RNS arithmetic in $\mathbb{F}_{p^k}$ and application to fast pairing computation

Sylvain Duquesne

IRMAR, UMR CNRS 6625
Université Rennes 1
Campus de Beaulieu
35042 Rennes cedex
France
sylvain.duquesne@univ-rennes1.fr

October 30, 2010

Abstract

In this work, we are interested in arithmetic in large prime field and their extensions of small degree. We explain why it is very interesting to use RNS arithmetic for the base field $\mathbb{F}_p$ when computations in $\mathbb{F}_{p^k}$ have to be done, essentially thanks to lazy reduction. This is for example the case for pairing computations on ordinary curves (as MNT or BN curves). We prove that using RNS can considerably decrease the number of basic operations required for a pairing computation in many popular situations.

Contents

1 Introduction

In recent years, pairing based cryptography became more and more popular. Thus efficient software and hardware implementation are necessary. For some time past, ordinary curves superseded supersingular curves [13, 43] on large prime fields. Such pairing involve many arithmetic in extensions of small degrees of the base field. Hence, one approach angle for efficient pairing computation is the optimization of extension field arithmetic. The Residue Number System (RNS) has already been introduced in elliptic curve cryptography in [4, 3]. This is a system to represent numbers which uses the Chinese Remainder Theorem. It is recalled in Section 2. Contrary to standard arithmetic in $\mathbb{F}_p$, the RNS introduces a gap of complexity between the multiplication and the reduction step of a modular multiplication. In extension fields, we show in Section 3 how lazy reduction allow to considerably decrease the number of reduction. Hence the RNS is particularly well suited for extension field arithmetic and

consequently for pairing computations. In Section 4, after some background on pairings and their computation, we study in details the complexity of an RNS implementation of current pairings (Tate, Ate, R-Ate) and compare it to standard complexities. We will concentrate on MNT and BN curves which are the most popular today for embedding degrees 6 and 12 since their cardinality can be prime or almost prime contrary to other constructions in the literature. In all cases we obtain significant savings so that the RNS is very interesting for future pairing implementations.

This work was supported by the ANR projects no. 07-BLAN-0248 ALGOL and 09-BLAN-0020-01 CHIC

2 Efficient arithmetic on prime fields

2.1 Modular multiplication

The elliptic curve arithmetic over $\mathbb{F}_p$ mainly involves modular multiplications modulo p. Such a modular multiplication can be decomposed into one classic multiplication followed by one modular reduction. Because of the small size of the numbers used in elliptic curve cryptography (192 to 512 bits, i.e. 6 to 16 32-bit words), the multiplication is performed by a common method. Let us consider a and b, as two n-word integers given in radix representation (i.e., $x = \sum_{i=0}^{n} x_i \mathbf{B}^i$ with $0 \leq x_i < \mathbf{B}$). Then ab 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 this by the equation

$$ab = b_0 a + \mathbf{B}(b_1 a + \mathbf{B}(b_2 a \cdots + \mathbf{B} b_n a) \dots).$$

The complexity is then n^2 word multiplications. We note that for the current ECC key size, Karatsuba or Toom-Cook approaches are not competitive as discussed in the study made by the GMP group [30].

The reduction of an integer modulo p consists of finding the remainder of the Euclidean division of this integer by p. This operation is costly. It can be substantially sped up by using the Montgomery reduction or by using a special modulo.

2.1.1 Montgomery general reduction algorithm

In [42], Montgomery proposed to substitute the reduction modulo p by a division by a power of the radix $\mathbf{B}$ (which is a simple shift). The result is not exactly $a \mod p$ but $a\mathbf{B}^{-n} \mod p$. This problem can be overcome by using Montgomery representation where $a' = a \times \mathbf{B}^n \mod p$.

Algorithm 1: Montgomery$_p(c)$

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

The complexity of this reduction is $n^2 + n$ word operations [17]. For $a < \mathbf{B}^n$, its Montgomery representation is obtained via Algorithm 1 with $c = a \times (\mathbf{B}^n \bmod p)$. By the same way, if a' is the Montgomery representation of a, then we recover a using Algorithm 1 with $c = a'$. Of course, such conversion is done only once at the beginning and once at the end of the complete cryptographic computing so that all the computations can be done in Montgomery representations. Hence, we ignore the cost of the conversion from Montgomery to classic representation (and reciprocally) in the following. We note, as $r < 2p$, that 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 $4p < \mathbf{B}^n$.

2.1.2 Reduction using special modulo

Usually, when using ECC, one can choose the underlying field without restrictions. In this case, the cost of a modular reduction can be reduced to several shifts and additions. This is why the generalized Mersenne number class was introduced [49, 19]. This is used in most of the standards. However this approach has several drawback

- It requires a dedicated architecture to such a particular p 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.
- In pairing based cryptosystems based on ordinary curves, the underlying fields cannot be chosen because curves are build via complex multiplication methods. Moreover, it has been shown in [44] that the use of such special moduli can introduce weakness in pairing based cryptosystems.

For these reasons we do not consider this approach in this paper.

2.2 The Residue Number Systems (RNS)

The Residue Number Systems are a corollary of 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 $a_i = |a|_{m_i} = a \bmod m_i$. We generally assume that $0 \leq a < M = \prod_{i=1}^{n} m_i$. The elements a_i are called RNS-digits. The strongest advantage of a such system is that it distributes large integer operations on the small residue values. The operations are performed independently on the

residues. In particular, there is no carry propagation. These systems were introduced and developed in [28, 50, 51]. A good introduction can be found in [37].

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. 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, 8]).

$$r = (|a_1 \times b_1|_{m_1}, \ldots, |a_n \times b_n|_{m_n}) \equiv a \times b \pmod{M} \tag{1}$$

In this paper, we consider RNS basis $(m_1, ..., m_n)$ with elements such that, $m_i = 2^h - c_i$, where c_i is small and sparse, $c_i < 2^{h/2}$. The reduction modulo m_i is, in this case, obtained with few shift and additions as in 2.1.2. As explained in [3] this property ensures that a RNS digit-product can be considered to be equivalent to 1.1 word-product (word = h-bits).

We now focus on the multiplication modulo p using the Montgomery algorithm presented in [1, 2]. For two numbers a and b given in RNS, this algorithm evaluates $r = abM^{-1} \mod p$ in RNS. As in the classical Montgomery method given in subsection 2.1.1, this problem can be overcome by using Montgomery representation where $a' = a \times M \mod p$ which is stable for Montgomery product and addition. Of course, the conversion is done only once at the beginning by performing Montgomery product with a and $(M^2 \bmod p)$ 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 [6].

2.2.1 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 $\mathbf{B}$ number system, the value $\mathbf{B}^n$ occurs in the reduction, division and Montgomery factor. In RNS, this value is replaced by M. However, an auxiliary RNS basis is needed to handle the inverse of M. Hence some operations as the initial product must be performed on the two bases, which cost $2n$ words-products.

Algorithm 2 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 clarify on which basis they are done.

Algorithm 2: 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 $\gcd(M, M') = 1$;
A prime number p such that $4p < M$ and $\gcd(p, M) = 1$. p is represented in basis $\mathcal{B}'$ and $-p^{-1}$ is precomputed in basis $\mathcal{B}$;
A positive integer c represented in RNS in both bases, with $c < Mp$.
Result: A positive integer $r \equiv cM^{-1} \pmod{p}$ represented in RNS in both bases, with $r < 2p$.
begin

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*;

end

Instructions 1 and 3 of Algorithm 2 are RNS additions or multiplications which are performed independently for each element of the basis, so they are very efficient (linear). Instructions 2 and 4 represent RNS base extensions which are quadratic and then costly. To reduce this cost, we can use two different full RNS extensions as shown in [1, 2].

Finally, it is shown in [3] that the overall complexity of algorithm 2 is $\frac{7}{5}n^2 + \frac{8}{5}n$ RNS digit-products.

If we operate with an architecture of n basic word-arithmetic cells, RNS arithmetic can be easily performed in a parallel manner due to the independence of the RNS digit operations. A parallel evaluation of the multiplication (in both bases) requires only 2 steps whereas Algorithm 2 can be done in $\frac{12}{5}n + \frac{3}{5}$ steps [3].

2.2.2 Advantages of the RNS

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.

- If we assume that for ECC size the multiplication needs n^2 word-products, the RNS approach is asymptotically quite interesting for a modular multiplication which represents $2n^2 + n$ word-products in classical systems and $\left(\frac{7}{5}n^2 + \frac{18}{5}n\right) \times 1.1$ in RNS.
- As shown in [5], RNS is easy to implement, particularly in hardware, and it provides a reduced cost for multiplication and addition and a competitive modular reduction. Furthermore, due to the independence of the modular operations, computations can be performed in a random way and the architecture can be parallelized.
- A RNS based architecture is flexible : with a given structure of n modular digit operators, it is possible to handle any values of p such that $4 \times p < M$.

Hence, the same architecture can be used for different levels of security and several base field for each of these levels.

- There is a gap between the cost of the reduction and the cost of the multiplication which does not occur in classical systems. We can take a great advantage of this gap by accumulating multiplications before reduction. This method is called lazy reduction.

2.3 Lazy reduction

Lazy reduction is often used in optimized implementations. It consists in delaying the reduction step after computing several products which must be summed. For example, assume that we want to compute $ab + cd$ with a, b, c and d in $\mathbb{F}_p$ where p is a n-word prime number. A classical implementation involves 2 modular multiplications and then requires $4n^2+2n$ word-products. In a lazy reduction implementation, we first compute the two multiplications and add them before a unique reduction step. Thus it requires only $3n^2 + n$ word-products. Of course, this implies that the reduction algorithm can take larger integers as input (less than $2p^2$ instead of less than p^2 in the above example) but this is not really cumbersome.

This method is particularly interesting if an RNS arithmetic is used because of the gap of complexity between the multiplication and the reduction step. As an example, while the classical computation of $ab+cd$ requires $\frac{14}{5}n^2 + \frac{36}{5}n$ RNS digit-products, the use of lazy reduction requires only $\frac{7}{5}n^2 + \frac{28}{5}n$ RNS digit-products. Hence, lazy reduction is particularly well adapted to RNS arithmetic. This has already been used in [4] and [3] for elliptic curve cryptography. The goal of this paper is to use it for efficient arithmetic on extension fields and consequently to pairing based cryptography.

