Combining Montgomery ladder for elliptic curves defined over $\mathbb{F}_p$ and RNS representation

Jean-Claude Bajard[1] and Sylvain Duquesne[2] and Nicolas Meloni[1,2]

LIRMM[1], I3M[2], Université Montpellier 2

Abstract. In this paper, we combine the RNS representation and the Montgomery ladder on elliptic curves in Weierstrass form. These two techniques are relevant for implementation of elliptic curve cryptography on embedded devices since they have leak-resistance properties. We optimize formulae for addition and doubling of the Montgomery ladder in terms of modular reductions to obtain a competitive and secure implementation. Afterwards, we explain the advantages of the RNS representation, especially in hardware and for embedded devices, and show that, contrary to other approaches, ours takes optimal advantage of a dedicated parallel architecture.

Keywords: Elliptic curves, Montgomery, Leak-resistance, RNS, Modular multiplication.

1 Introduction

Elliptic curve cryptosystems, because of their small key length, has become popular to such a point that they have recently been recommended by the NSA. Their small key size is especially attractive for small cryptographic devices like smart cards, however such devices are sensitive to side channel attacks. These attacks consist in analyzing side channel informations like timings [16], power consumptions [17] or electromagnetic radiations [23] of a device. They have become such a threat that protecting ECC against them has become itself a whole research area giving rise to various countermeasures [11].

The most efficient protection is an algorithm for scalar multiplication due to Montgomery [19] for a family of curves defined over $\mathbb{F}_p$ and generalized in [12], [6] and [13]. This algorithm has many advantages for constrained environments: it is leak-resistant, very simple to implement, careful in memory and does not required precomputations. On the other hand the RNS representation of numbers in $\mathbb{F}_p$ has interesting leak-resistance properties for the arithmetic on the base field and is easily parallelizable in hardware [3] and scalable.

The aim of this paper is to combine these two techniques, especially in the case of curves in Weierstrass form, to obtain an implementation of ECC which is leak-resistant, both at the level of the curve and at the level of the field, and which can be easily and efficiently parallelized in hardware.

In the following, $\mathbf{K}$ denotes a field of characteristic $\neq 2, 3$ (which is a prime field $\mathbb{F}_p$ in practice) and $|n|_2$ denotes the bit length of n.

2 Background properties of the different representations and algorithms

2.1 Modular multiplication

Elliptic curve arithmetic over $\mathbb{F}_p$ mainly involves modular multiplications modulo p. Such operation can be decomposed into one classic multiplication followed by a modular reduction. Because of the small size of the numbers used with ECC (160 bits ie five 32-bits words, the multiplication is performed by the so called schoolbook method. Let us consider A and B two n-word integers given in radix representation (ie $X = \sum_{i=0}^{n} x_i \beta^i$ with $0 \leq a_i < \beta$), then $A \times B$ can be computed by a succession of word multiplications and additions (which will be considered in the following as basic word operations). We can summarize it by the equation

$$A \times B = b_0 A + \beta(b_1 A + \beta(b_2 A \cdots + \beta b_n A) \dots).$$

We get a complexity of n^2 word operations.

The reduction of an integer k modulo another integer p consist in finding the remainder of the euclidean division of k by p. We recall now the Montgomery reduction (as it is what we use in RNS modular multiplication) and then we briefly study reductions using special modulo.

Montgomery general reduction algorithm: In [18] Montgomery proposed to substitute the reduction modulo p by a division by a power of the radix β (a simple shift). The result is not exactly $k \mod p$ but $k\beta^{-n} \mod p$, using Montgomery representation allows to overcome this problem.

Algorithm 1: Montgomery$_p(R)$

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

The complexity of this reduction is known to be $n^2 + n$ word operations [5]. As all the computations can be done in Montgomery representation, we neglect the cost of the conversion from Montgomery to classic representation.

