Combining leak–resistant arithmetic for elliptic curves defined over Fp and RNS representation

Jean-Claude Bajard, Sylvain Duquesne, Milos Ercegovac

▶ **To cite this version:**

Jean-Claude Bajard, Sylvain Duquesne, Milos Ercegovac. Combining leak–resistant arithmetic for elliptic curves defined over Fp and RNS representation. Publications Mathématiques de Besançon. Algèbre et Théorie des Nombres, 2013, pp.67-87. hal-01098795

HAL Id: hal-01098795

https://hal.sorbonne-universite.fr/hal-01098795v1

Submitted on 29 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

COMBINING LEAK–RESISTANT ARITHMETIC FOR ELLIPTIC CURVES DEFINED OVER $\mathbb{F}_p$ AND RNS REPRESENTATION

by

J. C. Bajard, S. Duquesne & M. Ercegovac

Abstract. — In this paper we give a survey of a method combining the residue number system (RNS) representation and the leak-resistant arithmetic on elliptic curves. These two techniques are relevant for implementation of elliptic curve cryptography on embedded devices.

It is well known that the RNS multiplication is very efficient whereas the reduction step is costly. Hence, we optimize formulae for basic operations arising in leak-resistant arithmetic on elliptic curves (unified addition, Montgomery ladder) in order to minimize the number of modular reductions. We also improve the complexity of the RNS modular reduction step. As a result, we show how to obtain a competitive secured implementation.

Finally, we show that, contrary to other approaches, this method takes optimally the advantage of a dedicated parallel architecture.

Résumé. — Dans cet article, nous donnons un survol d'une méthode qui combine le système de représentation des nombres basé sur les restes chinois (RNS) avec les formules de loi de groupe sur les courbes elliptiques qui sont résistantes aux attaques par fuites. Ces deux techniques sont particulièrement intéressantes pour l'implémentation de cryptosystèmes basés sur les courbes elliptiques dans des systèmes embarqués.

Dans le système de représentation RNS, la multiplication de grands entiers est très efficace contrairement à la réduction modulaire. Nous réécrivons donc ici les formules de la loi de groupe pour les modèles de courbes elliptiques résistant aux fuites (loi d'addition unifiée, forme de Montgomery) dans le but de minimiser le nombre de réductions modulaires. Nous améliorons aussi la complexité de l'étape de réduction. Ces deux aspects permettent d'obtenir des implémentations sécurisées compétitives.

Finalement, nous montrons que, contrairement aux autres approches, cette méthode peut profiter pleinement d'une architecture parallèle dédiée.

2010 ***Mathematics Subject Classification.*** — 14G50.

Key words and phrases. — Elliptic curves, Montgomery ladder, leak-resistance, residue number system (RNS), modular multiplication, modular reduction.

1. Introduction

Elliptic curve cryptosystems (ECC) have become popular to such a degree that they have been recommended by the NSA in 2005 for public key cryptography. However, as usual in cryptography, the algorithms involved must be protected against side-channel attacks which take advantage of leaks of information during computations ([**32, 33, 41**]). There are two types of such attacks, the simple power analysis (SPA) and the differential power analysis (DPA). In particular, elliptic curves are especially sensitive to SPA because of the difference in complexity between the addition of two points and the doubling of a point. Two ways have been used in the past 10 years to deal with this: the use of unified addition formulae ([**13, 29, 35**]) and the Montgomery ladder ([**13, 24, 28, 37**]). Unfortunately these protections consequently increase the algorithmic cost of the scalar multiplication (λ times a point of an elliptic curve, λ is an integer) so that it is very important to reduce the cost of the base field arithmetic. In this paper, as common in ECC, the base field is a random prime field $\mathbb{F}_p$ where the central operation is the modular multiplication. For standard representation of $\mathbb{F}_p$ the multiplication and the reduction step have similar cost ([**8, 36**]). On the contrary, the Residue Number System (RNS) introduces a gap of complexity between the multiplication and the reduction steps. Indeed, it allows a very efficient multiplication by distributing the computations on small independent integers whereas the reduction step remains costly. The RNS has other advantages. In particular, it is easily parallelizable and it is not specific to a value of p contrary to Mersenne number based arithmetic.
The aim of this paper is to combine SPA-resistant arithmetic on elliptic curve and the RNS. To obtain an interesting and efficient combination, we work on two aspects:

- The adaptation of the RNS reduction to the context of ECC (Section 3). It merges ideas from [**6**] and [**7**] and leads to intermediate RNS bases which are less restrictive than [**6**] and more efficient than [**7**]. We give a detailed complexity study of this algorithm and use it to provide comparisons between RNS and standard arithmetic in Section 5. This is made in a context as generic as possible (architecture for which additions and shifts are more efficient than products).
- The modifications of the formulae on elliptic curve for minimizing the number of modular reductions (Section 4), in exchange for an increased number of multiplications with respect to the RNS properties (i.e. costly reductions and cheap multiplications). In several cases, we propose new formulae adapted to these RNS features.

2. Background properties of the different representations and algorithms

2.1. Modular multiplication. — Let us assume that $p < \beta^n$ and that elements of $\mathbb{F}_p$ are given in radix representation (i.e., $X = \sum_{i=0}^{n-1} x_i \beta^i$ with $0 \leq x_i < \beta$). The elliptic curve arithmetic over $\mathbb{F}_p$ mainly involves modular multiplications modulo p. Because of the small size of the numbers used in ECC (192 to 512 bits, i.e., $n = 6$ to 16 32-bit words), the multiplication is performed by the schoolbook method with a complexity of n^2 word

operations. Indeed, for the current ECC key size, Karatsuba or Toom-Cook approaches remain costlier, as discussed in the study made by the GMP group [1].
The reduction step consists of finding the remainder of the Euclidean division by p. It can be substantially speeded up by using the Montgomery reduction or by using a special modulo.

2.1.1. Montgomery general reduction algorithm. — In **[36]**, Montgomery proposed to substitute the reduction modulo p by a division by a power of the radix β (i.e., a simple shift). The result is not exactly $X \mod p$ but $X\beta^{-n} \mod p$. This can be overcome by using Montgomery representation where $X' = X \times \beta^n \mod p$.

Algorithm 1: Montgomery$_p(R)$

Data: $R = A \times B < p\beta^n, p < \beta^n$, the precomputed value $-p^{-1} \bmod \beta^n$;
Result: r such that $r = \beta^{-n} \bmod p$, with $r < 2p$;
$q \leftarrow -R \times p^{-1} \bmod \beta^n$;
$r \leftarrow (R + qp)/\beta^n$;

The complexity of this reduction is $n^2 + n$ word operations **[10]**. As $r < 2p$, a comparison and a final subtraction could occur, but the output of Algorithm 1 can be used as input by adding a condition on p, specifically $4p < \beta^n$.
Finally, for $A < \beta^n$, its Montgomery representation is obtained via Algorithm 1 with $R = A \times (\beta^n \mod p)$. Similarly, if A' is the Montgomery representation of A, we recover A using Algorithm 1 with $R = A'$. However, since all the computations can be done in Montgomery representation, we ignore the cost of the conversion between Montgomery and classic representation.

2.1.2. Reduction using special modulo. — When p has some special form (as generalized Mersenne prime **[17, 44]**), the modular reduction can be performed with only some shifts and additions. This is used in most of the ECC standards but the main drawback is that a dedicated architecture is necessary which cannot be used for other prime fields. Consequently, it is not practical in either software or hardware implementation and many customers prefer flexible products. For this reason we do not consider this restrictive approach in this paper.

2.2. Leak-resistant arithmetic in elliptic curve cryptography. — As it is the dominant operation in all elliptic curve based schemes (such as encryption/decryption or signature generation/verification), the scalar multiplication of points on the curve must be very efficient. It is usually done by using standard scalar multiplication methods (double and add, sliding window) combined with recoding of the exponent. However, these methods are not leak-resistant which means that they are sensitive to side channel attacks like SPA or DPA. We are not interested in this paper in the protection against DPA. Indeed, in elliptic curve cryptography, this problem can be efficiently solved by randomizing the projective coordinates, the base point or the scalar **[20]**. In general these protections do not affect the arithmetic algorithms. On the other hand protections against SPA consequently increase the algorithmic cost of the scalar multiplication and, therefore, must be implemented as efficiently as possible.