3 Fast arithmetic in $\mathbb{F}_{p^k}$ combining lazy reduction and RNS

3.1 Polynomial reduction

Efficient arithmetic in finite extensions of prime fields are usually done with sparse polynomials with small coefficients so that the cost of the reduction modulo this polynomial is given by some additions. More precisely, we give this cost when the extension is defined by a trinomial, which is almost always the case in practice.

Proposition 1. *Let p be a prime number and $X^k - \delta X^d - \varepsilon$ in $\mathbb{F}_p[X]$ such that $d \leq k/2$ and δ, ε small (by small, we mean that the multiplication by such a number is cheap in $\mathbb{F}_p$). The reduction modulo $X^k - \delta X^d - \varepsilon$ of a degree $2k-1$ polynomial requires only few additions in $\mathbb{F}_p[X]$.*

Proof. Write a degree $2k-1$ polynomial P as $X^{2k-d}P_1 + X^k P_2 + P_3$ where P_1, P_2 and P_3 are polynomials of degree respectively at most $d-1$, $k-d-1$ and $k-1$.

$$\begin{aligned}
P &= (\delta X^d + \varepsilon)X^{k-d}P_1 + (\delta X^d + \varepsilon)P_2 + P_3 && \bmod X^k - \delta X^d - \varepsilon \\
&= \delta X^k P_1 + \varepsilon X^{k-d}P_1 + \delta X^d P_2 + \varepsilon P_2 + P_3 && \bmod X^k - \delta X^d - \varepsilon \\
&= \delta(\delta X^d + \varepsilon)P_1 + \varepsilon X^{k-d}P_1 + \delta X^d P_2 + \varepsilon P_2 + P_3 && \bmod X^k - \delta X^d - \varepsilon \\
&= \delta^2 X^d P_1 + \delta\varepsilon P_1 + \varepsilon X^{k-d}P_1 + \delta X^d P_2 + \varepsilon P_2 + P_3 && \bmod X^k - \delta X^d - \varepsilon \\
&= \delta X^d(\delta P_1 + P_2) + \varepsilon(\delta P_1 + P_2 + X^{k-d}P_1) + P_3 && \bmod X^k - \delta X^d - \varepsilon
\end{aligned}$$

It is easy to verify that all polynomials involved in this sum have degree less than or equal to $k-1$, so that this last expression is the reduced form of P. This sum requires only

- 4 additions in $\mathbb{F}_p[X]$,
- 2 multiplications by monomials in $\mathbb{F}_p[X]$ which are nothing but shifts of the coefficients and then are almost free,
- 3 multiplications by δ or ε which can be counted as few additions in $\mathbb{F}_p[X]$ since δ and ε are assumed to be small.

□

This means that, if the irreducible polynomial defining $\mathbb{F}_{p^k}$ is well chosen, the cost of the reduction step in $\mathbb{F}_{p^k}$ arithmetic is negligible compared to a multiplication in $\mathbb{F}_p[X]$. In all the cases we are interested in this paper, and more generally in cryptography, such cheap reduction always holds. Hence we will focus in this paper on multiplication in $\mathbb{F}_p[X]$.

3.2 Multiplication in $\mathbb{F}_{p^k}$

These multiplications can be done using Schoolbook method or using alternative well-known method like Karatsuba or Toom-Cook. In this paper, we are interested in small values of k so that methods based on FFT are not interesting. Several people already studied in details which method must be used for each value of k and each construction of the extension. For example, [21] is very complete for extensions used in pairing-based cryptography. We will not recall these results here but use them for our comparisons. Anyway, whatever the method used, it requires k^λ multiplications in $\mathbb{F}_p$, with $1 < \lambda \leq 2$. The use of lazy reduction in this case is immediate. We just have to delay the reduction steps at the end of the computation. Then, only k reductions in $\mathbb{F}_p$ are required.

Example with $k = 2$

Assume $p \equiv 3$ modulo 4 so that -1 is not a square in $\mathbb{F}_p$. Then $\mathbb{F}_{p^2}$ can be defined by $\mathbb{F}_p[X]/(X^2+1)$. We want to compute the product of $P = a_0 + a_1 X$ and $Q = b_0 + b_1 X$. Using schoolbook multiplication, we have

$$PQ = a_0 b_0 - a_1 b_1 + (a_0 b_1 + a_1 b_0)X.$$

This is the typical case where lazy reduction is interesting since $ab + cd$ patterns occur. Finally, such a multiplication in $\mathbb{F}_{p^2}$ involves 4 multiplications in $\mathbb{F}_p$ but only 2 modular reductions. Note that, as elements in $\mathbb{F}_{p^2}$ have 2 independent components, it is not possible to have less than 2 reductions in $\mathbb{F}_p$ in the general case. Thus, using Karatsuba multiplication allow to perform only 3 multiplications in $\mathbb{F}_p$ but always 2 reductions thanks to the formula

$$PQ = a_0b_0 - a_1b_1 + ((a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1b_1)X.$$

The interest of using the RNS for the arithmetic in $\mathbb{F}_{p^k}$ then becomes evident. Indeed, the expensive step of the RNS, namely the reduction step, is used linearly when $\mathbb{F}_{p^k}$ arithmetic is performed whereas, the cheaper step, namely the multiplication step, is used quadratically or sub-quadratically. More precisely, we have the following property

Proposition 2. *Let p be a prime number which can be represented by n words in radix representation and n RNS-digit in RNS representation. Let $\mathbb{F}_{p^k}$ be a finite extension of $\mathbb{F}_p$ defined by a sparse polynomial with small coefficients. We assume that the multiplication in $\mathbb{F}_p[X]$ requires k^λ multiplications in $\mathbb{F}_p$, with $1 < \lambda \leq 2$ and that we use lazy reduction in $\mathbb{F}_p$. A multiplication in $\mathbb{F}_{p^k}$ then requires*

- $(k^\lambda + k)n^2 + kn$ *word multiplications in radix representation,*
- $1.1\times\left(\frac{7k}{5}n^2 + \frac{10k^\lambda+8k}{5}n\right)$ *word multiplications if RNS is used.*

Proof. A complete multiplication in $\mathbb{F}_{p^k}$ requires k^λ multiplications in $\mathbb{F}_p$ thanks to Karatsuba-like methods and k reductions in $\mathbb{F}_p$ thanks to the use of the lazy reduction method. We have seen in subsection 2.1 that a multiplication in $\mathbb{F}_p$ requires n^2 word multiplications and that a reduction requires $n^2 + n$ of them. This trivially gives the first assertion. The second one is obtained thanks to the cost of RNS multiplication ($2n$ RNS digit-products) and reduction ($\frac{7}{5}n^2 + \frac{8}{5}n$ RNS digit-products) given in subsection 2.2 which must be multiplied by 1.1 to have an equivalent in word multiplications. □

Most of the gain is due to the accumulation of many products before reducing and not only 2 as in [4] or [3]. Of course, both the classical and the RNS reduction algorithms must be adapted. Indeed, input data can have a large size compared to p because of this accumulation process. More precisely, input data have maximal size $k'p^2$ where k' has the same size than k (it is not equal to k only because of the polynomial reduction step). Then it is sufficient to choose the radix such that $\mathbf{B}^n > k'p$ (or the RNS basis such that $M > k'p$). Moreover, if we want to use the output of the reduction algorithm (which is in $[0, 2p[$) as an input without a final comparison and subtraction, each product becomes less than $4p$ so that we have to choose $\mathbf{B}^n > 4k'p$ (or $M > 4k'p$). This is not restrictive in practice as long as k is not too large.

For values of k and n greater than or equal to 6, the gain is spectacular. For instance if $n = k = 6$ and $\lambda = 1.5$ (which is a mean between Karatsuba and

Toom-Cook complexities), a multiplication in $\mathbb{F}_{p^6}$ requires 781 word multiplications in radix representation while it requires only 590 in RNS. Of course this is just a rough estimation to give an idea of the expected gain. Each particular situation must be studied in details. In this paper, we do it for the extension degrees 6 and 12 for two reasons :

- there are of particular interest in pairing-based cryptography as we will see at the end of this paper,
- the classical arithmetic on such extensions is well studied ([21, 22]) which facilitates comparisons.

Some other extension degrees, like 2, 8 or 10, also have an interest in pairing-based cryptography but do not involve new materials relatively to 6 and 12. They can be done by the interested reader or asked to the author.

3.3 Lazy arithmetic in $\mathbb{F}_{p^6}$

In this section, we recall the different ways to perform efficient arithmetic in $\mathbb{F}_{p^6}$ and combine them with lazy reduction and of course RNS arithmetic. In fact, we are especially interested in multiplication in $\mathbb{F}_{p^6}$. There are three different ways to build $\mathbb{F}_{p^6}$, namely as a quadratic extension of a cubic one, as a cubic extension of a quadratic one or directly as a sextic extension (i.e. with an irreducible polynomial of degree 6). For each of these constructions Devegilli et al [21] studied in details all the possible arithmetic (Schoolbook, Karatsuba, Toom-Cook and Chung-Hasan for squaring). There conclusions are

- Toom-Cook method requires asymptotically less multiplications or squarings in $\mathbb{F}_p$ but is inefficient in practice for cryptographic sizes because it requires many additions in $\mathbb{F}_p$.
- The most efficient implementation for multiplication and squaring is obtained when $\mathbb{F}_{p^6}$ is build as a quadratic extension of a cubic one. This is due to the fact that there are very efficient formulas for quadratic and cubic extensions, but not for higher degrees.

We then assume in the following that $\mathbb{F}_{p^6}$ is build as a quadratic extension of a cubic one. Of course, those extensions are build as in Proposition 2 so that the reduction step in $\mathbb{F}_{p^6}$ is negligible compared to the multiplication step. In this case, according [21], the most efficient way to perform a multiplication is to use Karatsuba method for both the multiplication in the quadratic extension and in the cubic extension. The cost of a multiplication in $\mathbb{F}_{p^6}$ is then 18 multiplications in $\mathbb{F}_p$. Concerning squaring, there are several method more or less equivalent in [21]. The choice is depending on the cost of the squaring in $\mathbb{F}_p$ compared to a multiplication. In this paper, we assume that these costs are the same so that the best squaring in $\mathbb{F}_{p^6}$ requires 12 multiplications in $\mathbb{F}_p$ using Karatsuba squaring for the cubic extension and the complex method for the quadratic one.