Reduction using special modulo: When using ECC, one can choose the underlying field without restriction. In this case, the cost of a modular reduction can be reduced to some additions. Compared to the cost of a general reduction, in can be considered as almost free. As an example, if the field $\mathbb{F}_p$ is such that p is a Mersenne number (ie $p = 2^k - 1$) then the reduction of a $2n$-word integer R modulo p requires only $2n$ word additions. Just write $R = R_1 2^k + R_0$, then $R \pmod p = R_1 + R_2$, if $R_1 + R_2 \geq p$ then $R \pmod p = R_1 + R_2 - p$. Prime Mersenne numbers are rare, that is why the generalized Mersenne number class

has been introduced [26, 8] (integer of the form $P(2^k)$ where $P(X) = X^n - C(X)$ and C is a polynomial with coefficients equal to $-1, 0$ or 1 and $\deg(C) \leq \frac{n}{2}$) . Modular reduction is still a question of additions but one can not find generalized Mersenne numbers for all the number length. However, a dedicated architecture cannot be used for other values of p.

2.2 The Montgomery scalar multiplication in elliptic curve cryptography

In all elliptic curves based schemes (such as encryption/decryption or signature generation/verification) the dominant operation is the scalar multiplication of points on the curve. Hence, the efficiency of this operation is central in elliptic curve cryptography. This is usually done by using standards scalar multiplication methods such as double and add or sliding window methods combined with recoding of the exponent.

However, these methods are not leak-resistant and their protection is expensive. For example, if one wants to protect a double and add algorithm against side-channel attacks, one can perform extra useless additions [11]. By this way, for each bit of the exponent we perform both an addition and a doubling so that bits of the exponent are indistinguishable.

Montgomery proposed in [19] to work only with the x-coordinate. Of course, the group law is lost but traces remain. So doubling is still possible and the addition of two points P and Q is possible if $P - Q$ is known. Montgomery gives the formulae for those operations when the curve is in Montgomery form, that is defined by an equation of the type

$$By^2 = x^3 + Ax^2 + x.$$

Proposition 1. *Let E be an elliptic curve defined over $\mathbf{K}$ in Montgomery form. Let also $P = (X_p, Y_p, Z_p)$ and $Q = (X_q, Y_q, Z_q) \in E(\mathbf{K})$ given in projective coordinates. Assume that $P - Q = (x, y)$ is known in affine coordinates. Then the X and Z-coordinates for $P + Q$ and $2P$ are given by*

$$\begin{aligned}
X_{p+q} &= \left((X_p - Z_p)(X_q + Z_q) + (X_p + Z_p)(X_q - Z_q)\right)^2, \\
Z_{p+q} &= x\left((X_p - Z_p)(X_q + Z_q) - (X_p + Z_p)(X_q - Z_q)\right)^2, \\
4X_pZ_p &= \left((X_p + Z_p)^2 - (X_p - Z_p)^2\right), \\
X_{2p} &= (X_p + Z_p)^2(X_p - Z_p)^2, \\
Z_{2p} &= 4X_pZ_p\left((X_p - Z_p)^2 + \tfrac{A+2}{4}4X_pZ_p\right).
\end{aligned}$$

By this way, both an addition and a doubling takes only 3 multiplications and 2 squares which is much faster than usual operations [10]. The fact that the difference $P - Q$ must be known to compute $P + Q$ implies that a new algorithm must be used to compute the scalar multiplication of a point G by an integer k. The solution is to use pairs of consecutive multiples of P, so that the difference between the two components of the pair is always known and equal to G. The algorithm for scalar multiplication is as follows:

Algorithm 2: Montgomery_Scalar()

Data: $G \in E(\mathbf{K})$ and $k \in \mathbb{Z}$.
Result: x and z-coordinate of kG.
1 Initialize $Q = (P, Q) = (\mathcal{O}, G)$ where $\mathcal{O}$ is the point at infinity.;
2 If the bit of k is 0, $Q = (2P, P + Q)$.;
3 If the bit of k is 1, $Q = (P + Q, 2Q)$.;
4 After doing that for each bit of k, return P.;

Both an addition and a doubling are done for each bit of the exponent. So the cost of this algorithm is about $10|k|_2$ multiplications for a curve in Montgomery form which is better than other algorithms.
Moreover, the operations we have to perform do not depend on the bits of the exponent so that this method has interesting leak-resistance properties.
Finally, the x-coordinate of kG is usually sufficient but some cryptosystems, like ECDSA, require the y-coordinates. It can easily be recovered [20].