1. Intel Pentium-4 gmp-mparam.h
#define MUL_KARATSUBA_THRESHOLD 23
#define MUL_TOOM3_THRESHOLD 137

Currently there are two means to perform SPA-resistant arithmetic on elliptic curves. The first one is to use a representation of the curve for which the addition and the doubling can be done with the same formulae. In this work, we are interested in unified formulae for three representations of the curve: the Hessian form, the Jacobi form and the short Weierstrass form, but this can easily be generalized to other representations (as Edwards curves) with similar results. The second one is to use the Montgomery ladder where both an addition and a doubling are performed at each step of the scalar multiplication. As for unified formulae, this method is more efficient on restrictive models of the curve and we are dealing both with the restrictive and the general model in this work.

2.2.1. Unified addition formulae. — An elliptic curve in Hessian form is given by an equation

$$X^3 + Y^3 + Z^3 = 3dXYZ$$

where $d \in \mathbb{F}_p$ is not a third root of unity. In **[29]**, Joye and Quisquater described formulae for the addition of two projective points (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2):

$$\left\{ \begin{array}{rcl} X_3 & = & Y_1^2 X_2 Z_2 - Y_2^2 X_1 Z_1, \\ Y_3 & = & X_1^2 Y_2 Z_2 - X_2^2 Y_1 Z_1, \\ Z_3 & = & Z_1^2 X_2 Y_2 - Z_2^2 X_1 Y_1. \end{array} \right. \tag{1}$$

These formulae require 12 field multiplications and can be used both for addition and doubling because $2(X, Y, Z) = (Z, X, Y) + (Y, Z, X)$.
At the same time, the use of the Jacobi model was introduced by Liardet and Smart in **[35]**. It is improved in **[9]** and, more recently, in **[21]**. It is easy to prove that any elliptic curve containing a 2-torsion point is birationally equivalent to a Jacobi quartic given by an equation

$$Y^2 = \varepsilon X^4 - 2\delta X^2 Z^2 + Z^4. \tag{2}$$

In this case, the formulae for adding two points are also valid for doubling:

$$\left\{ \begin{array}{rcl} X_3 & = & X_1 Z_1 Y_2 + Y_1 X_2 Z_2, \\ Y_3 & = & \left(Z_1^2 Z_2^2 + \varepsilon X_1^2 X_2^2\right)(Y_1 Y_2 - 2\delta X_1 X_2 Z_1 Z_2) + 2\varepsilon X_1 X_2 Z_1 Z_2 \left(X_1^2 Z_2^2 + Z_1^2 X_2^2\right), \\ Z_3 & = & Z_1^2 Z_2^2 - \varepsilon X_1^2 X_2^2. \end{array} \right.$$

In most cases, ε can be rescaled to a small value so that these formulae also require 12 multiplications **[21]**. However, these methods can only be used for curves having their cardinality even (Jacobi quartic) or divisible by 3 (Hessian form). To overcome this restriction, Brier and Joye give, in **[13]**, unified formulae for the short Weierstrass form

$$Y^2 Z = X^3 + aXZ^2 + bZ^3. \tag{3}$$

Again, formulae given for the addition are also valid for doubling:

$$\left\{ \begin{array}{rcl} X_3 & = & 2\lambda_d \left(\lambda_n^2 - (X_1 Z_2 + X_2 Z_1)(Y_1 Z_2 + Y_2 Z_1)\lambda_d\right), \\ Y_3 & = & \lambda_n \left(3(X_1 Z_2 + X_2 Z_1)(Y_1 Z_2 + Y_2 Z_1)\lambda_d - 2\lambda_n^2\right) - \left((Y_1 Z_2 + Y_2 Z_1)\lambda_d\right)^2, \\ Z_3 & = & 2\lambda_d^3, \end{array} \right.$$

where $\lambda_n = (X_1 Z_2 + X_2 Z_1)^2 - X_1 X_2 Z_1 Z_2 + a Z_1^2 Z_2^2$ and $\lambda_d = (Y_1 Z_2 + Y_2 Z_1) Z_1 Z_2$. However, these formulae are less efficient since they require 18 multiplications. Note that by using an isomorphism or an isogeny as in **[14]**, it is possible in most cases to rescale a to a small value and then to save one multiplication. This is explained and improved in Section 4.3 within Montgomery ladder context.

2.2.2. Montgomery scalar multiplication. — Montgomery proposed in **[37]** to work only with the x-coordinate. In this case, doubling is still possible but adding two points P and Q is possible only if $P-Q$ is known (this is called pseudo-addition).

Proposition 1. — *Let E be an elliptic curve defined over $\mathbb{F}_p$ in Montgomery form:*

$$BY^2Z = X^3 + AX^2Z + XZ^2.$$

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 $P-Q=(x,y)$ is known in affine coordinates. Then

$$\begin{aligned}
X_{P+Q} &= ((X_P - Z_P)(X_Q + Z_Q) + (X_P + Z_P)(X_Q - Z_Q))^2, \\
Z_{P+Q} &= x\left((X_P - Z_P)(X_Q + Z_Q) - (X_P + Z_P)(X_Q - Z_Q)\right)^2, \\
X_{2P} &= (X_P + Z_P)^2(X_P - Z_P)^2, \\
Z_{2P} &= 4X_PZ_P\left((X_P - Z_P)^2 + \tfrac{A+2}{4}4X_PZ_P\right), \\
&\qquad \text{with } 4X_PZ_P = \left((X_P + Z_P)^2 - (X_P - Z_P)^2\right).
\end{aligned}$$

In this way, both a pseudo-addition and a doubling requires only 3 multiplications and 2 squarings, which is much faster than usual operations (**[19]**). However, the pseudo-addition is not sufficient for usual scalar multiplication algorithms. Thus, Montgomery introduced a new algorithm for computing kG using pairs of consecutive multiples of G, so that the difference between the two components is always equal to G:

Algorithm 2: Montgomery_Scalar()

Data: $G \in E(\mathbb{F}_p)$ and $k \in \mathbb{N}^*$, $k=\sum_{i=0}^{t} k_i 2^i$, $k_i \in \{0,1\}$
Result: The X and Z-coordinate of kG

```
1 Initialize T = (O, G) where O is the point at infinity;
2 For i from [log_2 k] to 0 do;
3    If k_i = 0 then T = (2T[1], T[1] + T[2]) else T = (T[1] + T[2], 2T[2]);
4 return T[1];
```

Both an pseudo-addition and a doubling are done for each bit of the exponent which ensure natural SPA protection. The cost of this algorithm is about $10\,[\log_2 k]$ multiplications which is better than other algorithms (even those which are not SPA-resistant). Finally, the x-coordinate of kG is usually sufficient but some cryptosystems, like ECDSA, require the y-coordinate but it can easily be recovered as explained in **[38]**.
Unfortunately, in odd characteristic, all elliptic curves cannot be transformed into Montgomery form (which implies even cardinality). The Montgomery ladder can be generalized to curves in short Weierstrass form but is more time consuming (**[13]**,**[24]** and **[28]**).

Proposition 2. — *Let E be an elliptic curve defined over $\mathbb{F}_p$ by equation (3). 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*

$P - Q = (x, y)$ is known in affine coordinates. Then

$$\begin{aligned} X_{P+Q} &= -4bZ_PZ_Q(X_PZ_Q + X_QZ_P) + (X_PX_Q - aZ_PZ_Q)^2, \\ Z_{P+Q} &= x(X_PZ_Q - X_QZ_P)^2, \\ X_{2P} &= \left(X_P^2 - aZ_P^2\right)^2 - 8bX_PZ_P^3, \\ Z_{2P} &= 4Z_P\left(X_P^3 + aX_PZ_P^2 + bZ_P^3\right). \end{aligned}$$

Hence, pseudo-addition requires 10 multiplications and doubling only 9. So, the scalar multiplication can be performed in about $19\,[\log_2 k]$ multiplications in $\mathbb{F}_p$. Note that the y-coordinate can also be recovered in this case ([**13**]).
In this paper, we use Residue Number Systems for the arithmetic on the base field. As a consequence, the cost of the multiplication becomes negligible compared to the cost of the modular reduction. Thus, it is necessary to rewrite the formulae given above in order to minimize the number of modular reductions.