Performing lazy reduction is immediate assuming that it is possible to reduce sums of several products. To ensure this, we have to relax the condition

on $\mathbf{B}^n > 4p$ (for Montgomery reduction) or $M > 4p$ (for RNS arithmetic). This condition can become several hundreds times larger than p depending on the polynomial representing $\mathbb{F}_{p^6}$. However this is not restrictive for cryptographic implementation. Indeed field sizes are usually a multiple of 32 so that an additional word is necessary to handle $\mathbf{B}^n$ or M if lazy reduction is used. Even on a 16 bit architecture, this additional word can easily handle integers of size several hundreds. If larger words are used like 36 bits words on FPGA, there is also sufficiently extra bits to handle $\mathbf{B}^n$ or M for cryptographic applications [5].

Finally the cost of a multiplication in $\mathbb{F}_{p^6}$ is 18 multiplication in $\mathbb{F}_p$ and 6 modular reductions whereas a squaring requires only 12 multiplications but also 6 reductions.

3.4 Example of degree 6 extension in 192 bits

In this section, we give an explicit example of degree 6 extension of a 192 bits prime field. This example comes from [43] and is linked to a MNT curve suitable for pairing based cryptography, which is the subject of the next section. Let $\mathbb{F}_p$ be defined by the prime number

$$p = 4691249309589066676602717919800805068538803592363589996389.$$

In this case $\mathbb{F}_{p^6}$ can be defined by a quadratic extension of a cubic one thanks to the polynomials $X^3 - 2$ and $Y^2 - \boldsymbol{\alpha}$ where $\boldsymbol{\alpha}$ is a cubic root of 2.

$$\mathbb{F}_{p^3} = \mathbb{F}_p[X]/(X^3 - 2) = \mathbb{F}_p[\boldsymbol{\alpha}] \text{ and}$$

$$\mathbb{F}_{p^6} = \mathbb{F}_{p^3}[Y]/(Y^2 - \boldsymbol{\alpha}) = \mathbb{F}_{p^3}[\boldsymbol{\beta}].$$

As we want to use lazy reduction, the arithmetic of this extension must be completely unrolled. Hence let

$$\begin{aligned} A &= a_0+a_1\boldsymbol{\alpha}+a_2\boldsymbol{\alpha}^2+\left(a_3+a_4\boldsymbol{\alpha}+a_5\boldsymbol{\alpha}^2\right)\boldsymbol{\beta} \text{ and} \\ B &= b_0+b_1\boldsymbol{\alpha}+b_2\boldsymbol{\alpha}^2+\left(b_3+b_4\boldsymbol{\alpha}+b_5\boldsymbol{\alpha}^2\right)\boldsymbol{\beta} \end{aligned}$$

be two elements of $\mathbb{F}_{p^6}$. Using Karatsuba on the quadratic extension leads to

$$\begin{aligned} AB = {} & \left(a_0+a_1\boldsymbol{\alpha}+a_2\boldsymbol{\alpha}^2\right)\left(b_0+b_1\boldsymbol{\alpha}+b_2\boldsymbol{\alpha}^2\right)+\boldsymbol{\alpha}\left(a_3+a_4\boldsymbol{\alpha}+a_5\boldsymbol{\alpha}^2\right)\left(b_3+b_4\boldsymbol{\alpha}+b_5\boldsymbol{\alpha}^2\right)+ \\ & \left[\left(a_0+a_3+(a_1+a_4)\boldsymbol{\alpha}+(a_2+a_5)\boldsymbol{\alpha}^2\right)\left(b_0+b_3+(b_1+b_4)\boldsymbol{\alpha}+(b_2+b_5)\boldsymbol{\alpha}^2\right)\right. \\ & \left.-\left(a_0+a_1\boldsymbol{\alpha}+a_2\boldsymbol{\alpha}^2\right)\left(b_0+b_1\boldsymbol{\alpha}+b_2\boldsymbol{\alpha}^2\right)-\left(a_3+a_4\boldsymbol{\alpha}+a_5\boldsymbol{\alpha}^2\right)\left(b_3+b_4\boldsymbol{\alpha}+b_5\boldsymbol{\alpha}^2\right)\right]\boldsymbol{\beta} \end{aligned}$$

Using Karatsuba again to compute each of these 3 products leads to

$$\begin{aligned}
AB &= a_0b_0+2(a_4b_4+(a_1+a_2)(b_1+b_2)-a_1b_1+(a_3+a_5)(b_3+b_5)-a_3b_3-a_5b_5)\\
&+ [a_3b_3+(a_0+a_1)(b_0+b_1)-a_0b_0-a_1b_1+2(a_2b_2+(a_4+a_5)(b_4+b_5)-a_4b_4-a_5b_5)]\boldsymbol{\alpha}\\
&+ [a_1b_1+2a_5b_5+(a_0+a_2)(b_0+b_2)-a_0b_0-a_2b_2+(a_3+a_4)(b_3+b_4)-a_3b_3-a_4b_4]\boldsymbol{\alpha}^2\\
&+ [(a_0+a_3)(b_0+b_3)-a_0b_0-a_3b_3+2((a_1+a_2+a_4+a_5)(b_1+b_2+b_4+b_5)-(a_1+a_4)(b_1+b_4)\\
&\quad -(a_2+a_5)(b_2+b_5)-(a_1+a_2)(b_1+b_2)+a_1b_1+a_2b_2-(a_4+a_5)(b_4+b_5)+a_4b_4+a_5b_5)]\boldsymbol{\beta}\\
&+ [(a_0+a_1+a_3+a_4)(b_0+b_1+b_3+b_4)-(a_0+a_3)(b_0+b_3)-(a_1+a_4)(b_1+b_4)-(a_0+a_1)(b_0+b_1)\\
&\quad +a_0b_0+a_1b_1-(a_3+a_4)(b_3+b_4)+a_3b_3+a_4b_4+2((a_2+a_5)(b_2+b_5)-a_2b_2-a_5b_5)]\boldsymbol{\alpha\beta}\\
&+ [(a_1+a_4)(b_1+b_4)-a_1b_1-a_4b_4+(a_0+a_2+a_3+a_5)(b_0+b_2+b_3+b_5)-(a_0+a_3)(b_0+b_3)\\
&\quad -(a_2+a_5)(b_2+b_5)-(a_0+a_2)(b_0+b_2)+a_0b_0+a_2b_2-(a_3+a_5)(b_3+b_5)+a_3b_3+a_5b_5]\boldsymbol{\alpha}^2\boldsymbol{\beta}
\end{aligned}$$

It is easy to verify that this formula requires 18 multiplications in $\mathbb{F}_p$. Of course it also requires many additions but this is due to the Karatsuba method, not to lazy reduction. As explained in subsection 3.3, it requires only 6 reductions thanks to the accumulation of all the operations in each component. However, this accumulation implies that the input of the reduction step can be very large. More precisely, thanks to the existence of the schoolbook method for computing AB, we can easily prove that if the components of A and B (ie the a_i and the b_i) are between 0 and $2p$ (which is the case when algorithm 1 or 2 is used for reduction) then each component of AB is between 0 and $44p^2$. This means that $\mathbf{B}^n$ in Montgomery representation and M in RNS representation must be greater than $44p$ to perform lazy reduction in this degree 6 field.

3.5 Lazy arithmetic in $\mathbb{F}_{p^{12}}$

The same work as in the section 3.3 can be done in the case of $\mathbb{F}_{p^{12}}$. This has already been done in the recent literature because of the success of Barreto-Naehrig curves. For example Devegili, Scott and Dahab explain in [22] that $\mathbb{F}_{p^{12}}$ must be build as a tower of extensions: quadratic on top of a cubic on top of a quadratic. This is nothing but the section 3.3 applied to a quadratic extension of $\mathbb{F}_p$. Then, if Karatsuba method is used for the multiplication in this quadratic extension, a multiplication in $\mathbb{F}_{p^{12}}$ requires 54 multiplications in $\mathbb{F}_p$ and 12 modular reductions and a squaring requires 36 multiplications in $\mathbb{F}_p$ and 12 modular reductions.

3.6 Other useful operations in $\mathbb{F}_{p^k}$

Three other operations in $\mathbb{F}_{p^k}$ are necessary for pairing computation, the inversion, the Frobenius action (ie powering to the p) and the squaring of unit elements in quadratic extensions.

Performing an inversion in $\mathbb{F}_{p^k}$ must be done very carefully because it is an expensive operation. The general idea is that the inverse of an element is the product of its conjugates divided by its norm. This allows to replace an inversion in $\mathbb{F}_{p^k}$ by an inversion in $\mathbb{F}_p$ and some multiplications. For example

in $\mathbb{F}_{p^2} = \mathbb{F}_p[X]/X^2 - \varepsilon$, we have

$$\frac{1}{a_0 + a_1\sqrt{\varepsilon}} = \frac{a_0 - a_1\sqrt{\varepsilon}}{a_0^2 - \varepsilon a_1^2}$$

and an inversion in $\mathbb{F}_{p^2}$ requires 1 inversion, 2 squarings, 2 multiplications and 3 reductions in $\mathbb{F}_p$. In the same way an inversion in $\mathbb{F}_{p^3}$ (defined by a cubic root) requires 1 inversion, 9 multiplications, 3 squarings and 7 reductions in $\mathbb{F}_p$ [46]. We can easily deduce that an inversion in $\mathbb{F}_{p^6}$, build as a quadratic extension of a cubic one, requires 1 inversion, 36 multiplications and 16 reductions in $\mathbb{F}_p$. In the same way, if $\mathbb{F}_{p^2}$ is build as in section 3.5, it is easy to show that an inversion requires 1 inversion, 97 multiplications ([31]) and 35 reductions in $\mathbb{F}_p$.

Contrary to the inversion, the Frobenius action in $\mathbb{F}_{p^k}$ is cheap. Indeed, it is easy to prove that if $\{\xi_i\}_{i=0..k-1}$ (with $\xi_0 = 1$) is a basis for $\mathbb{F}_{p^k}$ as a $\mathbb{F}_p$ vector space, we have

$$\left(\sum_{i=0}^{k-1} a_i\xi_i\right)^p = a_0 + \sum_{i=1}^{k-1} a_i\xi_i^p.$$

Thus, if the ξ_i^p are precomputed this Frobenius operation requires only $k(k-1)$ multiplications in $\mathbb{F}_p$ and $k-1$ reductions. In fact, we can do even better with a good choice of the basis. For example if the extension is defined by a root $\boldsymbol{\gamma}$ of the polynomial $X^k - \varepsilon$ then $\xi_i = \boldsymbol{\gamma}^i$ and for $1 \leq i \leq k-1$,

$$\xi_i^p = \boldsymbol{\gamma}^{ip} = c_i\boldsymbol{\gamma}^{r_i} = c_i\xi_{r_i}$$

with $c_i \in \mathbb{F}_p$ and $0 \leq r_i < k$. In this case computing the Frobenius require only $k-1$ multiplications and $k-1$ reductions in $\mathbb{F}_p$. This is the case for the example given in 3.4 where additionally $p \equiv 1 \bmod 6$ so that $r_i = i$. Of course, the same holds also for raising up an element to any power of p.