Unfortunately, in odd characteristic, all the elliptic curves cannot be transformed into Montgomery form. This is for example the case for most of the standards. The reason is that any curve which can be transformed into Montgomery form has a 2-torsion point so that its cardinality is not prime (it is divisible by 2).

In all generality, namely when the curve is defined by an equation of the form

$$y^2 = x^3 + ax + b, \tag{1}$$

this method can also be applied but is more time consuming ([6],[12] and [13]).

Proposition 2. *Let E be an elliptic curve defined over $\mathbf{K}$ by (1). Let also $P = (X_p, Y_p, Z_p)$ and $Q = (X_q, Y_q, Z_q) \in E(\mathbf{K})$ given in projective coordinates. Assume that $P - Q = (x, y)$ is known in affine coordinates. Then we obtain the X and Z-coordinates for $P + Q$ and $2P$ by the following formulae :*

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

Addition can be evaluated in 10 multiplications and doubling in 9. Hence, the scalar multiplication can be performed in about $19|n|_2$ multiplications on $\mathbf{K}$ which is not interesting in term of performance but is interesting in term of leak-resistance. Note that, the y-coordinate can also be recovered in this case [6]

Proposition 3. *Suppose that $Q = P + G$ with $G = (x, y)$, $P = (x_p, y_p)$ and $Q = (x_q, y_q)$. Then, if $y \neq 0$, one has*

$$y_p = -\frac{2b + (a + xx_p)(x + x_p) - x_q(x - x_p)^2}{2y}$$

With the Montgomery scalar multiplication method, we always have to perform both an addition and a doubling for each bit of the exponent, so that this method is resistant against side-channel attacks and that's the reason why this method is always interesting even with 19 multiplications at each step.

In this paper, we will explain that one can almost always assume that one of the coefficients (a or b) is a small number (this is already the case in the standards since $a = -3$) so that only 17 multiplications are required. Afterwards, we will rewrite the formulae to minimize the number of modular reductions in order to use the RNS representation of numbers. Let us now give some recall about this system of representation.

3 Residue Number Systems

3.1 Presentation

The Residue Number Systems (RNS) are based on the very known Chinese Remainder Theorem (CRT). They were introduced in computer science in [14] and [27], a good presentation can be found in [15].

These systems are based on the fact that a number x can be represented by its residues $(x_1, x_2, \ldots, x_n)$ modulo a set of coprimes numbers $(m_1, m_2, \ldots, m_n)$, generally called RNS basis. We generally assume that $0 \leq x < M = \prod_{i=1}^{n} m_i$. The elements x_i are called RNS-digits, or simply digits if there is no ambiguity.

The biggest interest of a such system, is to distribute integer operations on the residues values. Large integer operations are made on the residues, in other words on small numbers independently.

We consider in this part a RNS base $(m_1, ..., m_n)$ with elements such that, $m_i = \beta - c_i$ where c_i is small (with few non null digits). 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 is reduced to n digit modular digit-products. A modular digit-product is equivalent to a classical digit product following by few additions (which are due to the number of ones in the binary representation of c_i)[4]. Thus this modular digit-operation can be done in one clock cycle in hardware.

$$r = (a_1 \times b_1 \pmod{m_1}, \ldots, a_n \times b_n \pmod{m_n}) \tag{2}$$

It is clear that if a product is followed by an addition, the cost is just increased of one addition on each modulo, and so done in the same cycle.

$$r = (a_1 \times b_1 + d_1 \pmod{m_1}, \ldots, a_n \times b_n + d_n \pmod{m_n}) \tag{3}$$

