# Montgomery-friendly primes and applications to cryptography

Jean-Claude Bajard, Sylvain Duquesne

# Montgomery-friendly primes and applications to cryptography

Jean Claude Bajard[1] and Sylvain Duquesne[2]

jean-claude.bajard@sorbonne-universite.fr, sylvain.duquesne@univ-rennes1.fr
[1]Sorbonne Université, CNRS, INRIA, Institut de Mathématiques de Jussieu-Paris Rive Gauche, Ouragan, F-75005 Paris, France.
[2]Univ Rennes, CNRS, IRMAR - UMR 6625, F-35000 Rennes, France.

**Abstract**

This paper deals with Montgomery-friendly primes designed for the modular reduction algorithm of Montgomery. These numbers are scattered in the literature and their properties are partially exploited. We exhibit a large family of Montgomery-friendly primes which give rise to efficient modular reduction algorithms. We develop two main uses. The first one is dedicated directly to cryptography, in particular for isogeny based approaches and more generally to Elliptic Curves Cryptography. We suggest more appropriate finite fields and curves in terms of complexity for the recommended security levels, for both isogeny-based cryptography and ECC. The second use is purely arithmetic, and we propose families of alternative RNS bases. We show that, for dedicated architectures with word operators, we can reach, for a same or better complexity, larger RNS bases with Montgomery-friendly pairwise co-primes than the RNS bases generally used in the literature with Pseudo-Mersenne numbers. This is particularly interesting for modular arithmetic used in cryptography.

## Introduction

Montgomery-friendly primes were introduced in [27, 14, 15] for cryptographic applications on elliptic or hyperelliptic curves. They are an alternative to Mersenne or pseudo-Mersenne primes as long as the Montgomery reduction is used. More precisely, they have the form $p = 2^{e_2}\alpha \pm 1$ where $2^{e_2}$ is an upper bound for the reduction coefficient $2^r$ in Montgomery reduction, so that the reduction steps are simplified by the fact that $p = \pm 1 \bmod 2^r$.
In isogeny-based cryptography, the involved primes naturally have this form, so that optimized Montgomery reduction can be used for efficient implementations [4]. In fact, the initial motivation of this work was to propose efficient Montgomery-friendly primes of the form $2^{e_2}\alpha - 1$ for isogeny-based cryptography. However, the results obtained eventually went fare beyond this field and proved to have particularly interesting applications in a wider range of areas of both public-key cryptography and Residue Number Systems (RNS) arithmetic. As an example, we exhibit new prime numbers for elliptic curve cryptography which provide a simpler reduction algorithm than the ones obtained with the prime numbers given in [27, 15].
We thus present three main results. We propose new primes for SIKE-like isogeny-based protocols. They are are better balanced between the powers of 2 and 3, and therefore offer better complexity costs in function of the security level. Then, we construct Montgomery-friendly primes which offer both efficient modular reduction and secure elliptic curves for ECC (Elliptic Curve Cryptography). And we suggest some Montgomery-friendly RNS bases offering an efficient internal reduction algorithm, which makes them competitive with respect to pseudo-Mersenne reductions.

The organization of the paper is as follows. We first present in Section 1 a brief review of the primes offering specific efficient modular reduction algorithms like the Mersenne primes, the pseudo-Mersenne primes and their generalization as Solinas primes. We also recall the Montgomery reduction as well as its Residue Number System (RNS) version, and remind that most of the bases in the literature use pairwise pseudo-Mersenne co-primes. Then, in Section 2, we give general modular reduction algorithms for Montgomery-friendly primes and explain how to smartly choose their parameters. Section 3 is dedicated to the analysis of the SIKE protocol based on isogeny, and we produce new primes offering an optimized complexity for a given security. We also exhibit, in Section 4, new primes and curves for elliptic curves cryptography that can be very interesting alternatives to the existing ones (e.g. Curve25519) from an efficiency viewpoint. Section 5 is devoted to the construction of RNS bases with Montgomery's pairwise co-primes, and to the adaptation of the RNS Montgomery reduction, as could be used in cryptography. We compare our approach with those in the literature.

## 1 Modular arithmetic for prime fields

The goal of this section is to recall the main techniques for efficient modular reduction in large prime fields of characteristic $p$. It will be used in the context of modular multiplication in $\mathbb{F}_p$, which means that we want to reduce an integer $a$ which is smaller than $p^2$ (as the result of the multiplication step). For simplicity reasons and without loss of generality, we will in fact assume that $a < 2^e p$ if $p$ is $e$-bits long.

### 1.1 Use of Mersenne or pseudo-Mersenne primes

#### 1.1.1 Mersenne primes

The simplest prime numbers for efficient reduction are the so-called Mersenne primes. They have the form $p = 2^e - 1$ and are historically used to break large prime numbers records. In this case, we use that $2^e = 1 \bmod p$ to get the reduction Algorithm 1: the input $a$ is split after $e$ bits and the sum of the low and the high parts equals to $a$ modulo $p$ and is less than $2p$. We then only have to substract $p$ if this sum is greater than or equal to $p$ to get the expected reduction. As this comparison with $p$ is not trivial, we used the standard trick which consists in subtracting $p - 2^e$ (which is $-1$ in this case) and then compare with $2^e$ (which is easy in base 2).