Finally, squaring in a quadratic extension requires 2 multiplications (and 2 reductions) in the base field thanks to the complex method. However, as noticed in [29, 31], if the element to be squared is a unit, this can be done with 2 squarings in the base field (and always 2 reductions) assuming the extension is defined by a polynomial of the form $X^2 - \varepsilon$. Indeed, if $a_0 + a_1X$ is such an element (which means that $a_0^2 - \varepsilon a_1^2 = 1$), we have

$$\begin{aligned}(a_0 + a_1X)^2 &= a_0^2 + \varepsilon a_1^2 + \left[(a_0 + a_1)^2 - a_0^2 - a_1^2\right] X \\ &= 1 + 2\varepsilon a_1^2 + \left[(a_0 + a_1)^2 - 1 - \varepsilon a_1^2 - a_1^2\right] X\end{aligned}$$

4 Pairing on elliptic curves and their computation

4.1 Pairings in cryptography

Bilinear pairings on elliptic curves have been introduced in cryptography in the middle of 90's for cryptanalysis. Indeed, they allow to transfer the discrete

logarithm on an elliptic curve to a discrete logarithm in the multiplicative group of a finite field, where subexponential algorithms are available [25, 40]. In 2000, Joux introduces the first constructive use of pairings with a tripartite key exchange protocol [34]. Since, it has been shown that pairings can be used to construct new protocols like identity based cryptography [15] or short signature [16]. As a consequence, pairings become very popular in asymmetric cryptography and computing them as fast as possible is very important. There are essentially three ways to speed up the computation of the pairings. Let us first briefly recall the state of the art in this field and then explain how a RNS arithmetic can be helpful.

4.2 The Tate pairing

The most popular pairing used in cryptography is the Tate pairing. We present it here in a simplified and reduced form because it is the one usually used in cryptographic applications. More details and generalities can be found in [20] or [14]. In this paper we assume that E is an elliptic curve defined on $\mathbb{F}_p$ by an equation

$$y^2 = x^3 + a_4 x + a_6. \tag{2}$$

Let ℓ be a prime divisor of $\#E(\mathbb{F}_p) = p + 1 - t$, where t is the trace of the Frobenius map on the curve. The embedding degree k of E with respect to ℓ is the smallest integer such that ℓ divides $p^k - 1$. This means that the full ℓ-torsion of the curve is defined on the field $\mathbb{F}_{p^k}$. For any integer m and ℓ-torsion point P, if P_m is the point mP on E, $f_{(m,P)}$ is the function defined on the curve whose divisor is $\mathrm{div}(f_{(m,P)}) = m\,P - P_m - (m-1)\mathcal{O}$. The Tate pairing can then be defined by

$$\begin{aligned} e_T : E(\mathbb{F}_p)[\ell] \times E(\mathbb{F}_{p^k}) &\rightarrow \mathbb{F}_{p^k}^* / \left(\mathbb{F}_{p^k}^*\right)^\ell \\ (P, Q) &\mapsto f_{(\ell,P)}(Q)^{\frac{p^k-1}{\ell}} \end{aligned}$$

The first step to compute the Tate pairing is the computation of $f_{(\ell,P)}(Q)$. It is done thanks to an adaptation of classical scalar multiplication algorithm due to Miller [41] which is given here in a more general case to cover other pairings. In this algorithm $g_{(A,B)}$ is the equation of the line passing through the points A and B (or tangent to A if $A = B$) and v_A is the equation of the vertical line passing by A, so that $\frac{g_{(A,B)}}{v_{A+B}}$ is the function on E involved in the addition of A and B.

The second step is to raise f to the power $\frac{p^k-1}{\ell}$. There are several ways to speed up the pairing computation :

- simplifying and optimizing the operations inside the Miller loop [9, 11, 12, 45],
- constructing pairing-friendly elliptic curves [10, 13, 18, 23, 26, 35, 39, 47] and a good survey is [24],

Algorithm 3: Miller(m, P, Q)

Data: An integer m with binary representation $(m_{s-1}, \cdots, m_0)_2$, two points P and Q in $E(\mathbb{F}_{p^k})$.
Result: $f_{(m,P)}(Q) \in \mathbb{F}_{p^k}$.
begin
  $T \leftarrow P$
  $f \leftarrow 1$
  for *i from* $s-2$ *downto 0* **do**
    $f \leftarrow f^2 . \frac{g_{(T,T)}(Q)}{v_{2T}(Q)}$
    $T \leftarrow 2T$
    if $m_i = 1$ **then**
      $f \leftarrow f . \frac{g_{(T,P)}(Q)}{v_{T+P}(Q)}$
      $T \leftarrow T + P$
    endif
  endfor
  return f
end

- more recently, reducing the length of the Miller loop [32, 33, 38, 52] thanks to the introduction of new pairings,
- simplifying the final exponentiation [27, 48].

Note that it is not interesting to use better exponentiation techniques as sliding windows for pairing computations. This is because even if $3P$, for instance, can be precomputed, the writing up of f requires an additional (expensive) multiplication in $\mathbb{F}_{p^k}$ by the function involved in the computation of $3P$.

4.3 Ordinary curves with prescribed embedding degrees

The embedding degree k is usually very large so that computing in $\mathbb{F}_{p^k}$ is not reasonable. This is reassuring regarding the destructive use of pairings but annoying if one wants to use pairing based cryptosystems. Curves with small embedding degrees can be obtain in two different ways. The first one is to use supersingular curves. However, this work focuses on large characteristic base fields and, in this case, the embedding degree is less than or equal to 2 which is too small for security reasons. The second one is to use ordinary curves with prescribed embedding degrees constructed via the complex multiplication method as surveyed in [24]. We will focus here on the most popular ones, namely the MNT curves having an embedding degree equal to 6 [39] and the BN curves having embedding degree equal to 12 [13].

4.3.1 MNT curves

In [39], the authors explain how to use the complex multiplication method to construct ordinary elliptic curves with embedding degrees 3, 4 and 6. Their goal was to characterize elliptic curves with small embedding degrees to protect against destructive use of pairings but it has also been used as a mean to construct ordinary curves with embedding degree 6.

Theorem 1. *Let p be a large prime and E be an ordinary elliptic curve defined over $\mathbb{F}_p$ such that $\#E(\mathbb{F}_p) = p + 1 - t$ is prime. E has embedding degree 6 if and only if there exists $l \in \mathbb{Z}$ such that $p = 4l^2 + 1$ and $t = 1 \pm 2l$.*

The strategy to generate ordinary elliptic curve of prime order with embedding degree 6 is the following :

- Select a small discriminant D which is 3 mod 8 but not 5 mod 10 and such that -8 is a quadratic residue modulo $3D$.
- Compute solutions $(X = 6l \pm 1, Y)$ of the generalized Pell equation $X^2 - 3DY^2 = -8$ until the values of p and $\#E(\mathbb{F}_p)$ corresponding to this value of l are primes of the desired size.
- Repeat with another D if not found.

The main drawback of this method is that the consecutive solutions of generalized Pell equation grow exponentially so that only very few curves are found. However, it is possible to relax the constraints in order to obtain more curves as done in [47] and in [26]. As an example, the following 192 bits curve has been find with this method [43].

Proposition 3. *Let p be the prime number given in 3.4. The curve defined over $\mathbb{F}_p$ by the equation*

$$y^2 = x^3 - 3x + 3112017650516467785865101962029621022731658738965186527433$$

has embedding degree 6 and cardinality 2ℓ where ℓ is the prime number

$$\ell = 2345624654794533338301358959942345572918215737398529094837.$$

4.3.2 Barreto-Naehrig curves

Barreto and Naerigh devised in [13] a method to generate pairing friendly elliptic curves over a prime field, with prime order and embedding degree 12. The equation of the curve is

$$y^2 = x^3 + a_6, \quad a_6 \neq 0$$

and the trace of the Frobenius t, the cardinality of the curve r and the base field $\mathbb{F}_p$ are parametrized as

- $t = 6l^2 + 1$

- $r = 36l^4 - 36l^3 + 18l^2 - 6l + 1$
- $p = 36l^4 - 36l^3 + 24l^2 - 6l + 1$

For example, $a_6 = 3$ and $l = -6000000000001F2D$ (hex) are such that both r and p are prime numbers [22].

4.4 Fast Tate pairing computation

In this section we explain how to efficiently compute the Tate pairing.

4.4.1 Formulas for Miller loop

Of course, the easiest way to speed up the pairing computation is to choose ℓ as sparse as possible. As this can be done in many cases we give only formulas for the "doubling" step of the Miller loop which are therefore representative of the whole Miller loop.

Moreover, we assume in this section that Jacobian coordinates have been chosen. Of course other choices are possible but this has almost no consequences for the pairing computation and for the results obtained in this paper. Let E be an elliptic curve defined as (2).

Let $P = (X_P, Y_P, Z_P)$ and $T = (X_T, Y_T, Z_T)$ be two points in $E(\mathbb{F}_{p^k})$ given in Jacobian coordinates and $Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^k})$ given in affine coordinates. Formulas for the "doubling" step of the Miller loop are given by

$$\begin{aligned} A &= 3X_T^2 + a_4 Z_T^4 \\ C &= 4X_T Y_T^2 \\ X_{2T} &= A^2 - 2C \\ Y_{2T} &= A(C - X_{2T}) - 8Y_T^4 \\ Z_{2T} &= 2Y_T Z_T \\ g_{T,T}(Q) &= \frac{2Y_T Z_T^3 y_Q - A(Z_T^2 x_Q - X_T) - 2Y_T^2}{2Y_T Z_T^3} \\ v_{T,T}(Q) &= \frac{x_Q Z_{2T}^2 - X_{2T}}{Z_{2T}^2} \end{aligned}$$

If the Tate pairing is used, then the point P (and consequently the point T) is in $E(\mathbb{F}_p)$ so that most of the terms in these formulas are in $\mathbb{F}_p$. The only elements lying in the extension field $\mathbb{F}_{p^k}$ are x_Q and y_Q and we will now see that, thanks to the use of twists, we can almost assume that there are lying in a proper subfield of $\mathbb{F}_{p^k}$. This will have important consequences on the efficiency of the Miller loop.