Now, we focus our attention on the multiplication modulo p, for this we propose to use the algorithm presented in [1]. This algorithm for two numbers a and b given in RNS, evaluates in fact $r = abM^{-1} \mod p$. To obtain the

right result we need to call it again with r and $M^2 \mod p$ as operands. To prevent this fact, we convert the values in a Montgomery representation where $a' = a \times M \mod p$ which is stable for Montgomery product and addition. Thus, this conversion is done one time at the beginning by calling Montgomery product with a and $M^2 \mod p$ as operands, and one time at this end of the complete cryptographic computing with 1 as second operand. Hence, this transformation will be neglected in the following.

3.2 RNS Montgomery multiplication

This algorithm is a direct transposition of the classical Montgomery method. The main difference is due to the representation system. When Montgomery is applied in a classical radix β representation, the value β^n occurs for reduction, division and Montgomery factor. In RNS this value is replaced by M. Thus an auxiliary RNS Bases is need to handle the inverse of M.

All the operations considered are in RNS. Instructions 1, 2 and 4 deal with

Algorithm 3: MontgM_RNS(a, b, 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 $\gcd(M, M') = 1$; a redundant modulus m_r, $\gcd(m_r, m_i) = 1 \; \forall i = 1...2n$; a positive integer p represented in RNS in both bases such that $0 < (n+2)^2 p < M$ and $\gcd(p, M) = 1$ (p is prime); two positive integers a and b represented in RNS in both bases, with $ab < Mp$.

Result: A positive integer $r \equiv abM^{-1} \pmod p$ represented in RNS in both bases, with $r < (n+2)p$.

1 $r \leftarrow (a \times b)$ in $\mathcal{B}$ and $\mathcal{B}'$;
2 $q \leftarrow (r) \times (-p^{-1})$ in $\mathcal{B}$;
3 $[q \text{ in } \mathcal{B}] \longrightarrow [\hat{q} \text{ in } \mathcal{B}' \text{ and } m_r]$ *First base extension*;
4 $r \leftarrow (r + \hat{q} \times p) \times M^{-1}$ in $\mathcal{B}'$ and m_r;
5 $[r \text{ in } \mathcal{B} \text{ and } m_r] \longleftarrow [r \text{ in } \mathcal{B}']$ *Second base extension*;

RNS operations as presented in the previous section, are linear, so very efficient. But, lines 3 and 5 represent RNS bases extensions which are quadratic, so costly. To reduce this cost, [1] proposes to use two different full RNS extensions.

The extension to base $\mathcal{B}'$ of q, obtained in its RNS form $(q_1, \ldots, q_n)$ in the base $\mathcal{B}$, is done by evaluating first:

$$\sigma_i = q_i \left| M_i^{-1} \right|_{m_i} \mod m_i,$$

and then,

$$\hat{q}_j = \Big| \sum_{i=1}^{n} \left| M_i \right|_{m_j} \sigma_i \Big|_{m_j}, \quad \forall j = n+1 \ldots 2n \quad \text{and} \quad m_r \tag{4}$$

we have $\hat{q} = q + \alpha M$ with $\alpha < n$.

Then we compute in the base $\mathcal{B}'$ the value

$$r = (ab + \hat{q}p)M^{-1} = (ab + qp)M^{-1} + \alpha p < M'. \tag{5}$$

After instruction 4, we get r such that $r \equiv r \equiv abM^{-1} \pmod{p}$. The conditions $\alpha < k$, $q < M$ and $ab < Mp$ gives $\hat{q} < (n+1)M$ and thus $r < (n+2)p < M'$.

In order to use this algorithm within a cryptographic protocol, we must be able to compute $x^2 \mod p$, where x is the output of a former evaluation verifying $x < (n+2)p$. The condition $ab < Mp$ then implies $(n+2)^2p^2 < Mp$ which rewrites:

$$(n+2)^2 p < M. \tag{6}$$

The first base extension then requires, taking into account that m_r is a small power of two [2], $n^2 + n$ elementary modular digit-operations.