3. Residue Number Systems

After a short introduction of these systems of representation (Section 3.1), we explain how the RNS bases used in this paper are selected. This choice is based on results that can be found in [**7, 6**]. But here, the number of possible RNS bases is much larger than in [**6**] which is very restrictive, and the performances obtained are clearly better than in [**7**].
Then, we recall (Section 3.2) the translation in RNS of the well-known Montgomery algorithm for modular reduction [**36**] (see Section 2), as usually described in the literature [**1, 2, 30, 40**]. The main differences between these versions of this RNS Montgomery algorithm rely in the RNS bases conversions. We justify (Section 3.3) our choice of a conversion algorithm with respect to the bases selected in this paper. The complexity study (Section 3.4) supports our approach.

3.1. Representation. — The Residue Number Systems (RNS) are issued in a natural way from the Chinese Remainder Theorem (CRT). They are based on the fact that a number a can be represented by its residues $(a_1, a_2, \ldots, a_n)$ modulo a set of coprime numbers $(m_1, m_2, \ldots, m_n)$, called RNS basis, thus $a_i = a \bmod m_i = |a|_{m_i}$. We generally assume that $0 \leq a < M = \prod_{i=1}^{n} m_i$. The elements a_i are called RNS-digits, or simply digits if there is no ambiguity.
The strongest advantage of a such system is that the operations on large integers are performed independently on the residues. These systems were introduced and developed in [**45, 26, 46**]. A good introduction can be found in [**31**].
For constructing an arithmetic over $\mathbb{F}_p$, we assume that $M = \prod_{i=1}^{n} m_i$ is such that $p < M$. In this system, two numbers a and b can be represented by their remainders modulo the m_i, $i = 1, \ldots, n$.

$$a = (a_1, \ldots, a_n) \quad \text{and} \quad b = (b_1, \ldots, b_n)$$

A multiplication modulo M is reduced to n independent RNS-digit products.

$$r = (|a_1 \times b_1|_{m_1}, \ldots, |a_n \times b_n|_{m_n}) = a \times b \bmod M \tag{4}$$

A RNS-digit product is equivalent to a classical digit product followed by a modular reduction modulo m_i, which represents few additions (see **[7, 6]**). The digit modular reduction is frequent, so the choice of the RNS base appears to be fundamental in a such arithmetic. The bases considered in this paper are constructed for obtaining an efficient digit reduction. As in **[7]**, we consider RNS base $(m_1, ..., m_n)$ with elements such that, $m_i = 2^k - c_i$, where c_i is small, and as in **[6]** we consider $c_i = 2^{t_i} \pm 1$, with $c_i < 2^{k/2}$. For example, for $m_i < 2^{32}$, it is easy to find 16 coprime values with $c_i = 2^{t_i} \pm 1$, with $t_i = 0...16$ for $c_i = 2^{t_i} - 1$ and $t_i = 1...15$ if $c_i = 2^{t_i} + 1$. This ensures that $c_i < 2^{16}$. Then, if we want more co-prime values, we can consider the $c_i = 2^{t_i} \pm 2^{s_i} \pm 1$.
The reduction modulo m_i is, in this case, obtained with few shifts and adds. For $\rho < 2^{2k}$ (for example the result of a MAC operation with operands lower than 2^k), we consider that

$$\rho = \rho_h 2^k + \rho_l = c_i \rho_h + \rho_l \bmod m_i$$

where the product by c_i is just a shift and add processing. Hence, if we note $\rho' = c_i\rho_h + \rho_l = \rho'_h 2^k + \rho'_l$, we have $\rho' < 2^{3k/2}$ when $\rho < 2^{2k}$, thus we ensure that $\rho'' = c_i\rho'_h + \rho'_l \leq 2(2^k - 2^{k/2}) < 2m_i$. This property confirms that the reduction part on each m_i, in the case of a product (or MAC) operations, represents around 10% of the total cost **[7, 6, 11]**.Then in the following we consider that an RNS digit-product is equivalent to 1.1 word-product (word = k-bits).
We now focus on the multiplication modulo p using the Montgomery algorithm presented in **[1, 2]**.

3.2. RNS Montgomery reduction. — This algorithm is a direct transposition of the classical Montgomery method. The main difference is due to the representation system. When the Montgomery method is applied in a classical radix β number system, the value β^n occurs in the reduction, division and Montgomery factor. In RNS, this value is replaced by M. Thus an auxiliary RNS base is needed to handle the inverse of M. Hence some operations will be performed on the two bases, it is the case of the initial product which costs $2n$ RNS-digitproducts.
Algorithm 3 presents the RNS Montgomery reduction (c can be considered as the result of an RNS product on the two bases), where all the operations considered are in RNS. We specify on which basis they are done.

Algorithm 3: MontgR_RNS(c, p)

Data: Two RNS bases $\mathcal{B} = (m_1, \ldots, m_n)$, and $\mathcal{B}' = (m_{n+1}, \ldots, m_{2n})$, such that
$M = \prod_{i=1}^{n} m_i < M' = \prod_{i=1}^{n} m_{n+i}$ and $\gcd(M, M') = 1$;
p a prime number represented in RNS in both bases such that $0 < 4p < M$ and $\gcd(p, M) = 1$;
A positive integer c represented in RNS in both bases, with $c < Mp$.
Result: A positive integer $r = cM^{-1} \bmod p$ represented in RNS in both bases, with $r < 2p$.

1 $q \leftarrow (c) \times (-p^{-1})$ in $\mathcal{B}$;
2 $[q \text{ in } \mathcal{B}] \longrightarrow [q \text{ in } \mathcal{B}']$ *First base extension*;
3 $r \leftarrow (c + q \times p) \times M^{-1}$ in $\mathcal{B}'$;
4 $[r \text{ in } \mathcal{B}\,] \longleftarrow [r \text{ in } \mathcal{B}']$ *Second base extension*;

To summarize, q is computed modulo M, such that $(c + q \times p)$ is a multiple of M (thus its RNS representation in $\mathcal{B}$ is composed of zeros). Hence, we extend the representation of q in the base $\mathcal{B}'$ where M^{-1} exists and where $(c + qp) \times M^{-1}$ can be performed.
Instructions 1 and 3 of Algorithm 3 deal with RNS operations as presented in the previous section, which are performed independently for each element of the basis, so they are very efficient. These two instructions represent $3n$ RNS-digit products (which is equivalent to 3 RNS-digit operations on a n cells parallel architecture). Instructions 2 and 4 are dedicated to RNS base extensions, and remain quadratic (or linear on an n-cells architecture). We can use different full RNS extensions as shown in **[1, 2, 30, 40]**.
For two numbers a and b given in RNS, we compute $c = a \times b$ in RNS, then this algorithm actually evaluates $r = abM^{-1} \mod p$. To obtain the right result, we need to use it again with r and $(M^2 \mod p)$ as operands. To avoid this, we convert the values in a Montgomery representation where $a' = a \times M \mod p$ which is stable for Montgomery product and addition. This conversion is done once at the beginning by performing Montgomery product with a and $(M^2 \mod p)$ as operands, and once at the end of the complete cryptographic computing with 1 as second operand. Hence, this transformation will be neglected in the following. Moreover, as the RNS is not redundant, this representation is well suited for cryptography without any conversion **[4]**.