4.4.2 Use of twists

Definition 1. *Two elliptic curves E and $\tilde{E}$ defined over $\mathbb{F}_q$ are said to be twisted if there exists an isomorphism Ψ_d between E and $\tilde{E}$ defined over an extension*

$\mathbb{F}_q^d$ *of* $\mathbb{F}_q$*. The degree of this twist is the degree of the smallest extension on which* Ψ_d *is defined.*

The possible twist degree are 2,3,4 and 6 depending on the embedding degree k. We will focus in this paper on even embedding degrees k and twists of degree 2 and 6. We have the following properties

- Let ν be a non quadratic residue in $F_{p^{k/2}}$. The curves E and $\tilde{E}$ given by the equations

$$E : y^2 = x^3 + a_4x + a_6 \qquad \tilde{E} : \nu y^2 = x^3 + a_4x + a_6$$

  are twisted by the twist of order 2 (ie defined over $\mathbb{F}_{p^k}$)

$$\begin{array}{rcl} \Psi_2 : \tilde{E} & \rightarrow & E \\ (x, y) & \mapsto & (x, y\nu^{\frac{1}{2}}) \end{array}$$

- An elliptic curve E defined as in (2) has a degree 6 twist if and only if $a_4 = 0$. If ν is an element in $\mathbb{F}_{p^{k/6}}$ which is not a sixth power, then E is the twisted of the curve $\tilde{E}$ defined by

$$y^2 = x^3 + \frac{b}{\nu}$$

  by the twist of order 6

$$\begin{array}{rcl} \Psi_6 : \tilde{E} & \rightarrow & E \\ (x, y) & \mapsto & (x\nu^{\frac{1}{3}}, y\nu^{\frac{1}{2}}) \end{array}$$

Let Ψ be such a twist. As E and $\tilde{E}$ are isomorphic over $\mathbb{F}_{p^k}$, we can define a variant of the Tate pairing as $e_T(P, Q) = f_{(\ell,P)}(\Psi(Q))^{\frac{p^k-1}{\ell}}$ without loss of generality. This is nothing but the Tate pairing defined on $E(\mathbb{F}_p)[\ell] \times \tilde{E}(\mathbb{F}_{p^k})$. For the twists given above, this means that the coordinates of Q can be written as $(x_Q, y_Q\nu^{\frac{1}{2}})$ or $(x_Q\nu^{\frac{1}{3}}, y_Q\nu^{\frac{1}{2}})$ where x_Q and y_Q are defined over $\mathbb{F}_{p^{k/d}}$. There are three important consequences on the Miller loop.

- The computation of f involves only $\mathbb{F}_{p^{k/d}}$ arithmetic (but the result is still in $\mathbb{F}_{p^k}$).
- The denominators, and more generally all the factors of f lying in a proper subfield of $\mathbb{F}_{p^k}$ (as $\mathbb{F}_p$ or $\mathbb{F}_{p^{k/d}}$), are wiped out by the final exponentiation. Hence, only the expression $2Y_TZ_T^3y_Q - A(Z_T^2x_Q - X_T) - 2Y_T^2$ has to be computed before the writing up of f. This is the famous denominator elimination introduced in [9].
- In the case of twists of order 6 for Tate pairings, this expression has the particular form

$$g_0 + g_2\nu^{\frac{1}{2}} + g_3\nu^{\frac{1}{3}}$$

where $g_0 \in \mathbb{F}_p$ and $g_1, g_2 \in \mathbb{F}_{p^2}$ which contains only 5 coefficients instead of 12. Hence the multiplication by such an element during the writing up of f is cheaper than a complete multiplication in $\mathbb{F}_{p^k}$. More precisely the multiplication of an arbitrary element of $\mathbb{F}_{p^k}$ by g_0 requires 12 multiplications in $\mathbb{F}_p$ and the one by $g_2\nu^{\frac{1}{2}} + g_3\nu^{\frac{1}{3}}$ requires 27 multiplications in $\mathbb{F}_p$ using Karatsuba. Finally this operation requires 39 multiplications in $\mathbb{F}_p$ instead of 54 for a complete Karatsuba multiplication in $\mathbb{F}_{p^k}$. Even if this expression has a different form for other pairings, it is still sparse and we can also perform this multiplication with only 39 multiplications in $\mathbb{F}_p$.

4.4.3 Final exponentiation

For the Tate pairing (but also for other pairings), the exponent of the final exponentiation is $\frac{p^k-1}{\ell}$ which has the size of p^{k-1} and the base field is $\mathbb{F}_{p^k}$. Hence, at first sight, the final exponentiation seems to be very expensive. Hopefully, Koblitz and Menezes show in [36] that this cost can be reduced thanks to the factorization

$$\frac{p^k-1}{\ell} = \left(p^{k/2}-1\right)\left(\frac{p^{k/2}+1}{\Phi_k(p)}\right)\left(\frac{\Phi_k(p)}{\ell}\right)$$

where Φ_k is the k^{th} cyclotomic polynomial. The fact that $\Phi_k(p)$ divides $p^{k/2}+1$ (as polynomials in p) is a direct consequence of the definition of k as the smallest integer such that ℓ divides p^k-1. Thanks to this factorization is the two first part of the exponentiation are very fast since there are obtained via cheap Frobenius computations.

The hard part of the exponentiation is given by the exponent $\frac{\Phi_k(p)}{\ell}$. In the general case, the best way is to develop this exponent in base p so that, thanks to multi-exponentiation, its cost is the same as if an exponent of the size of p were used. However, both in the case of MNT curves and BN curves, we can do better thanks to the parametrization of p and ℓ.

For example, in the case of MNT curves, we have

$$\frac{p^6-1}{\ell} = \left(p^3-1\right)\left(\frac{p^3+1}{\Phi_6(p)}\right)\left(\frac{\Phi_6(p)}{\ell}\right)$$

with

$$\begin{aligned}\Phi_6(p) &= p^2-p+1\\ p &= 4l^2+1\\ \ell &= 4l^2-2l+1\end{aligned}$$

and an elementary calculation gives

$$\frac{p^6-1}{\ell} = \left(p^3-1\right)(p+1)(p+2l).$$

Then, the final exponentiation is obtained thanks to several easy Frobenius applications and an exponentiation with $2l$ as an exponent (which size is the half of the size of p). Note that it also involves an inversion in $\mathbb{F}_{p^6}$.

Of course, it is better to chose l as sparse as possible, but MNT curves are rare so that it is not feasible in practice. So efficient exponentiation methods, as sliding window, must be used. On the contrary, the parameter can be chosen sparse for BN curves and it is not so easy to reduce the size of the exponent. More precisely, the hard part of the final exponentiation for BN curves is [22]

$$f^{p^3}\left(b\left(f^p\right)^2 f^{p^2}\right)^{6l^2+1} b\left(f^p f\right)^9 a f^4$$

with $a = f^{6l-5}$ and $b = a^{p+1}$. So the cost is around three-quarters of the cost of an exponentiation with an exponent having the same size than p.

Finally, the first step of the final exponentiation is $f^{p^{k/2}-1}$ so that its result has order $p^{k/2}+1$. Then, as noticed in [29, 31], it is a unit so squaring such an element (which is the most used operation in the hard part of the exponentiation) is less expensive than a classical squaring in $\mathbb{F}_{p^k}$ as explain in 3.6.

4.5 Other pairings

More recently, some variants of the Tate pairing appear in the literature. The main goal is to reduce the length of the Miller loop. The price to be paid is that the points P and Q are swapped. This means that the elliptic curve arithmetic will hold in $\mathbb{F}_{p^k}$ (or $\mathbb{F}_{p^{k/d}}$ for twisted versions) instead of $\mathbb{F}_p$ so that even if the number of steps in the Miller loop will decrease, the cost of each step will increase. We give here the twisted versions of the Ate and R-Ate pairings.

4.5.1 The Ate pairing

This pairing were first introduced in [33]. It is defined by

$$\begin{aligned} e_A : E(\mathbb{F}_{p^k}) \cap \mathrm{Ker}(\pi - p) \times E(\mathbb{F}_p)[\ell] &\rightarrow \mathbb{F}_{p^k}^* / \left(\mathbb{F}_{p^k}^*\right)^\ell \\ (Q, P) &\mapsto f_{(t-1,Q)}(P)^{\frac{p^k-1}{\ell}} \end{aligned}$$

where π is the Frobenius map on the curve. This construction works because, since $Q \in \mathrm{Ker}(\pi - p)$ and $\ell | \#E(\mathbb{F}_p) = p+1-t$, we have $\pi(Q) = (t-1)Q$. Finally, because $t - 1 \approx \sqrt{\ell}$, the length of the Miller loop is divided by 2 compared to the Tate pairing. This pairing is optimal for MNT curves in the sense of [52].

4.5.2 The R-Ate pairing

This is a generalization of the Ate pairing introduced in [38]. We give here its expression only in the case of BN curves. If l is the parameter used to construct the BN curve and $b = 6l + 2$, the R-Ate pairing is defined by

$$\begin{aligned} e_R \;:\; & E(\mathbb{F}_{p^k}) \cap \mathrm{Ker}(\pi - p) \times E(\mathbb{F}_p)[\ell] \rightarrow \mathbb{F}_{p^k}^* / \left(\mathbb{F}_{p^k}^*\right)^\ell \\ & (Q, P) \mapsto \left(f_{(b,Q)}(P) . \left(f_{(b,Q)}(P) . g_{(bQ,Q)}(P)\right)^p . g_{(\pi((b+1)Q),bQ)}(P)\right)^{\frac{p^k-1}{\ell}} \end{aligned}$$

where $g_{(A,B)}$ is the equation of the line passing through A and B. In this case, $l \approx \sqrt[4]{\ell}$, so the length of the Miller loop is divided by 4 compared to the Tate pairing. This pairing is optimal for BN curves in the sense of [52].

5 RNS arithmetic for fast pairing computation

When the embedding degree is large, the RNS does not yield to any noteworthy improvement on the computation of T or $g(Q)$ but, thanks to its linear complexity regarding the extension degree, we obtain very important improvements on the writing up of f and the final exponentiation. As these are the most expensive steps of the Tate pairing computation, we can say that the RNS arithmetic is particularly well adapted to fast pairing computation. But let us see more in details the expected gains for most popular pairing friendly curves (in large characteristic), say MNT curves and BN curves.