The second base extension is done using a different algorithm due to Shenoy and Kumaresan [24] where the factor α is evaluated with the redundant modulus m_r. That represents a cost of $n^2 + 2n$ elementary modular digit-operations.

Hence, a modular multiplication in RNS uses $2n^2 + 8n$ basic modular digit-operations that we can decompose in $2n$ for the product and $2n^2 + 6n$ for the modular reduction (called Montgomery reduction in the next)[2].

3.3 Discussion about the advantages

Even if the number of operations needed is a little bit higher than in a classical representation, RNS got some advantages. It is easy to implement, particularly in hardware, and provides a reduced cost for multiplication and addition and a competitive modular reduction. Furthermore, RNS allow, due to the independence of the modular operations, to perform calculus in a random way and to parallelize the architecture.

Moreover, it is shown that RNS can be used as a leak resistant arithmetic [9, 3], by selecting randomly $\mathcal{B}$ and $\mathcal{B}'$ in a set of $2n$ coprime numbers. It is shown that we got $\binom{2n}{n}$ ways to do the same calculus. Hence, DPA attacks are very difficult to operate. Against SPA it can be possible to exchange the bases during the evaluation.

The parallelization of the architecture, with n basic operators (the extra modulus m_r can be included inside [2]), gives a time complexity of 2 modular digit-operation for the multiplication (or multiplication-addition) and $2n+5$ for the modular reduction. According to this point we see that if we accumulate some operations before reduction we obtain an efficient implementation. We develop this approach in the next section with ECC.

Last advantage of the RNS is the natural scalability of the architecture. With a given structure of n modular digit operators, it is possible to drive many p whose verify (6). It is easy, for smaller values, to use a partial part of the architecture which will run, in this case, only the needed time. In fact, the RNS bases will be adapted to the value p, thus only the needed cells will be active.

4 Improvement of the Montgomery formulae

Let us show that we can almost always assume that either a or b is small.

4.1 First approach

If a is a square in $\mathbf{K}$, it can be replaced by 1 in the formulae by introducing a new coordinate $Z' = \sqrt{a}Z$ which will replace Z.

Proposition 4. *Let E be an elliptic curve defined over $\mathbf{K}$ by (1) such that a is a square in $\mathbf{K}$. Let also $P = (X_p, Y_p, Z_p)$ and $Q = (X_q, Y_q, Z_q) \in E(\mathbf{K})$ given in projective coordinates. Assume that $P - Q = (x, y)$ is known in affine coordinates. Put $Z' = \sqrt{a}Z$. Then we 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 formulae :*

$$\begin{aligned}
X_{p+q} &= -4\tfrac{b}{a\sqrt{a}} Z'_p Z'_q (X_p Z'_q + X_q Z'_p) + (X_p X_q - Z'_p Z'_q)^2, \\
Z'_{p+q} &= \tfrac{x}{\sqrt{a}} (X_p Z'_q - X_q Z'_p)^2, \\
X_{2p} &= \left(X_p^2 - Z_p'^2\right)^2 - 8\tfrac{b}{a\sqrt{a}} X_p Z_p'^3, \\
Z'_{2p} &= 4Z'_p \left(X_p^3 + X_p Z_p'^2 + \tfrac{b}{a\sqrt{a}} Z_p'^3\right).
\end{aligned}$$

Of course, $\frac{b}{a\sqrt{a}}$ and $\frac{x}{\sqrt{a}}$ must be precomputed. In this case, addition can be evaluated in 9 multiplications and doubling in 8.

Notes and Comments. This approach is nothing else but a trivial change of variable on the original curve given by $x' = \frac{x}{\sqrt{a}}$ and $y' = \frac{y}{\sqrt[4]{a^3}}$. However, as the y-coordinate is not used in the Montgomery scalar multiplication, a is not required to be a fourth root. That is why we preferred to introduce Z' as a new coordinate instead of writing the change of variables.