3.3. RNS Base extensions. — For the RNS bases extensions, there exist two main approaches coming from the conversion from RNS to standard number systems. One proceeds directly from the CRT (Chinese Remainders Theorem) and looks like a Lagrange interpolation $a = \left| \sum_{i=1}^{n} \left| a_i |M_i|_{m_i}^{-1} \right|_{m_i} M_i \right|_M$, with $M_i = M/m_i$ and $|.|_m$ the reduction modulo m. The other approach corresponds to a Newton interpolation using an intermediate representation calls Mixed Radix System (MRS), and is depicted later.
The drawback of the CRT version can be described in two points: the modulo M operation which means that we must subtract to the summation a multiple of M, and the randomness of the M_i which refrains from any improvement of the products by these values. To find the multiple of M, there are two main methods: one exact proposed in **[43]** using an extra modulo, or one by approximation **[40, 30]**. In **[2]** the authors propose to avoid the calculus of this multiple for the first extension (the one of q). But in all the cases, the evaluation of the summation stays necessary and requests n multiplications-accumulations. As this operation must be done modulo each element of the new base ($\mathcal{B}'$ or $\mathcal{B}$ depending of the extension), the real cost in RNS is, at least, n^2 RNS-digit products. Due to our choice of bases, looking in detail the calculus, we prove that our approach is 30% better.
We consider bases of the following form: $m_i = 2^k - 2^{t_i} \pm 1$ (or $2^k - 2^{t_i} \pm 2^{s_i} \pm 1$ if we need more elements, see Appendices) with $2^{t_i} \pm 1 < 2^{k/2}$ (for ECC, k can be equal to 32, for keys up to 1024 bits and maybe more). Due to properties of such bases, the best alternative for the base extension is, a priori, the MRS approach.
The MRS representation $(\tilde{a}_1, \tilde{a}_2, \cdots, \tilde{a}_n)$ of an integer a is such that:

$$a = \tilde{a}_1 + \tilde{a}_2 m_1 + \tilde{a}_3 m_1 m_2 \cdots + \tilde{a}_{n-1} m_1 m_2 ... m_{n-2} + \tilde{a}_n m_1 m_2 ... m_{n-1} \quad \text{with } 0 \leq \tilde{a}_i < m_i \tag{5}$$

This MRS representation can be deduced modulo the m_i directly from the RNS representation $(a_1, a_2, \cdots, a_n)$ with a classical radix conversion algorithm (which can be seen also as the

Newton interpolation):

$$
\begin{aligned}
&\tilde{a}_1 = a_1\\
&\tilde{a}_2 = \Big|(a_2-\tilde{a}_1)m_{1,2}^{-1}\Big|_{m_2}\\
&\tilde{a}_3 = \Big|(\Big|(x_3-\tilde{a}_1)m_{1,3}^{-1}\Big|_{m_3}-\tilde{a}_2)m_{2,3}^{-1}\Big|_{m_3}\\
&\tilde{a}_4 = \Big|(\Big|(\Big|(x_4-\tilde{a}_1)m_{1,4}^{-1}\Big|_{m_4}-\tilde{a}_2)m_{2,4}^{-1})\Big|_{m_4}-\tilde{a}_3)m_{3,4}^{-1}\Big|_{m_4}\\
&\vdots\\
&\tilde{a}_n = \Big|(.\Big|(\Big|(x_n-\tilde{a}_1)m_{1,n}^{-1}\Big|_{m_n}-\tilde{a}_2)m_{2,n}^{-1})\Big|_{m_n}-.-\tilde{a}_{n-1})m_{n-1,n}^{-1}\Big|_{m_n}
\end{aligned}
\tag{6}
$$

where $m_{i,j}^{-1}$ is the inverse of m_i modulo m_j, and $(x_1, x_2, ..., x_n)$ is the RNS representation of a.

We transpose Equation (5) modulo the elements of the new RNS base $\mathcal{B}'$ with $j = n+1...2n$, applying the Horner scheme.

$$
a_j = \big|\tilde{a}_1 + m_1|\tilde{a}_2 + m_2|\tilde{a}_3 \cdots + m_{n-2}|\tilde{a}_{n-1} + m_{n-1}\tilde{a}_n|_{m_j} \cdots |_{m_j}|_{m_j}\big|_{m_j} \tag{7}
$$

Due to the properties of the m_i, this last transformation does not need any products, only shift and add operations.

We note that we use this MRS approach for the two base extensions of Algorithm 3.

In **[7]** the bases are more general with the form $2^k - c_i$, this work presents an RNS-MRS conversion using properties of the Montgomery reduction for the multiplication by the $m_{i,j}^{-1}$, for that they need very small c_i (few bits, 8 or 10), hence they maintain some products in the modular reductions over the bases elements. On the other hand, in **[6]** the goal was to construct optimal bases for all operations with sparse elements and constants, but the number of interesting bases is small and the elements are bigger (at least 64 bits). Here, we had observed that it is not the best choice to try to minimize the cost of the products by the $m_{i,j}^{-1}$, it is really better to concentrate on the inner reductions of the RNS.

3.4. Analysis of the complexity. — To easily compare the RNS approach of this paper to classical number systems one, we evaluate the cost of Algorithm 3 by giving the number of word-multiplications needed, as it is commonly done in the literature. As it is written in the introduction we consider an architecture as generic as possible where additions and shifts are more efficient than products (and more however than division).

In step 1 and 3, the evaluations of q and r are made in RNS independently for each modulus. The value q is computed in the base $\mathcal{B}$, which represents n RNS-digit multiplications by a constant value $|p|_{m_i}^{-1}$. The calculation of r involves $2n$ RNS-digit multiplications in the base $\mathcal{B}'$.

Now for the base extension, multiplications occur only in the conversion to Mixed Radix given by Equation 6. The number of RNS-digit multiplications by a constant (the $|m_i|_{m_j}^{-1}$) is:

$$
T_{RNS-MRS}(n) = \frac{n^2-n}{2} \text{ RNSdigit-products} \tag{8}
$$

In the conversion from MRS to RNS, the basic operation $|a + m_i b|_{m_j}$ is split in two: first we consider $a + m_i b = a + 2^k b - 2^{t_i} b \pm b$ which is performed with two additions ($a + 2^k b$ is just a concatenation), then we deal with the reduction modulo m_j which represents 3 additions, as we note in section 3.1. We notice that we are not forced to make a complete reduction, we can move on the calculus with values lower than $2m_j$. This remark implies that $2^{t_i} \pm 1 < 2^{\frac{k}{2}-1}$ for a direct reuse of the result inside a series of calculus (for example in the Horner process). In fact, the only complete reduction modulo m_j is requested at the last step of the construction of the MRS digits. Thus, the evaluation of each a_j requests $5(n-1)$ word-additions. If we refer to Brent and Kung theorem **[11]**, the ratio, taking into account the area, between the complexity of a multiplication and an addition is in $\Theta(\sqrt{k})$, which represents, for 32 or 64 bits words (and greater), a ratio α greater than 10 for a RNSdigit-product compared to an addition. Software algorithms are described in **[12]**, and hardware ones can be found in **[23]**.

$$T_{MRS-RNS}(n) = \frac{5}{\alpha}(n^2 - n) \text{ RNSdigit-products} \tag{9}$$

As we need two extensions in Algorithm 3, the total complexity of this algorithm is :

$$T_{Algo3}(n) = n^2 - n + 2\frac{5}{\alpha}(n^2 - n) + 3n = \frac{\alpha + 10}{\alpha}(n^2 - n) + 3n \text{ RNSdigit-products} \tag{10}$$

As $\alpha > 10$, this approach is asymptotically better than previous results **[30, 27]** which are, at least, in $2n^2$.
If we operate with an architecture of n basic word-arithmetic cells, Algorithm 3 can be performed in a parallel manner. In this case, due to the independency of the RNS, the evaluation requires, $(n-1)$ steps for the conversion RNS-MRS, and $\frac{5}{\alpha}(n-1)$ for the RNS-MRS conversion, one step for each RNS product. Hence, a parallel evaluation of Algorithm 3 can be done in $\frac{2\alpha+10}{\alpha}(n-1) + 3$ steps.

3.5. Discussion of the advantages of this approach. — Even though the number of operations needed for the reduction is somewhat higher than in a classical representation $(n^2 + n)$ words products for the classical Montgomery reduction), RNS has some important advantages. In general, a modular reduction occurs after a series of operations such as a summation of products. Because in RNS additions and multiplications are very cheap, the gain obtained in the computation of these operations compensate the cost of the reduction. For example, Assuming that for ECC size (192 to 512 bits), the classical multiplication needs n^2 word-products, the RNS approach proposed in this paper is quite interesting, the modular multiplication which represents $2n^2 + n$ word-products in classical systems, is done in $\left(\frac{\alpha+10}{\alpha}(n^2 - n) + 3n\right) \times 1.1$ word-products in RNS.
Furthermore, due to the independence of the RNSdigit operations, computations can be performed in a random order and, consequently, the architecture can be parallelized. With n basic operators (arithmetic units), the time complexity of a modular multiplication can reach 2 modular digit-operations for the multiplication (or multiplication-addition) and $\frac{2\alpha+10}{\alpha}(n-1) + 3$ for the modular reduction.
These two points are developed in section 4, where by reformulating addition formulae on elliptic curves, we propose solutions up to 30% better than the classical approaches.