5.1 Using RNS for MNT curves

We assume in this section that E is an elliptic curve defined over $\mathbb{F}_p$ by an equation (2) and obtained as described in section 4.3.1 so that $p = 4l^2 + 1$ for some integer l. We also assume that $\mathbb{F}_{p^6}$ is built as a quadratic extension of $\mathbb{F}_{p^3}$

$$\mathbb{F}_{p^6} = \mathbb{F}_{p^3}[Y]/(Y^2 - \nu) = \mathbb{F}_{p^3}[\boldsymbol{\beta}].$$

5.1.1 Tate Pairing

As MNT curves have a twist of order 2, the input elements of the Tate pairing can be written in the form $P = (x_P, y_P) \in E(\mathbb{F}_p)[\ell]$ and $Q = (x_Q, y_Q\boldsymbol{\beta})$ with x_Q and $y_Q \in \mathbb{F}_{p^3}$. Applying the improvements explained in section 4.4 the Tate pairing is then given by the algorithm 4.

Let us now precisely analyze the complexity of each line of this algorithm. Note that we choose to not distinguish multiplication and squaring in $\mathbb{F}_p$ for simplicity and because this has no notable consequence on our study.

- The lines 1 to 4 are the standard doubling of the point $T \in E(\mathbb{F}_p)$ in Jacobian coordinates and require 10 multiplications in $\mathbb{F}_p$ (if a_4 has no special form). Lazy reduction can be used when computing A and Y_{2T} so that 8 modular reductions are necessary.

- In the same way, the lines 7 to 10 are for the mixed addition of T and P and require 11 multiplications and 10 reductions in $\mathbb{F}_p$.

- As x_Q and y_Q are in $\mathbb{F}_{p^3}$, the line 5 requires 9 multiplications in $\mathbb{F}_p$. Note that this is not a good idea to write $A(Z_T^2 x_Q + X_T)$ in line 5 because it requires two multiplications of an element of $\mathbb{F}_{p^3}$ by an element of $\mathbb{F}_p$ instead of one. However, we can use the lazy reduction technique on the constant term of this expression. Finally, 8 modular reductions are necessary.

Algorithm 4: Tate(P, Q)

Data: $p = 4l^2 + 1$ prime, E a MNT elliptic curve defined over $\mathbb{F}_p$.
$\ell | \#E(\mathbb{F}_p)$ prime with binary representation $(\ell_{s-1}, \cdots, \ell_0)_2$.
$P = (x_P, y_P) \in E(\mathbb{F}_p)[\ell], Q = (x_Q, y_Q\boldsymbol{\beta})$ with x_Q and $y_Q \in \mathbb{F}_{p^3}$
Result: $e_T(P, Q) \in \mathbb{F}_{p^6}$.
begin
  $T = (X_T, Y_T, Z_T) \leftarrow (x_P, y_P, 1)$
  $f \leftarrow 1$
  for *i from* $s-2$ *downto 0* **do**
1 $A = 3X_T^2 + a_4 Z_T^4, \qquad C = 4X_T Y_T^2$
2 $X_{2T} = A^2 - 2C$
3 $Y_{2T} = A(C - X_{2T}) - 8Y_T^4$
4 $Z_{2T} = 2Y_T Z_T$
5 $g = Z_{3T} Z_T^2 y_Q \boldsymbol{\beta} - A Z_T^2 x_Q + A X_T - 2Y_T^2$
6 $f \leftarrow f^2.g$
  $T \leftarrow [X_{2T}, Y_{2T}, Z_{2T}]$
  if $\ell_i = 1$ **then**
7 $E = X_T - x_P Z_T^2, \qquad F = Y_T - y_P Z_T^3$
8 $X_{T+P} = F^2 - 2X_T E^2 - E^3$
9 $Y_{T+P} = F(X_T E^2 - X_{T+P}) - Y_T E^3$
10 $Z_{T+P} = Z_T E$
11 $g = Z_{T+P} y_Q \boldsymbol{\beta} - Z_{T+P} y_P - F(x_Q - x_P)$
12 $f \leftarrow f.g$
  $T \leftarrow [X_{T+P}, Y_{T+P}, Z_{T+P}]$
  endif
  endfor
13 $f \leftarrow f^{p^3 - 1}$
14 $f \leftarrow f^{p+1}$
15 $f \leftarrow f^p . f^{2l}$
  return f
end

- The situation is the same for the line 11 which requires 7 multiplications and 6 modular reductions in $\mathbb{F}_p$.
- The line 6 involves both a multiplication and a squaring in $\mathbb{F}_{p^6}$. The cost of such operations is detailed in section 3.3. We deduce that the total complexity for this line is 30 multiplications and 12 modular reductions in $\mathbb{F}_p$.
- The situation for the line 12 is similar and leads to 18 multiplications and 6 reductions.
- The line 13 is computed as $\frac{f^{p^3}}{f}$. As $f \in \mathbb{F}_{p^6}$, the exponentiation by p^3 is nothing but the conjugation on $\mathbb{F}_{p^3}[\boldsymbol{\beta}]$ so it is for free. Finally, this step requires a multiplication and an inversion in $\mathbb{F}_{p^6}$. As recalled in section 3.6, such an inversion can be done with only one inversion, 36 multiplications and 16 reductions in $\mathbb{F}_p$. Finally, this first step of the final exponentiation requires 54 multiplications, 22 modular reductions and one inversion in $\mathbb{F}_p$.
- The line 14 involves one multiplication in $\mathbb{F}_{p^6}$ and one application of the Frobenius map. We have seen in section 3.6 that, if the polynomial defining $\mathbb{F}_{p^6}$ is well chosen (which is always the case in practice) the Frobenius map requires only 5 modular multiplications in $\mathbb{F}_p$. Hence this second step of the final exponentiation require 23 multiplications and 11 reductions in $\mathbb{F}_p$.
- The hard part of the final exponentiation involves one Frobenius (ie 5 modular multiplications), one multiplication in $\mathbb{F}_{p^6}$ (18 multiplications and 6 reductions) and one exponentiation by $2l$. We have already seen that l cannot be chosen sparse for MNT curves, so that advanced exponentiation methods must be used. In line 13, f has been raised to the power p^3-1, so that it is a unit and can be squared with only 2 squarings and 2 reductions in $\mathbb{F}_{p^3}$ (and then 12 multiplications and 6 reductions in $\mathbb{F}_p$) as explained in section 3.6. Then, for each step of this exponentiation, the cost is 12 multiplications and 6 reductions and 18 additional multiplications and 6 reductions if a multiplication is required by the exponentiation algorithm.

It is now necessary to fix the security level to have an idea of the overall complexity of the Tate pairing. We choose a 96 bits security level which is quite reasonable for MNT curves, so that ℓ has bit-length $s = 192$. This means that lines 1 to 6 are done 191 times and lines 7 to 12 around 96 times. Hence the Miller loop requires

$$\begin{array}{ll} 191\times(10+9+30)+96\times(11+7+18) = \mathbf{12815} & \text{multiplications and} \\ 191\times(8+8+12)+96\times(10+6+6) = \mathbf{7460} & \text{reductions} \end{array}$$

The easy parts of the final exponentiation requires one inversion, 77 multiplications and 33 reductions in $\mathbb{F}_p$. Concerning the hard part of the final exponentiation, as $2l$ is 96 bits long, it is reasonable to use sliding window method with

a window size of 3 for computing f^{2l}. This requires 96 squarings in $\mathbb{F}_{p^6}$ and, on average, 24 multiplications in $\mathbb{F}_{p^6}$ (plus 3 for precomputations). Finally, the hard part of the final exponentiation requires

$$\begin{array}{ll} 5+18+97\times12+27\times18 = \mathbf{1673} & \text{multiplications and} \\ 5+6+97\times6+27\times6 = \mathbf{755} & \text{reductions} \end{array}$$

Then, the full Tate pairing computation requires 14565 multiplications but only 8248 reductions. This is only an estimate. We can have more precise complexity if the curve is given but it will have negligible consequences on our result. For example, for the curve given in proposition 3, ℓ has Hamming weight 102 and the computation of f^{2l} involves 28 multiplication using 3 as window size.

For this level of security, 6 (32 bits) words are necessary so a radix implementation requires $14565\times6^2+8248\times(6^2+6) = 870756$ word multiplications whereas a RNS implementation requires $1.1\left(14565\times2\times8+8248\times\left(\frac{7}{5}8^2+\frac{8}{5}8\right)\right) = 736626$ word multiplications. This represents a gain of 15.4%.

5.1.2 Ate pairing

The algorithm is very similar to the algorithm 4 but the arguments P and Q are swapped. This means that operations of the lines 1 to 4 are done in $\mathbb{F}_{p^3}$ so they require 10 multiplications and 8 reductions in $\mathbb{F}_{p^3}$, i.e. 60 multiplications and 24 reductions in $\mathbb{F}_p$. In the same way the lines 7 to 10 require 11 multiplications and 10 reductions in $\mathbb{F}_{p^3}$, i.e. 66 multiplications and 30 reductions in $\mathbb{F}_p$. It is easy to prove that, if the coordinates of T are $(X_T, Y_T\boldsymbol{\beta}, Z_T)$, the lines 5 and 11 must be replaced by

$$\begin{array}{rrcl} \mathbf{5'} & g & = & Z_{2T}Z_T^2y_P\boldsymbol{\beta} + A(X_T - Z_T^2x_P) - 2\nu Y_T^2 \\ \mathbf{11'} & g & = & -Z_{T+P}y_Q\boldsymbol{\beta} + Z_{T+P}y_P - F(x_P - x_Q) \end{array}$$

where Z_{2T}, Z_T^2, $A = 3X_T^2 - a_4Z_T^4$, Y_T^2, Z_{T+P} and $F = Y_T - y_QZ_T^3$ were computed in $\mathbb{F}_{p^3}$ during the previous steps. The first requires 18 multiplications and 12 reductions in $\mathbb{F}_p$ whereas the second requires 15 multiplications and 6 reductions.

Finally, since $t-1$ has bitlength 96 and Hamming weight around 48 the total cost of the Miller loop is

$$\begin{array}{ll} 95\times(60+18+30)+47\times(66+15+18) = \mathbf{14913} & \text{multiplications and} \\ 95\times(24+12+12)+47\times(30+6+6) = \mathbf{6534} & \text{reductions} \end{array}$$

The final exponentiation is the same as for the Tate pairing then, the full Ate pairing computation requires 16663 multiplications but only 7322 reductions. This yields to 907392 word multiplications in radix representation but only 703204 in RNS. This represents a gain of 22.5%. Note that the Ate pairing is not really interesting for MNT curves because most of the conputations are done in $\mathbb{F}_{p^3}$. This is not the case for BN curves because the twist used has order 6 so that the arithmetic involved in Ate pairing will be in $\mathbb{F}_{p^2}$. Moreover, we can again half the exponent in this case thanks to the R-Ate pairing.