4.2 Generalization

If a is not a square, one can hope that $\frac{a}{k}$ is a square with k small, so multiplications by a can be replaced by multiplications by k.

Theorem 1. *Let E be an elliptic curve defined over $\mathbf{K}$ by (1) and k be a small integer such that $\frac{a}{k}$ is a square in $\mathbf{K}$. Let also $P = (X_p, Y_p, Z_p)$ and $Q = (X_q, Y_q, Z_q) \in E(\mathbf{K})$ given in projective coordinates. Assume that $P - Q = (x, y)$ is known in affine coordinates. Put $Z' = \sqrt{\frac{a}{k}}Z$. Then we 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 formulae :*

$$\begin{aligned}
X_{p+q} &= -4\tfrac{b}{\frac{a}{k}\sqrt{\frac{a}{k}}} Z'_p Z'_q (X_p Z'_q + X_q Z'_p) + (X_p X_q - k Z'_p Z'_q)^2, \\
Z'_{p+q} &= \tfrac{x}{\sqrt{\frac{a}{k}}} (X_p Z'_q - X_q Z'_p), \\
X_{2p} &= \left(X_p^2 - k Z_p'^2\right)^2 - 8\tfrac{b}{\frac{a}{k}\sqrt{\frac{a}{k}}} X_p Z_p'^3, \\
Z'_{2p} &= 4Z'_p \left(X_p^3 + k X_p Z_p'^2 + \tfrac{b}{\frac{a}{k}\sqrt{\frac{a}{k}}} Z_p'^3\right).
\end{aligned}$$

These formulae also requires 9 multiplications for addition and 8 for doubling, but is it always possible to find such a k? One can hope that the probability that such a small k does not exist is very low. For instance, if $\mathbf{K}= \mathbb{F}_p$ with p prime, the multiplicativity of the Legendre symbol trivially implies that, if a is not a square, $\frac{a}{k}$ is a square if and only if k is not a square and there are only about $\frac{1}{2^n}$ prime field for which the n first prime numbers are squares. For example, only 3 percent of the prime fields are such that -1, 2, 3, 5 and 7 are squares. Hence it will really be bad luck if we cannot apply this theorem. If so, one can use the same strategy with the coefficient b.

4.3 Alternative approach

It is also possible to modify the coordinate Z to avoid the multiplications by b.

Theorem 2. *Let E be an elliptic curve defined over $\mathbf{K}$ by (1) and k a small integer such that $\frac{4b}{k}$ is a cube in $\mathbf{K}$. Let also $P = (X_p, Y_p, Z_p)$ and $Q = (X_q, Y_q, Z_q) \in E(\mathbf{K})$ given in projective coordinates. Assume that $P - Q = (x, y)$ is known in affine coordinates. Put $Z' = \sqrt[3]{\frac{4b}{k}}Z$. Then we 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 formulae :*

$$
\begin{aligned}
X_{p+q} &= -kZ'_pZ'_q(X_pZ'_q + X_qZ'_p) + (X_pX_q - \tfrac{a}{\sqrt[3]{\left(\frac{4b}{k}\right)^2}}Z'_pZ'_q)^2,\\
Z'_{p+q} &= \tfrac{x}{\sqrt[3]{\frac{4b}{k}}}(X_pZ'_q - X_qZ'_p)^2,\\
X_{2p} &= \left(X_p^2 - \tfrac{a}{\sqrt[3]{\left(\frac{4b}{k}\right)^2}}Z_p'^2\right)^2 - 2kX_pZ_p'^3,\\
Z'_{2p} &= Z'_p\left(4X_p^3 + \tfrac{4a}{\sqrt[3]{\left(\frac{4b}{k}\right)^2}}X_pZ_p'^2 + kZ_p'^3\right).
\end{aligned}
$$

The new coordinates we introduce for Montgomery scalar multiplication allow to add and double in 17 multiplications instead of 19. Moreover this improvement can be applied to most of the curves since the condition is just that there exist a small integer k such that either $\frac{a}{k}$ is a square or $\frac{4b}{k}$ is a cube in $\mathbf{K}$.