Another advantage of the RNS is the flexibility of the architecture. With a given structure of n modular digit operators, it is possible to handle any values of p which satisfy : $4 \times p < M$. Thus, by reinitializing the pre-computed values of Algorithm 3, the system can be adapted for a new value of p. If p is relatively small as compared to M, we can adjust the RNS base by reducing the number of elements.
We also note that all the products which occur in the RNS conversions, are with constant factors. Hence, some of them can be reduced to some shift and add operations. For example, in [**6**] the elements of the bases are selected for having low Hamming weight inverses. As, it is shown in [**34**], there are many different rewriting ways for minimizing these products. For example, the constant can be decomposed in a product of two low Hamming weight factors. However, the present work is focused on a most general case for having a complexity study suitable to our objectives.

4. SPA-resistant arithmetic on elliptic curves optimized for RNS representation

The aim of this section is to rewrite or modify the formulae given in section 2.2 in order to minimize the number of modular reductions, since this is the most expensive operation in RNS representation. Thus we have to group together several multiplications before performing only one reduction.

4.1. Unified addition formulae. — This can be well illustrated by elliptic curves in Hessian form. We give here the steps required by the formulae (1).

step	operations	red.	mul.
computation of	$A = X_2Y_1,\ \ B = Y_1Z_2,\ \ C = X_1Y_2$	3	3
intermediate products	$D = Y_2Z_1,\ \ E = X_1Z_2,\ \ F = X_2Z_1$	3	3
computation of X_3	$AB - CD$	1	2
computation of Y_3	$EC - FA$	1	2
computation of Z_3	$EB - FD$	1	2

Thus the total cost in RNS representation is 9 modular reductions, which has to be compared to the 12 modular multiplications in standard representation. Concerning the Jacobi quartic, the cost in terms of modular reductions is given in [**21**] and is equal to 10, whereas 12 multiplications are necessary. Finally, we give details of the steps for computing the sum of two points using unified addition formulae for a curve given in short Weierstrass form.

step	operations	red.	mul.
λ_n	$A = X_2Z_1,\ B = X_1Z_2,\ C = Z_1Z_2,\ D = aC$	4	4
	$\lambda_n = (A+B)^2 - AB + CD$	1	3
λ_d	$E = Y_1Z_2 + Y_2Z_1,\ \lambda_d = EC$	2	3
intermediate	$F = E\lambda_d,\ G = \lambda_n^2,\ H = F(A+B)$	3	3
X_3	$2\lambda_d(G - H)$	1	1
Y_3	$\lambda_n(3H - 2G) - F^2$	1	2
Z_3	$2\lambda_d^3$	2	2

In this case, the total cost in RNS is 14 modular reductions, whereas 18 multiplications must be performed.

Thus, the computation of unified addition requires fewer reductions than multiplications. This means that using the RNS representation can become attractive in terms of performance. A detailed comparison is given in Section 5.

4.2. Montgomery formulae. — We have seen in Section 2.2.2 that, in the restrictive case of curves in Montgomery form, 10 modular multiplications are required at each step of the Montgomery ladder but it is not possible to have less reductions because of the degree of the formulae. The case of curves in short Weierstrass form is more interesting. Following the strategy used for the unified addition formulae (accumulating multiplications before reduction) leads to 16 reductions and 19 multiplications. But it is actually possible to further reduce this complexity by resuming, from the beginning, the Montgomery ladder from a more theoretical standpoint.
The Montgomery ladder is based on the fact that the y-coordinate carry only minor information. Indeed, it only allows to distinguish a point and its opposite (or equivalently a point and its image under the hyperelliptic involution). Thus, from a theoretical standpoint, dealing with the x-coordinate only results in working on the quotient of the curve by the hyperelliptic involution, which is called the Kummer variety. Of course, taking such a quotient implies that it is not possible to add two different points since $P+Q$ and $P-Q$ are not equal in the Kummer variety. However, doubling is still possible (it is easy to discern $P+P$ and $P-P=\mathcal{O}$) and if $P-Q$ is known, it is possible to discern $P+Q$ and $P-Q$. More precisely, it is proved in **[25]** that there are biquadratic forms M_x, M_z and M_{xz}, such that for any points $P=(X_P,Z_P)$ and $Q=(X_Q,Z_Q)$ on the Kummer variety, we have

$$\begin{array}{lcl} 2X_{P+Q}X_{P-Q} & = & M_x, \\ X_{P+Q}Z_{P-Q}+X_{P-Q}Z_{P+Q} & = & M_{xz}, \\ 2Z_{P+Q}Z_{P-Q} & = & M_z. \end{array}$$

These biquadratic forms are explicitely given by

$$\begin{array}{rcl} M_x & = & (X_PX_Q-aZ_PZ_Q)^2-4bZ_PZ_Q(Z_PX_Q+X_PZ_Q), \\ M_{xz} & = & X_PX_Q(Z_PX_Q+X_PZ_Q)+Z_PZ_Q(a(Z_PX_Q+X_PZ_Q)+2bZ_PZ_Q), \\ M_z & = & (Z_PX_Q-X_PZ_Q)^2. \end{array}$$

If $P-Q$ is known, one can easily deduce formulae to compute X_{P+Q} and Z_{P+Q} from these biquadratic forms. In fact, only two of them are necessary. For instance, formulae obtained (in another way) by Brier and Joye in **[13]** and given in Proposition 2 can be easily deduced from M_x and M_z. Here, we use M_{xz} and M_z, in order to minimize the number of reductions.

$$\begin{array}{rcl} X_{P+Q} & = & 2(M_{xz}-X_{P-Q}Z_{P+Q}) \\ Z_{P+Q} & = & M_z \end{array}$$

Let us note that the theory of Kummer varieties also provides formulae for doubling but these always lead to the same formulae as in Proposition 2. Therefore, we have the following theorem.

Theorem 1. — *Let E be an elliptic curve defined over $\mathbb{F}_p$ by (3). 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 $P-Q=(x,y)$*

is known in affine coordinates. Then

$$\begin{aligned} X_{P+Q} &= 2\left(X_P X_Q (Z_P X_Q + X_P Z_Q) + Z_P Z_Q (a(Z_P X_Q + X_P Z_Q) + 2bZ_P Z_Q) - xZ_{P+Q}\right), \\ Z_{P+Q} &= (X_P Z_Q + X_Q Z_P)^2 - 4X_P X_Q Z_P Z_Q, \\ X_{2P} &= \left(X_P^2 - aZ_P^2\right)^2 - 8bX_P Z_P^3, \\ Z_{2P} &= 4X_P Z_P \left(X_P^2 + aZ_P^2\right) + 4bZ_P^4. \end{aligned}$$

Finally, we detail the steps for computing these expressions

step	operations	red.	mul.
preliminary computations	$A = Z_P X_Q + X_P Z_Q,\ B = 2X_P X_Q$	2	3
	$C = 2Z_P Z_Q,\ D = aA + bC$	2	3
Z_{P+Q}	$A^2 - BC$	1	2
X_{P+Q}	$BA + CD + 2xZ_{P+Q}$	1	3
preliminary computations	$A = 2X_P Z_P,\ B = X_P^2,\ C = Z_P^2$	3	3
	$D = -4bA,\ E = aA$	2	2
X_{2P}	$BD + (C - E)^2$	1	2
Z_{2P}	$2B(C + E) - AD$	1	2

In this case, the total cost for each bit of the exponent in RNS representation is 13 modular reductions and 20 multiplications whereas 19 base field multiplications must be performed in a standard representation. Hence the use of an arithmetic which complexity is concentrated on the reduction step (as the RNS) becomes very attractive with these new formulae.
It is interesting to note that, contrary to the case of the standard representation, the extra cost for curves in short Weierstrass form compared to (more specific) curves in Montgomery form is not too large (33% in RNS representation compared to 90% in standard representation).
Lastly, if a (or b) is a small number, the cost becomes 12 modular reductions whereas 17 multiplications must be performed in a standard representation. Let us now show that we can almost always assume that either a or b is small.