5.2 Using RNS for BN curves

BN curves have a twist of order 6 so that all the improvements given in section 4.4 can be used. One of the reasons of the current success of BN curves.

We assume in this section that E is an elliptic curve defined over $\mathbb{F}_p$ by an equation of the form

$$y^2 = x^3 + a_6$$

and obtained as described in 4.3.2. We also assume that $\mathbb{F}_{p^{12}}$ is built as a quadratic extension of a cubic extension of $\mathbb{F}_{p^2}$ which is easily compatible with the use of twists of order 6. More precisely, let ν be an element in $\mathbb{F}_{p^2}$ which is not a sixth power, we built

$$\begin{array}{rcccl} \mathbb{F}_{p^6} & = & \mathbb{F}_{p^2}[Y]/(Y^3 - \nu) & = & \mathbb{F}_{p^2}[\boldsymbol{\beta}] \\ \mathbb{F}_{p^{12}} & = & \mathbb{F}_{p^6}[Z]/(Z^2 - \boldsymbol{\beta}) & = & \mathbb{F}_{p^6}[\boldsymbol{\gamma}]. \end{array}$$

Thus $\mathbb{F}_{p^{12}}$ can also be defined by $\mathbb{F}_{p^2}[\boldsymbol{\gamma}]$

5.2.1 Tate pairing

Thanks to the twist defined by ν, the second input of the Tate pairing can be written

$$Q = (x_Q\boldsymbol{\gamma}^2, y_Q\boldsymbol{\gamma}^3) \text{ with } x_Q \text{ and } y_Q \in \mathbb{F}_{p^2}.$$

Applying the improvements explained in section 4.4 the Tate pairing is given by to the algorithm 5.

Let us now analyze the complexity of each line of this algorithm.

- As explained for MNT curves, the lines 1 to 4 require 7 multiplications and 6 modular reductions whereas the lines 7 to 10 require 11 multiplications and 10 modular reductions.
- As x_Q and y_Q are in $\mathbb{F}_{p^2}$, the line 5 requires 8 modular multiplications in $\mathbb{F}_p$ and lazy reduction cannot be used.
- In the same way, the line 11 requires 6 multiplications but only 5 modular reductions because lazy reduction is used on the constant term.
- The line 6 involves both a squaring and a multiplication in $\mathbb{F}_{p^6}$. As explained in section 3.5, such a squaring involves 36 multiplications and 12 modular reductions. We have seen in 4.4.2 that, thanks to its special form, the multiplication by g requires only 39 multiplications and 12 reductions. We deduce that the total complexity for this line is 75 multiplications and 24 reductions in $\mathbb{F}_p$.
- The situation for the line 12 is similar and leads to 39 multiplications and 12 reductions.

Algorithm 5: Tate(P, Q)

Data:
$P = (x_P, y_P) \in E(\mathbb{F}_p)[\ell], Q = (x_Q\boldsymbol{\gamma}^2, y_Q\boldsymbol{\gamma}^3)$ with x_Q and $y_Q \in \mathbb{F}_{p^2}$
Result: $e_T(P, Q) \in \mathbb{F}_{p^{12}}$.
begin
  $T = (X_T, Y_T, Z_T) \leftarrow (x_P, y_P, 1)$
  $f \leftarrow 1$
  for *i from* $l-2$ *downto 0* **do**
1 $A = 3X_T^2, \qquad C = 4X_TY_T^2$
2 $X_{2T} = A^2 - 2C$
3 $Y_{2T} = A(C - X_{2T}) - 8Y_T^4$
4 $Z_{2T} = 2Y_TZ_T$
5 $g = Z_{2T}Z_T^2y_Q\boldsymbol{\gamma}^3 - AZ_T^2x_Q\boldsymbol{\gamma}^2 + AX_T - 2Y_T^2$
6 $f \leftarrow f^2.g$
  $T \leftarrow [X_{2T}, Y_{2T}, Z_{2T}]$
  if $m_i = 1$ **then**
7 $E = X_T - x_PZ_T^2, \qquad F = Y_T - y_PZ_T^3$
8 $X_{T+P} = F^2 - 2X_TE^2 - E^3$
9 $Y_{T+P} = F(X_TE^2 - X_{T+P}) - Y_TE^3$
10 $Z_{T+P} = Z_TE$
11 $g = Z_{T+P}y_Q\boldsymbol{\gamma}^3 - Fx_Q\boldsymbol{\gamma}^2 - Z_{T+P}y_P + Fx_P)$
12 $f \leftarrow f.g$
  $T \leftarrow [X_{T+P}, Y_{T+P}, Z_{T+P}]$
  endif
  endfor
13 $f \leftarrow f^{p^6-1}$
14 $f \leftarrow f^{p^2+1}$
15 $a \leftarrow f^{6l-5}, b \leftarrow a^{p+1}$
16 $f \leftarrow f^{p^3}.\left[b.(f^p)^2.f^{p^2}\right]^{6l^2+1}.b.\left(f^{p+1}\right)^9.a.f^4$
  return f
end

- The line 13 is computed as $\frac{f^{p^6}}{f}$. As computing f^{p^6} is for free (conjugation), this step requires a multiplication and an inversion in $\mathbb{F}_{p^{12}}$. We have seen in section 3.6 that an inversion can be done with only one inversion, 97 multiplications and 35 reductions in $\mathbb{F}_p$. Finally, this first step of the final exponentiation requires 151 multiplications, 47 modular reductions and one inversion in $\mathbb{F}_p$.
- The line 14 involves one multiplication in $\mathbb{F}_{p^{12}}$ and one powering to p^2. We have seen in section 3.6 that the Frobenius map (and its iterations) requires 11 modular multiplications in $\mathbb{F}_p$. Thus, this step requires 65 multiplications and 23 reductions in $\mathbb{F}_p$.
- The hard part of the final exponentiation is given by the lines 15 and 16. The line 15 involves one Frobenius (i.e. 11 modular multiplications), one multiplication in $\mathbb{F}_{p^{12}}$ (54 multiplications and 12 reductions) and one exponentiation. Contrary to MNT curves, l can be chosen sparse for BN curves, so classical square-and-multiply can be used. Moreover, in line 13, f has been raised to the power $p^6 - 1$, so it is a unit and can be squared with only 2 squarings and 2 reductions in $\mathbb{F}_{p^6}$ (and then 24 multiplications and 12 reductions in $\mathbb{F}_p$) as explained in section 3.6. Thus, for most steps of this exponentiation, the cost is 24 multiplications and 12 reductions. For the steps corresponding to non-zero bits of the exponent, 54 additional multiplications and 12 additional reductions are necessary.
- The line 16 involves 4 applications of the Frobenius map (or its iterations), 9 multiplications and 6 (unit) squarings in $\mathbb{F}_{p^{12}}$ (which means 674 multiplications and 224 reductions in $\mathbb{F}_p$). It also involves an exponentiation which has the same properties as the one of line 15 but which is two times larger.

Again, it is necessary to fix the security level to have an idea of the overall complexity of the Tate pairing. We choose a 128 bits security level for which BN curves are well suited. We also have to fix the Hamming weight of l (and consequently the one of ℓ). We assume here that l has weight 11 and ℓ has weight 90 as in the example given in section 4.3.2. Of course, smaller values of these weight can probably be found but this is not the aim of this paper and it has negligible effects on our results. This means that lines 1 to 6 are done 255 times and lines 7 to 12 89 times. Hence the Miller loop requires

$$\begin{array}{ll} 255\times(7+8+75)+89\times(11+6+39) = \mathbf{27934} & \text{multiplications and} \\ 255\times(6+8+24)+89\times(10+5+12) = \mathbf{12093} & \text{reductions} \end{array}$$

The easy parts of the final exponentiation require one inversion, 216 multiplications and 70 reductions in $\mathbb{F}_p$. Finally, the hard part of the final exponentiation involves one exponentiation by $6l - 5$ which has Hamming weight 11 and by $6l^2 + 1$ which has hamming weight 28. However, as mentioned in [31] the second exponentiation can be split in 2 parts with exponents l and $6l$ both having Hamming weight 11. Hence, only 21 multiplications are required for this exponentiation and the lines 15 and 16 require

$11+54+65\times24+9\times54+674+127\times24+21\times54 = \mathbf{6967}$ multiplications and
$11+12+65\times12+9\times12+224+127\times12+21\times12 = \mathbf{2911}$ reductions.

Then, the full Tate pairing computation requires 35117 multiplications but only 15074 reductions. Note that we obtain the same number of multiplications than in [31] (since we use the same optimizations) but the number of reductions is much smaller.

For this level of security, 8 (32 bits) words are necessary so a radix implementation requires $35117\times8^2+15074\times(8^2+8) = 3332816$ word multiplications whereas a RNS implementation requires $1.1\left(35117\times2\times8+15074\times\left(\frac{7}{5}8^2+\frac{8}{5}8\right)\right) = 2315994$ word multiplications. This represents a gain of 30.5%.

5.2.2 Ate pairing

The algorithm is very similar to algorithm 5 but the arguments P and Q are swapped. This means that operations of the lines 1 to 4 are done in $\mathbb{F}_{p^2}$ so they require 3 multiplications, 4 squaring and 6 reductions in $\mathbb{F}_{p^2}$, i.e. 17 multiplications and 12 reductions in $\mathbb{F}_p$. In the same way the lines 7 to 10 require 8 multiplications, 3 squaring and 10 reductions in $\mathbb{F}_{p^2}$, i.e. 30 multiplications and 20 reductions in $\mathbb{F}_p$. It is easy to prove that, if the coordinates of T are $(X_T\boldsymbol{\gamma}^2, Y_T\boldsymbol{\gamma}^3, Z_T)$, the lines 5 and 11 must be replaced by

$$\begin{aligned} \mathbf{5}'' \quad g &= Z_{2T}Z_T^2y_P - AZ_T^2x_P\boldsymbol{\gamma} + (AX_T - 2Y_T^2)\boldsymbol{\gamma}^3 \\ \mathbf{11}'' \quad g &= Z_{T+P}y_P - Fx_P\boldsymbol{\gamma} + (Fx_Q - Z_{T+P}y_Q)\boldsymbol{\gamma}^3 \end{aligned}$$

where Z_{2T}, $A = 3X_T^2$, Y_T^2, Z_{T+P} and $F = Y_T - y_QZ_T^3$ were computed during the previous steps. The first requires 15 multiplications and 12 reductions in $\mathbb{F}_p$ whereas the second requires 10 multiplications and 6 reductions. Moreover, the value obtained for g has only terms in $\boldsymbol{\gamma}, \boldsymbol{\gamma}^3$ and a constant term, so that a multiplication by g requires only 39 multiplications instead of 54 as explained in 4.4.2.

