cryptography IMACS'05, Applied Mathematics and Simulation, (2005) ???–???
7. Bosselaers, A., Govaerts, R., Vandewalle. J.: Comparison of the three modular reduction functions LNCS 773 (1994) 175–186
8. Bunimov, V., Schimmler, M.: Efficient Parallel Multiplication Algorithm for Large Integers Euro-Par 2003, International Conference on Parallel and Distributed Computing (2003) 923–928
9. Chung, J., Hasan, A.: More generalized mersenne numbers. SAC 2003, LNCS 3006 (2003) 335–347
10. Ciet, M., Neve, M., Peeters, E., Quisquater, J.J.: Parallel FPGA implementation of RSA with residue number systems– can side-channel threats be avoided? 46th IEEE International Midwest Symposium on Circuits and Systems (2003)
11. Garner, H.L.: The residue number system. IRE Transactions on Electronic Computers, EL 8:6 (1959) 140–147
12. The GNU Multiple Precision Arithmetic Library Edition 4.2.1, 2 May 2006 http://www.swox.com/gmp/gmp-man-4.2.1.pdf

13. Jullien G. A.: VLSI Digital Signal Processing: Some Arithmetic Issues Keynote at SPIE Conference on Advanced Signal Processing, August 1996 (167,566 bytes)
14. Knuth, D.: Seminumerical Algorithms. The Art of Computer Programming, vol. 2. Addison-Wesley (1981)
15. Montgomery, P.L.: Modular multiplication without trial division. Math. Comp. 44:170 (1985) 519–521
16. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comp. 48:177 (1987) 243–164
17. Okeya, O., Sakurai, K.: Efficient Elliptic Curve Cryptosystems from a Scalar Multiplication Algorithm with Recovery of the y-Coordinate on a Montgomery-Form Elliptic Curve. Cryptographic Hardware and Embedded Systems, LNCS 2162 (2001) 126–141
18. Shenoy, A.P., Kumaresan, R.: Fast base extension using a redundant modulus in RNS. IEEE Transactions on Computer 38:2 (1989) 292–296
19. Sanu, M.O., Swartzlander, E.E., Chase, C.M.: Parallel Montgomery Multipliers. 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'04) (2004) 63–72
20. Shenoy A. P., Kumaresan R.: Fast base extension using a redundant modulus in RNS. EEE Transactions on Computer, 38(2):292–296, 1989.
21. Solinas, J.: Generalized Mersenne numbers. Research Report CORR-99-39, Center for Applied Cryptographic Research, University of Waterloo (1999)
22. Szabo, N.S., Tanaka, R.I.: Residue Arithmetic and its Applications to Computer Technology. McGraw-Hill (1967)
23. Soderstrand M. A., Jenkins W. K. , Jullien G. A. , Taylor F. J.: Residue Number System Arithmetic: Modern Applications in Digital Signal Processing. IEEE Press, New York, 1986.
24. Svoboda, A., and Valach, M.: Rational system of residue classes. In Stroje na Zpraccorani Informaci, Sbornik, Nakl. CSZV, Prague, 1957, pp. 9-37