4.3. Rescaling the constant to a small value. — This section is not specific to the RNS representation and can be applied in other contexts. It is motivated by the fact that there are 2 multiplications by a in the general formulae for the Montgomery ladder. Thus, the gain will be attractive if a can be rescaled to a small value.
The standard way to perform such a rescaling is to find a small k such that $\frac{a}{k}$ is a fourth power u^4 in $\mathbb{F}_p$ and to use the isomorphism $(x, y) \mapsto \left(\frac{x}{u^2}, \frac{y}{u^3}\right)$ to send E on the curve E' defined by the equation $y^2 = x^3 + \frac{a}{u^4}x + \frac{b}{u^6}$. However, we can obtain a better result in the context of the Montgomery ladder. Indeed, y is not used in this representation so only u^2 will be used and it is actually sufficient that $\frac{a}{k}$ is a square in $\mathbb{F}_p$. The isomorphism is now defined over $\overline{\mathbb{F}_p}$, so that it is easier to use a change of variables to describe the rescaling.

Theorem 2. — *Let E be an elliptic curve defined over $\mathbb{F}_p$ by (3) and k be a small integer such that $\frac{a}{k}$ is a square v^2 in $\mathbb{F}_p$. 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 $P - Q = (x, y)$ is known in affine coordinates.*

Put $Z' = vZ$. If $b' = \frac{b}{v^3}$ and $x' = \frac{x}{v}$ are precomputed, we have:

$$\begin{aligned} X_{P+Q} &= -4b'Z'_PZ'_Q(X_PZ'_Q + X_QZ'_P) + (X_PX_Q - kZ'_PZ'_Q)^2, \\ Z'_{P+Q} &= x'(X_PZ'_Q - X_QZ'_P), \\ X_{2P} &= \left(X_P^2 - kZ_P'^2\right)^2 - 8b'X_PZ_P'^3, \\ Z'_{2P} &= 4Z'_P\left(X_P^3 + kX_PZ_P'^2 + b'Z_P'^3\right). \end{aligned}$$

In this case, pseudo-addition can be performed in 9 multiplications and doubling in 8. Of course the same idea can be applied to formulae optimized for the RNS representation given in Section 4.2.

As the constraint on k has been relaxed compared to former results in the literature ($\frac{a}{k}$ must be a square, not necessarily a fourth power), it is easier to rescale a to a small value in the context of Montgomery ladder than in the general context. Using the properties of the Legendre symbol, it is easy to prove that k is either 1 or the smallest non-square in $\mathbb{F}_p$ and that the proportion of prime fields such that the n first prime numbers are squares is only $\frac{1}{2^n}$. Anyway, if k is too large to neglect the multiplication by k, a can be rescaled to a small value thanks to isogenies [**14**]. Finally, the method explained for rescaling a to a small value can also be applied to b if there is a small k such that $\frac{4b}{k}$ is a cube in $\mathbb{F}_p$ which leads to the same gain (2 multiplications). This method is particularly well suited if $p \equiv 2 \bmod 3$ since any element in $\mathbb{F}_p$ is a third power. Then b can always be rescaled to 1 in this case.

In conclusion, the probability that neither a nor b can be rescaled (by using Z' or isogenies) to a small value is very low, especially in the Montgomery ladder context.

5. Performance comparisons

In this section, we compare the complexities of our approach to those using Montgomery modular multiplication. First, we summarize the complexities for base field operations in Table 1. Table 2 (resp. 3) shows the number of operations required for a doubling **or** an

Operation	RNS (in RNSdigit-products)	Montgomery (in word-products)
Multiplication	$2n$	n^2
Reduction	$\frac{\alpha+10}{\alpha}(n^2-n)+3n$	n^2+n

TABLE 1. Complexities for performing a multiplication and a modular reduction in RNS and with Montgomery approach for two n-word integers. A RNSdigit-product is equivalent to 1.1 word-product (see section 3.1). α represents the ratio between a RNSdigit-product and an addition (in general $\alpha > 10$ for words with more than 32 bits).

addition (resp. a doubling **and** a pseudo-addition) for the different representations of the curve we chose to deal with in this paper.

It is then easy to deduce the global complexity in each case. For instance, one step of the Montgomery exponentiation algorithm using the formulae given in Section 2.2 (for the Montgomery approach) or section 4.2 (for our approach) when a is small requires $17n^2 + 14(n^2+n)$ operations with Montgomery modular multiplication, and $\left(18(2n) + 12(\frac{\alpha+10}{\alpha}(n^2-n)+3n)\right) \times 1.1$

Curve representation	RNS representation	Standard representation
Hessian form	9 red. and 12 mul.	12 mul. and 9 red.
Jacobi form	10 red. and 12 mul.	12 mul. and 10 red.
Unified Weierstrass form	14 red. and 18 mul.	18 mul. and 14 red.

TABLE 2. Number of operations in RNS and standard representation for a unified addition

Curve representation	RNS representation	Standard representation
Montgomery ladder	13 red. and 20 mul.	19 mul. and 16 red.
Montgomery ladder (a small)	12 red. and 18 mul.	17 mul. and 14 red.

TABLE 3. Number of operations in RNS and standard representation for each bit of the exponent of a Montgomery ladder

using RNS. We summarize, in Table 4, the word complexity for each representation of the curve we considered in this paper (i.e., those having SPA-resistance properties). We also give these complexities for usual ECC sizes for a 32-bit architecture. All these complexities are given for one basic step of the scalar multiplication. For Montgomery ladder, such a step always requires both an pseudo-addition and a doubling, so that the complexities are easy to compute. For unified formulae, we assume that this step requires 1.25 unified additions in average using, for example, a sliding window method with window size 3. This is not necessarily the best choice (for example in 512 bits) but this has no effect on our comparisons.

The complexities we obtain in RNS are always asymptotically better than in the classical representation. This is due to the fact that we optimized formulae on elliptic curves in order to minimize the number of reductions. As a consequence, our method becomes more interesting for high level of security. For example, the gain obtained is anecdotal for unified formulas in 192 bits but becomes interesting for higher level of security. The gain is also important for the Montgomery ladder because we discovered new formulae which are well adapted to the RNS representation of numbers. Moreover, all of the advantages of the RNS arithmetic become evident when a parallel architecture is used. Indeed, assuming that we have an architecture equivalent to n word-operators on a single word-bus, Table 5 shows the complexities of the different approaches in number of word operations. Note that we only give these complexities in the case of the Montgomery ladder with a small in order to simplify the paper. The complexities for the other curves representations can be easily deduced from Table 4.
The estimation of the cost for the multiplication and for the Montgomery parallel product are based on systolic implementations **[39]** or on parallel implementations **[15, 42]**, where the given architectures are respectively in $O(n^2/log(n)^2)$ and $O(n^2)$ for the area and $O(\log(n))$ for the time. As we did not find an explicit complexity for multiplication using a $O(n)$ area architecture, we give two values for the complexity. The first one is minimal but certainly not realistic. The second one, which is not necessarily optimal, takes into account that :
– each product of a number by a digit will produce two numbers (high and low parts),

Curve rep.	size in bit	RNS	Montgomery	ratio in %
Hessian form	$32n$	$1.25\left(15n^2+36n\right)\times 1.1$	$1.25(21n^2+9n)$	
	192	1039	1012	-2.6 %
	256	1716	1770	3.1 %
	512	6072	6900	13.6 %
Jacobi form	$32n$	$1.25\left(16.6n^2+37.3n\right)\times 1.1$	$1.25(22n^2+10n)$	
	192	1133	1065	-6 %
	256	1877	1860	-1 %
	512	6688	7240	8.3 %
Unif. Weierstrass	$32n$	$1.25\left(23.3n^2+54.6n\right)\times 1.1$	$1.25(32n^2+14n)$	
	192	1606	1545	-3.8 %
	256	2654	2700	1.7 %
	512	9416	10520	11.7 %
Montg. ladder	$32n$	$\left(21.6n^2+57.3n\right)\times 1.1$	$35n^2+16n$	
	192	1236	1356	9.7 %
	256	2029	2368	16.7 %
	512	7110	9216	29.6 %
M. lad. (a small)	$32n$	$\left(20n^2+52n\right)\times 1.1$	$31n^2+14n$	
	192	1135	1200	5.7 %
	256	1865	2096	12.3 %
	512	6547	8160	24.6 %

TABLE 4. Cost in word-products (32-bits) of one scalar multiplication iteration (for $\alpha = 15$)

Operation	RNS	Montgomery
Multiplication	2×1.1	$n\ldots 2n$
Reduction	$\left(\frac{2a+10}{a}(n-1)+3\right)\times 1.1$	$2n\ldots 3n$
One iteration of algorithm 2	$35.2n+44$	$44n\ldots 75n$

TABLE 5. Number of cycles with parallel implementations on an n word-operators structure. ($\alpha = 15$)

– a carry-save adder will need an extra register for storing the carry and a final adder for absorbing these carries,
– 32-bit words look-up tables are not reasonable.