Finally as, in our example, $t-1$ has bitlength 128 and Hamming weight 29 the total cost of the Miller loop is

$127\times(17+15+36+39)+28\times(30+10+39) = \mathbf{15801}$ multiplications and
$127\times(12+12+24)+28\times(20+6+12) = \mathbf{7160}$ reductions

The final exponentiation is the same as for the Tate pairing. Then, the full Ate pairing computation requires 22984 multiplications but only 10241 reductions. This yields to 2208328 word multiplications in radix representation but only 1558065 in RNS. This represents a gain of 29.5%.

5.2.3 R-ate pairing

In the R-ate pairing, the operations in the (shorter) Miller loop are the same but an additional step is necessary at the end of the Miller loop. This step is the computation of

$$f.\left(f.g_{(T,Q)}(P)\right)^p.g_{(\pi(T+Q),T)}(P)$$

where $T = (6l+2)Q$ is computed during the Miller loop and π is the Frobenius map on the curve. This step involves following operations.

- One step of addition as in the Miller loop (computation of $T+Q$ and $g_{(T,Q)}(P)$) which requires 40 multiplications and 26 reductions in $\mathbb{F}_p$.
- One application of the Frobenius map on the curve. As $p \equiv 1 \bmod 6$ for BN curves, this operation requires only 2 multiplications in $\mathbb{F}_{p^2}$ by precomputed values.
- One non-mixed addition step (computation of $g_{(\pi(T+Q),T)}(P)$). It is easy to prove that it requires 60 multiplications and 40 reductions in $\mathbb{F}_p$.
- Two multiplication by the previous results, both requiring 39 multiplications and 12 reductions in $\mathbb{F}_p$.
- One Frobenius requiring 11 modular multiplications.
- One full multiplication in $\mathbb{F}_{p^{12}}$ requiring 54 multiplications and 12 reductions in $\mathbb{F}_p$.

Then, this step requires 249 multiplications and 117 reductions in $\mathbb{F}_p$. On the other hand, in our example, $6l+2$ has bitlength 66 and Hamming weight 9 so the cost of the Miller loop is

$$\begin{array}{ll} 65\times(17+15+36+39)+8\times(30+10+39) = \mathbf{7587} & \text{multiplications and} \\ 65\times(12+12+24)+8\times(20+6+12) = \mathbf{3424} & \text{reductions} \end{array}$$

The final exponentiation is the same as for the Tate pairing then, the full R-Ate pairing computation requires 15019 multiplications but only 6405 reductions. This yields to 1422376 word multiplications in radix representation but only 985794 in RNS. This represents a gain of 30.7%.

6 Conclusion

In this work we explained why the RNS arithmetic is particularly well suited for computation in extension fields essentially thanks to the use of lazy reduction. As a consequence, it is interesting to use such an arithmetic for pairing computation in large characteristic especially in contexts where the other advantages of the RNS arithmetic can be exploited (like hardware and/or parallel implementation). More precisely, we proved that using RNS for MNT curves when 96 bits of security are required involves 15 to 22.5% less basic operations. This gain reach 30% for BN curves with 128 bits of security whatever the pairing used (Tate, Ate, R-Ate). Most of the gains comes from arithmetic in extension fields so that choosing other systems of coordinate (affine, projective, Edwards, ...) or other pairings will not change these results. Moreover, it is beyond doubt that better gain will occur in other situations in pairing-based cryptography for two main reasons.

- For larger security levels or for curves of non-prime order (i.e. such that $\frac{\log(p)}{\log(\ell)} > 1$), the number of words necessary to represent elements in $\mathbb{F}_p$ will increase which is favorable to RNS arithmetic.
- Larger embedding degrees involve arithmetic in larger extension field which is linear in RNS but quadratic in radix representation.

Finally RNS arithmetic is very interesting for efficient pairing implementation and it would be attractive to develop a dedicated architecture such as the one described in [5].

References

[1] Bajard, J.C., Didier, L.S., Kornerup, P.: A RNS Montgomery's Modular Multiplication. IEEE Transactions on Computers, **47:7** (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.: Combining leak-resistant arithmetic for elliptic curves defined over $\mathbb{F}_p$ and RNS representation. Cryptology eprint archive 311 (2010).

[4] Bajard, J.C., Duquesne, S., Ercegovac, M., Meloni, N.: Residue systems efficiency for modular products summation: Application to Elliptic Curves Cryptography. SPIE **6313**, 631304 (2006).

[5] Guillermin, N.: A high speed coprocessor for elliptic curve scalar multiplications over $\mathbb{F}_p$. CHES 2010, LNCS **6225** (2010) 48–64.

[6] Bajard, J.C., Imbert, L.: A full RNS implementation of RSA. IEEE Transactions on Computers **53:6** (2004) 769–774.

[7] Bajard, J.C., Kaihara, M., Plantard, T.: Selected RNS Bases for Modular Multiplication. Proceedings of the 19th IEEE symposium on Computer Arithmetic (2009).

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

[9] Barreto, P., Kim, H., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. Crypto 2002, LNCS **2442** (2002) 354–369.

[10] Barreto, P., Lynn, B., Scott, M.: Constructing elliptic curves with prescribed embedding degrees. SCN 2002, LNCS **2576** (2002) 257-267.

[11] Barreto, P., Lynn, B., Scott, M.: On the selection of pairing friendly groups. SAC 2003, LNCS **3006** (2003) 17–25.

[12] Barreto, P., Lynn, B., Scott, M.: Efficient implementation of pairing-based cryptosystems. J. Cryptology **17:4** (2004) 321-334.

[13] Barreto, P., Naehrig, M.: Pairing-friendly elliptic curves of prime order. SAC 2005, LNCS **3897** (2005) 319-331.

[14] Blake, I.F., Seroussi, G., Smart, N.P.: Advances in Elliptic Curve Cryptography. Cambridge University Press (2004).

[15] Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. Crypto 2001, LNCS **2139** (2001) 213–229.

[16] Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. Journal of Cryptology **17:4** (2004) 297–319.

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

[18] Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Designs, Codes and Cryptography **37:1** (2005) 133-141.

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

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

[21] Devegili, A.,Ó hÉigeartaigh, C., Scott, M., Dahab, R.: Multiplication and squaring on pairing-friendly fields. Cryptology ePrint Archive, **71** (2006).

[22] Devegili, A., C., Scott, M., Dahab, R.: Implementing cryptographic pairings over Barreto-Naehrig curves. Pairing 2007, LNCS **4575** (2007) 197-207.

[23] Freeman, D.: Constructing pairing-friendly elliptic curves with embedding degree 10. ANTS 2006, LNCS **4076** (2006) 452-465.

[24] Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. Cryptology **23:2** (2010) 224-280.

[25] Frey, G., Ruck, H.G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Mathematics of Computation **62:206** (1994) 865–874.

[26] Galbraith, S., McKee, J. and Valenca, P.: Ordinary abelian variety having small embedding degree. Finite Fields and Applications **13** (2007) 800–814.

[27] Galbraith, S., Scott M. Exponentiation in pairing-friendly groups using homomorphisms. Pairing 2008, LNCS **5209** (2008) 211-224.

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

[29] Granger, R., Scott M. Faster Squaring in the Cyclotomic Subgroup of Sixth Degree Extensions PKC 2010, LNCS **6056** (2010) 7–13.

[30] GNU MP, `http://gmplib.org`

[31] Hankerson, D., Menezes, A., Scott, M.: Software implementation of pairings. In M. Joye and G. Neven, editors, Identity-based Cryptography, Cryptology and Information Security Series (2009) 188–206.

[32] Hess, F.: Pairing lattices. Pairing 2008, LNCS **5209** (2008) 18-38.

[33] Hess, F.,Smart, N.P.,Vercauteren, F.: The Eta-pairing revisited. IEEE Transactions on Information Theory, **52:10** (2006) 4595-4602.

[34] Joux, A.: A one round protocol for tripartite Diffie-Hellman. ANTS IV, LNCS **1838** (2000) 385–394.

[35] Kachisa, E., Schaefer, E., Scott, M.: Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. Pairing 2008, LNCS **5209** (2008) 126-135.

[36] Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. Cryptography and Coding, LNCS **3796** (2005) 13–36.

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

[38] Lee, E;, Lee, H.-S. Lee, Park, C.-M.: Efficient and generalized pairing computations on abelian varieties. IEEE Transactions on Information Theory **55** (2009) 1793-1803.

[39] Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for FR-reduction. IEICE Transactions on Fundamentals **84:5** (2001) 1234-1243.

[40] Menezes, A., Okamoto, T., Vanstone, S.: Reducing elliptic curve logarithms in a finite field. IEEE Transactions on Information Theory **39:5** (1993) 1639–1646.

[41] Miller, V.: The Weil pairing and its efficient calculation, Journal of Cryptology **17** (2004) 235261.

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

[43] Page, Smart, N., Vercauteren, F.: A comparison of MNT curves and supersingular curves. Applicable Algebra in Engineering, Communication and Computing **17:5** (2006) 379–392.

[44] Schirokauer, O.: The number field sieve for integers of low weight. Mathematics of Computation **79** (2010) 583–602.

[45] Scott, M.: Computing the Tate pairing. CT-RSA 2005, LNCS **3376** (2005) 13–36.

[46] Scott, M.: Implementing cryptographic pairings. Pairing 2007, LNCS **4575** (2007) 177-196.

[47] Scott, M. and Barreto, P.: Generating more MNT elliptic curves. Designs, Codes and Cryptography **38** (2006) 209–217.

[48] Scott, M., Benger, N., Charlemagne, M., Dominguez Perez, L., Kachisa, E.: On the final exponentiation for calculating pairings on ordinary elliptic curves. Pairing 2009, LNCS **5671** (2009) 78-88.

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

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

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

[52] Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory **56:1** (2010) 455-461.