5 Montgomery scalar multiplication and the RNS

In this section we assume that the coefficient a is small, either by applying the result of the previous section or by taking $a = -3$ as it is the case for most of the standardised curves.

If the RNS system of representation is used to represent elements of the base field, the number of multiplications is not significant to estimates the complexity of the algorithm. The most expensive operation is the modular reduction. Hence the formulae for addition and doubling must be rewritten to minimize the number of modular reduction.

Theorem 3. *Let p be a prime number and E be an elliptic curve defined over $\mathbb{F}_p$ by (1). 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 we 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 formulae :*

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

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

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

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

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

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

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

Hence, only 13 modular reductions (and 18 multiplications) are required for each bit of the exponent in the Montgomery scalar multiplication algorithm, which is better than the 17 multiplications (and 14 reductions) required in standard representation.

Notes and Comments. If a is not small, one more operation is required ($a\alpha$) in each case. Moreover, we obtained no gain for curves in Montgomery form.

6 Comparisons of performance

Now, let us compare the complexity of our approach to those using Montgomery modular multiplication or Mersenne numbers. The following table is a summary of the complexities. So, one step of Montgomery exponentiation al-

Operation	RNS	Montgomery	Mersenne
Multiplication	$2n$	n^2	n^2
Reduction	$2n^2 + 6n$	$n^2 + n$	0

Table 1. Number of word operations in RNS, Montgomery and Mersenne approach for two n-word integers

gorithm using the formulae given in section 2.2 (for Montgomery and Mersenne

approach) or section 5 (for our approach) requires $17n^2 + 14(n^2 + n)$ operations with Montgomery modular multiplication, $17n^2$ with Mersenne numbers and $18(2n) + 13(2n^2 + 6n)$ in RNS. We give in table 2 the number of 32 bits word operations for one iteration of Montgomery scalar multiplication algorithm for usual ECC sizes.

$\lvert p \rvert_2$	word	RNS	Montgomery	Mersenne
160	5	1220	845	425
192	6	1620	1200	612
256	8	2576	2096	1088
320	10	3740	3240	1700
512	16	8480	8160	4352

Table 2. Cost of one iteration of Montgomery scalar multiplication

We can see that in practice our approach is slower for 5 or 6 words but competitive for larger sizes. Note that, if a 16 bits architecture is used, our method is always competitive. Moreover RNS representation can be easily parallelized.

Indeed, if we assume that we dispose of an architecture equivalent to n word-operators on a single word-bus, we get in table 3 the complexities of the different approaches in number of word operations.

Operation	RNS	Montgomery	Mersenne
Multiplication	2	$n \dots 2n$	$n \dots 2n$
Reduction	$2n + 5$	$2n \dots 3n$	0
One iteration of algorithm 2	$26n + 101$	$44n \dots 75n$	$17n \dots 34n$

Table 3. Number of cycles with parallel implementations on a n word-operators structure (18M+13R for RNS and Montgomery and 17M+14R for Mersenne).

The estimation of the cost for the multiplication and for Montgomery parallel product are based on systolic implementations [21] or on parallel implementations [7, 25] where the given architecture 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 (the hight and the low part of digits multiplication),
- a carry save adder will need a number storing the carry and a final adder for absorbing those carries,

– 32 bits words look-up tables are not reasonable.

Then, to get an idea with ECC key size, we compare three different implementations in table 4 for the number of operations given in section 5.

$\lvert p\rvert_2$	word	RNS	Montgomery	Mersenne
160	5	231	220 . . . 375	85. . . 170
192	6	257	264 . . . 450	102. . . 204
256	8	309	352. . . 600	136. . . 272
320	10	361	440 . . . 750	170. . . 340
512	16	517	704. . . 1200	272. . . 544

Table 4. Comparison of parallel implementations

In this configuration, the RNS becomes interesting in terms of efficiency for a leak-resistant implementation of elliptic curve cryptosystems. Implementations based on generalized Mersenne primes are better in term of efficiency but one has to keep in mind that an architecture using such prime numbers has some disadvantages compared to RNS. In particular, it is highly not scalable.