Then, to get an idea with ECC key size, we compare three different implementations in table 6 for the number of operations required for one step of the Montgomery scalar multiplication on an elliptic curve in Weierstrass form with a small.

In this configuration, the RNS becomes very attractive compared to the Montgomery arithmetic in terms of efficiency for a leak-resistant implementation of elliptic curve cryptosystems, even if we use our non-realistic lower bound for the comparison.

$[\log_2 p]$	word	RNS	Montgomery	ratio inc
192	6	255.2	264 . . . 450	12.1%. . . 48.4%
256	8	325.6	352. . . 600	15.9%. . . 50.6%
512	16	606.2	704. . . 1200	21.6%. . . 54%

TABLE 6. Cost in word-products (32-bits) of one scalar multiplication iteration on a parallel architecture

6. Practical implementation

In this paper, we prove that using RNS arithmetic is theoretically interesting for leak-resistant elliptic curve scalar multiplication. This result is based on a fine complexity study which has been made in a context as generic as possible (architecture for which additions and shifts are more efficient than products). In **[27]**, Guillermin produced a practical implementation to validate this work. He used the results provided in this paper but had to adapt them to his specific architecture. For example, he used an algorithm for base extension deduced from the CRT method **[30]**. It involves 2 times more products than the MRS method (chosen for our complexity study) and less additions. This is justified by his choice of architecture where products are especially cheap. The FPGA implementation obtained is the fastest one for elliptic curves defined over non-Mersenne prime field. It would be interesting in the future to have implementation on other platforms (like ASIC, smart cards, other FPGA, etc).

7. Conclusion

By combining Residue Number System and SPA-resistant arithmetic on elliptic curves, we obtained an efficient and secure implementation of elliptic curves cryptosystems, especially suitable for parallel architectures.
Since the expensive operation in RNS is the reduction, we proposed to rewrite formulae for elliptic curve SPA-resistant arithmetic in order to minimize the number of reductions even if the number of multiplications is increased. In the case of the Montgomery ladder on elliptic curves in Weierstrass form, we obtain new formulae which are better suited to RNS representation of numbers and we show why multiplications by one of the coefficients of the curve can be neglected in most cases.
We also give an in-depth analysis of the complexity of the RNS reduction. We thus realized that some improvements could be made to obtain a final complexity of $\frac{\alpha+10}{\alpha}(n^2-n)+3n$ for a n-word number.
It is clear that, without the results we obtained in these two directions, the combination of RNS arithmetic and elliptic curves will be possible but less convincing. Thus, we theoretically obtain an efficient leak-resistant arithmetic especially for high security levels and in the case of the Montgomery ladder on elliptic curves in Weierstrass form. This has been validated by a successful efficient FPGA implementation **[27]** and generalized to pairing computations in **[22, 16]**.
This means that both on the theoretical and on the practical point of view, our approach is very attractive and promising. Of course, it is particularly interesting from a hardware

standpoint since the RNS representation of numbers has many advantages (easy to implement and to parallelize, flexibility).

References

[1] Bajard, J.C., Didier, L.S., Kornerup, P.: A RNS Montgomery's Modular Multiplication. IEEE Transactions on Computers, volume 47, no. 7, July 1998.

[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., Duquesne, S., Ercegovac M. and Meloni N.: Residue systems efficiency for modular products summation: Application to Elliptic Curves Cryptography, in Advanced Signal Processing Algorithms, Architectures, and Implementations XVI, part of the SPIE Optics & Photonics 2006 Symposium. August 2006 San Diego, USA.

[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., Kaihara, M., Plantard Th.: Selected RNS Bases for Modular Multiplication in Proceedings of the 19th IEEE symposium on Computer Arithmetic (ARITH 19) June 2009, Portland, USA.

[7] Bajard, J.C., Meloni, N., Plantard, T.: Efficient RNS bases for Cryptography. IMACS'05, Applied Mathematics and Simulation, (2005).

[8] Barrett, P.: Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. Advances in Cryptology CRYPTO LNCS **263** (1986) 311–326.

[9] Billet, O., Joye, M.: The Jacobi Model of an Elliptic Curve and Side-Channel Analysis. Applied Algebra, Algorithms and Error-Correcting Codes, LNCS **2643** (2003) 34–42.

[10] Bosselaers, A., Govaerts, R., Vandewalle. J.: Comparison of the three modular reduction functions LNCS **773** (1994) 175–186.

[11] Brent, R.P., Kung, H.T.: The Area-Time Complexity of Binary Multiplication. Journal of the Association for Computing Machinery, Vol 28, No 3, July 1981, 521–534.

[12] R. P. Brent, P. Zimmermann, Modern Computer Arithmetic, Cambridge Monographs on Computational and Applied Mathematics (No. 18), Cambridge University Press, November 2010, 236 pages.

[13] Brier, E., Joye, M.: Weierstrass Elliptic Curves and Side-Channel Attacks. Public Key Cryptography, LNCS **2274** (2002) 335–345.

[14] Brier, E., Joye, M.: Fast Point Multiplication on Elliptic Curves Trough Isogenies, Applied Algebra, Algorithms and Error-Correcting Codes, Lecture Notes in Comput. Sci., vol.2643, Springer, Berlin, 2003, pp. 43–50.

[15] Bunimov, V., Schimmler, M.: Efficient Parallel Multiplication Algorithm for Large Integers Euro-Par 2003, International Conference on Parallel and Distributed Computing (2003) LNCS **2790**, pp. 923–928.

[16] Cheung, R., Duquesne, S., Fan, J., Guillermin, N., Verbauwhede, I. and Yao, G.: FPGA Implementation of Pairings Using Residue Number System and Lazy Reduction. CHES 2011, LNCS **6917** 421–441.

[17] Chung, J., Hasan, A.: More generalized Mersenne numbers. SAC 2003, LNCS **3006** (2003) 335–347

[18] 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).

[19] Cohen, H., Frey, G.: Handbook of elliptic and hyperelliptic curve cryptography. Discrete Math. Appl., Chapman & Hall/CRC (2006).

[20] Coron, J.S.: Resistance against differential power analysis for elliptic curve cryptosystems. CHES'99, LNCS **1717** (1999) 292–302.

[21] Duquesne, S.: Improving the Arithmetic of Elliptic Curve in Jacobi Model. Information Processing Letters **104:3** (2007) 101–105.

[22] Duquesne, S.: RNS arithmetic in Fpk and application to fast pairing computation, Journal of Mathematical Cryptology **5:1** (2012) 51–88.

[23] M.D. Ercegovac and T. Lang, Digital Arithmetic, Morgan Kaufmann Publishers - An Imprint of Elsevier Science, 2004

[24] Fischer, W., Giraud, C., Knudsen, E.W., Seifert, J. P.: Parallel scalar multiplication on general elliptic curves over $\mathbb{F}_p$ hedged against Non-Differential Side-Channel Attacks. Preprint.

[25] Flynn, E.V.: An explicit theory of heights. Trans. Amer. Math. Soc. **347:8** (1995) 3003–3015.

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

[27] Guillermin, N.: A high speed coprocessor for elliptic curve scalar multiplications over $\mathbb{F}_p$. CHESS 2010, LNCS (2010)

[28] Izu, T., Takagi, T.: A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks. Public Key Cryptography, LNCS **2274** (2002) 280–296.

[29] Joye, M., Quisquater, J.J.: Hessian Elliptic Curves and Side-Channel Attacks. CHES 2001, LNCS **2162** 402–410.

[30] Kawamura, Sh., Koike, M., Sano, F., Shimbo,A.: Cox-rower architecture for fast parallel montgomery multiplication. In Advances in Cryptology EUROCRYPT 2000, LNCS, vol. 1807, p. 523-538, 2000.

[31] Knuth, D.: Seminumerical Algorithms. The Art of Computer Programming, vol. 2. Addison-Wesley (1981).

[32] Kocher, P.C.: Timing attacks on implementations of DH, RSA, DSS and other systems. CRYPTO'96, LNCS **1109** (1996) 104–113.

[33] Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. CRYPTO'99, LNCS **1666** (1999) 388–397.

[34] V. Lefèvre. Multiplication by an integer constant: lower bounds on the code length. In Proceedings of the 5th Conference on Real Numbers and Computers, pages 131-146, Ecole Normale Superieure de Lyon, France, september 2003.

[35] Liardet, P. Y., Smart, N.: Preventing SPA/DPA in ECC systems using the Jacobi form. CHESS 2001, LNCS **2162** 391–401.

[36] Montgomery, P.L.: Modular multiplication without trial division. Math. Comp. **44:170** (1985) 519–521.

[37] Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comp. **48:177** (1987) 243–164

[38] 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.

[39] Orlando, G., Paar, C.: A scalable GF(p) elliptic curve processor architecture for programmable hardware. Cryptographic Hardware and Embedded Systems, LNCS **2162** (2001) 348–363.

[40] Posch, K.C., Posch, R.: Modulo reduction in residue number systems. IEEE Transaction on Parallel and Distributed Systems **6:5** (1995) 449–454.

[41] Quisquater, J.J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. e-smart 2001, LNCS **2140** (2001) 200–210.

[42] 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.

[43] Shenoy, A.P., Kumaresan, R.: Fast base extension using a redundant modulus in RNS. IEEE Transactions on Computer **38:2** (1989) 292–296.

[44] Solinas, J.: Generalized Mersenne numbers. Research Report CORR-99-39, Center for Applied Cryptographic Research, University of Waterloo (1999)

[45] Svoboda, A. and Valach, M.: Operational Circuits. Stroje na Zpracovani Informaci, Sbornik III, Nakl. CSAV, Prague, (1955) 247-295.

[46] Szabo, N.S., Tanaka, R.I.: Residue Arithmetic and its Applications to Computer Technology. McGraw-Hill (1967)

Appendices

16 **coprimes of** 32 **bits with** $c_i = 2^{t_i} \pm 1 < 2^{12}$**.** —

$m_1 = 10000000000000000000000000000000$
$m_2 = 11111111111111111111111111111111$
$m_3 = 11111111111111111111111111111101$
$m_4 = 11111111111111111111111111111011$
$m_5 = 11111111111111111111111111110111$
$m_6 = 11111111111111111111111111110001$
$m_7 = 11111111111111111111111111101111$
$m_8 = 11111111111111111111111111011111$
$m_9 = 11111111111111111111111111000001$
$m_{10} = 11111111111111111111111110111111$
$m_{11} = 11111111111111111111111101111111$
$m_{12} = 11111111111111111111111100000001$
$m_{13} = 11111111111111111111110111111111$
$m_{14} = 11111111111111111111101111111111$
$m_{15} = 11111111111111111111011111111111$
$m_{16} = 11111111111111111111000000000001$

18 **coprimes of** 32 **bits with** $c_i = 2^{t_i} \pm 1 < 2^{15}$**.** —

$m_1 = 10000000000000000000000000000000$
$m_2 = 11111111111111111111111111111111$
$m_3 = 11111111111111111111111111111101$
$m_4 = 11111111111111111111111111111011$
$m_5 = 11111111111111111111111111110111$
$m_6 = 11111111111111111111111111110001$
$m_7 = 11111111111111111111111111101111$
$m_8 = 11111111111111111111111111011111$
$m_9 = 11111111111111111111111111000001$
$m_{10} = 11111111111111111111111110111111$
$m_{11} = 11111111111111111111111101111111$
$m_{12} = 11111111111111111111111100000001$
$m_{13} = 11111111111111111111110111111111$
$m_{14} = 11111111111111111111101111111111$
$m_{15} = 11111111111111111111011111111111$
$m_{16} = 11111111111111111111000000000001$
$m_{17} = 11111111111111111110111111111111$
$m_{18} = 11111111111111111101111111111111$

64 **coprimes of 32 bits with** $c_i = 2^{t_i} \pm 2^{s_i} \pm 1 < 2^{15}$**.** —

m_1 = 10000000000000000000000000000000
m_2 = 11111111111111111111111111111111
m_3 = 11111111111111111111111111111101
m_4 = 11111111111111111111111111111011
m_5 = 11111111111111111111111111110111
m_6 = 11111111111111111111111111110001
m_7 = 11111111111111111111111111101111
m_8 = 11111111111111111111111111101001
m_9 = 11111111111111111111111111100101
m_{10} = 11111111111111111111111111100011
m_{11} = 11111111111111111111111111011111
m_{12} = 11111111111111111111111111010001
m_{13} = 11111111111111111111111111000001
m_{14} = 11111111111111111111111110111111
m_{15} = 11111111111111111111111110100001
m_{16} = 11111111111111111111111110001001
m_{17} = 11111111111111111111111101111111
m_{18} = 11111111111111111111111101111001
m_{19} = 11111111111111111111111101110001
m_{20} = 11111111111111111111111101100001
m_{21} = 11111111111111111111111100011111
m_{22} = 11111111111111111111111100000001
m_{23} = 11111111111111111111111010000001
m_{24} = 11111111111111111111111000111111
m_{25} = 11111111111111111111111000100001
m_{26} = 11111111111111111111111000000101
m_{27} = 11111111111111111111111000000011
m_{28} = 11111111111111111111110111111111
m_{29} = 11111111111111111111110111110001
m_{30} = 11111111111111111111110111000001
m_{31} = 11111111111111111111110110000001
m_{32} = 11111111111111111111110001111111
m_{33} = 11111111111111111111110001000001
m_{34} = 11111111111111111111110000010001
m_{35} = 11111111111111111111110000000111
m_{36} = 11111111111111111111101111111111
m_{37} = 11111111111111111111101000000001
m_{38} = 11111111111111111111100011111111
m_{39} = 11111111111111111111100010000001
m_{40} = 11111111111111111111100001000001
m_{41} = 11111111111111111111100000100001
m_{42} = 11111111111111111111100000001111
m_{43} = 11111111111111111111100000000011
m_{44} = 11111111111111111111011111111111
m_{45} = 11111111111111111111011111100001
m_{46} = 11111111111111111111000100000001
m_{47} = 11111111111111111111000001111111
m_{48} = 11111111111111111111000000000111
m_{49} = 11111111111111111110111111111111
m_{50} = 11111111111111111110111111111001
m_{51} = 11111111111111111110111110000001
m_{52} = 11111111111111111110111000000001
m_{53} = 11111111111111111110010000000001
m_{54} = 11111111111111111110000011111111
m_{55} = 11111111111111111110000010000001
m_{56} = 11111111111111111110000000111111
m_{57} = 11111111111111111110000000001001
m_{58} = 11111111111111111110000000000011
m_{59} = 11111111111111111101111111111111
m_{60} = 11111111111111111101111111000001
m_{61} = 11111111111111111101111100000001
m_{62} = 11111111111111111101100000000001
m_{63} = 11111111111111111101000000000001
m_{64} = 11111111111111111100000100000001

---

4 novembre 2012

J. C. Bajard, LIP6 CNRS - Université Pierre et Marie Curie, Boite courrier 169 - Couloir 26-00, étage 3, Bureau 315, 4 place Jussieu, 75252 PARIS cedex 05, France • *E-mail :* jean-claude.bajard@lip6.fr

S. Duquesne, Université Rennes I, Laboratoire IRMAR, UMR CNRS 6625, Campus de Beaulieu, 35042 Rennes cedex, France • *E-mail :* sylvain.duquesne@univ-rennes1.fr

M. Ercegovac, Computer Science Department, Henry Samueli School of Engineering and Applied Science, University of California, Los Angeles 4731H Boelter Hall, USA • *E-mail :* milos@cs.ucla.edu