7 Conclusion

We combined two leak-resistance techniques to obtain an efficient and secure implementation of elliptic curves cryptosystems on embedded devices.

For this, after showing that one of the coefficient of the curve can almost always be transformed in a small number, we optimized formulae for addition and doubling in the Montgomery ladder in terms of reductions since this operation is the most costly in RNS.

Our approach is particularly interesting in the hardware point of view since the RNS representation of numbers has many advantages (leak-resistance, easy to implement and to parallelize, scalability). It becomes very attractive in the case of a dedicated parallel architecture.

References

1. 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
2. Bajard, J.C., Imbert, L.: A full RNS implementation of RSA. IEEE Transactions on Computers **53:6** (2004) 769–774
3. Bajard, J.C., Imbert, L., Liardet, P.Y., Teglia, Y.: Leak resistant arithmetic. CHES 2004, LNCS **3156** 59–65
4. Bajard, J.C., Meloni, N., Plantard, T.: Efficient RNS bases for Cryptography IMACS'05, Applied Mathematics and Simulation, (2005) ???–???

5. Bosselaers, A., Govaerts, R., Vandewalle. J.: Comparison of the three modular reduction functions LNCS **773** (1994) 175–186
6. Brier, E., Joye, M.: Weierstrass Elliptic Curves and Side-Channel Attacks. Public Key Cryptography, LNCS **2274** (2002) 335–345
7. Bunimov, V., Schimmler, M.: Efficient Parallel Multiplication Algorithm for Large Integers Euro-Par 2003, International Conference on Parallel and Distributed Computing (2003) 923–928
8. Chung, J., Hasan, A.: More generalized mersenne numbers. SAC 2003, LNCS **3006** (2003) 335–347
9. 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)
10. Cohen, H., Frey, G.: Handbook of elliptic and hyperelliptic curve cryptography. Discrete Math. Appl., Chapman & Hall/CRC (2006)
11. Coron, J.S.: Resistance against differential power analysis for elliptic curve cryptosystems. CHES'99, LNCS **1717** (1999) 292–302
12. 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
13. Izu, T., Takagi, T.: A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks. Public Key Cryptography, LNCS **2274** (2002) 280–296
14. Garner, H.L.: The residue number system. IRE Transactions on Electronic Computers, EL **8:6** (1959) 140–147
15. Knuth, D.: Seminumerical Algorithms. The Art of Computer Programming, vol. 2. Addison-Wesley (1981)
16. Kocher, P.C.: Timing attacks on implementations of DH, RSA, DSS and other systems. CRYPTO'96, LNCS **1109** (1996) 104–113
17. Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. CRYPTO'99, LNCS **1666** (1999) 388–397
18. Montgomery, P.L.: Modular multiplication without trial division. Math. Comp. **44:170** (1985) 519–521
19. Montgomery, P.L.: Speeding the Pollard and elliptic curve methods of factorization. Math. Comp. **48:177** (1987) 243–164
20. 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
21. G. Orlando and C. Paar. A scalable GF(p) elliptic curve processor architecture for programmable hardware. In Proceedings of Workshop on Cryptograpic Hardware and Embedded Systems (CHES 2001)
22. Posch, K.C., Posch, R.: Modulo reduction in residue number systems. IEEE Transaction on Parallel and Distributed Systems **6:5** (1995) 449–454
23. Quisquater, J.J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards. e-smart 2001, LNCS **2140** (2001) 200–210
24. Shenoy, A.P., Kumaresan, R.: Fast base extension using a redundant modulus in RNS. IEEE Transactions on Computer **38:2** (1989) 292–296
25. 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
26. Solinas, J.: Generalized mersenne numbers. Research Report CORR-99-39, Center for Applied Cryptographic Research, University of Waterloo (1999)

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