**Algorithm 1:** Reduction modulo a Mersenne prime

```
Data: $p = 2^e - 1$ a Mersenne prime
Input: $0 \leq a < 2^e p$
Result: $r = a \bmod p$ and $0 \leq r < p$
    Write $a = a_1 2^e + a_0$                          // division by $2^e$
    $r \leftarrow a_0 + a_1$
    $r' \leftarrow r + 1$                               // $r' \leftarrow r - p + 2^e$
    if $r' \geq 2^e$ then
        $r \leftarrow r' \bmod 2^e$                     // $r \leftarrow r - p$ if $r \geq p$
    end if
return $r$
```

The complexity of this algorithm is one addition of $e$-bits numbers and one incrementation. It is very attractive especially if $e$ is a multiple of the word-size $w$. Unfortunately, very few Mersenne primes are available for cryptographic sizes. The most known is $P521 = 2^{521} - 1$ discovered by Robinson in 1954 [44] and used as a standard for elliptic curve cryptography at the 256-bits security level. $2^{127}-1$ is also used for hyperelliptic curve cryptography at the 128-bits security level [12, 42] for efficiency reasons.

**Remark 1** *Using primes of the form $2^e+1$ provides an equivalent reduction algorithm. However there is no such primes for classical cryptographic sizes.*

### 1.1.2 Pseudo-Mersenne primes

Pseudo-Mersenne numbers were introduced in order to improve modular reduction as a generalization of Mersenne primes. They have the form $p = 2^e - c$ with usually $0 < c < 2^{\frac{e}{2}}$. The reduction of an integer $a < 2^e p$ modulo $p$ follows the Algorithm 2 which essentially consists in replacing two times $2^e$ by $c$ in the input. The costs of the steps of this reduction algorithm

**Algorithm 2:** Pseudo-Mersenne Reduction

```
Data: $p = 2^e - c$ with $0 < c < 2^{\frac{e}{2}}$
Input: $0 \leq a < 2^e p$
Result: $r = a \bmod p$ and $0 \leq r < p$
    Write $a = a_1 2^e + a_0$                    // $a_0 < 2^e$ and $a_1 < p$
    $b \leftarrow a_1 \times c + a_0$              // $b \leq 2^{\frac{3e}{2}}$
    Write $b = b_1 2^e + b_0$                    // $b_0 < 2^e$ and $b_1 \leq 2^{\frac{e}{2}}$
    $r = b_1 \times c + b_0$                     // $r < 2.2^e < 3p$
    $r' \leftarrow r + c$                        // $r' \leftarrow r - p + 2^e$
    if $r' \geq 2^e$ then                        // else $r < p$
        $r \leftarrow r' - 2^e$                  // $r \leftarrow r - p$ if $r \geq p$
        $r' \leftarrow r + c$                    // repeat the
        if $r' \geq 2^e$ then                    // previous step
            $r \leftarrow r' - 2^e$              // if $r \geq p$
return $r$
```

are:

- one multiplication of a $e$-bits number ($a_1$) with a $\frac{e}{2}$ one ($c$), denoted $M_{e,\frac{e}{2}}$,
- one multiplication of 2 $\frac{e}{2}$-bits number ($b_1$ and $c$) denoted $M_{\frac{e}{2}}$,
- 2 additions of $e$-bits numbers ($a_0$, $b_0$), denoted $A_e$,
- at worst 2 additions of $c$ which has at most $\frac{e}{2}$ bits.

So the overall cost of Algorithm 2 is at most

$$M_{e,\frac{e}{2}} + M_{\frac{e}{2}} + 2A_e + 2A_{\frac{e}{2}}.$$

**Remark 2** *In this paper, complexities for additions are given in terms of the smallest number involved. In other words, we do not take into account carry propagation. We made this choice because it is more representative of the complexity in average and it has no consequence on our results which are relative ones (long carry propagation may exceptionally occur in our methods as well as in any other one).*

The value of $c$ plays of course a central role in the complexity of Algorithm 2. Then, in 1992, Crandall suggested pseudo-Mersenne primes such that $c$ is small (in the sense that it fits on a machine word) [21]. The most known for cryptographic applications is probably $2^{255}-19$ that has been used by Bernstein to define the elliptic curve Curve25519 [11].

In 1999, Solinas from NSA published a report on generalized Mersenne numbers [46] to counter Crandall's patent. These primes are such that $p = f(2^k)$ where $f(t) = t^d - c_1 t^{d-1} - \cdots - c_d$ with $|c_i|$ small integers ($|c_i| << 2^k$) or null. The polynomial $f(t)$ should be irreducible [39] and $|c_d|$ odd. The goal is to get simple multiplications by $c$ in Algorithm 2 using sparse values for $c$ instead of small ones.
Solinas numbers are suggested by the NIST[1] and included in many standards for elliptic curve cryptography. For example $p_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ is used for the 128-bits security level. Note that, contrary to the Solinas primes suggested by the NIST for other security levels, $c > 2^{\frac{e}{2}}$ for $p_{256}$, so more rounds are needed in Algorithm 2.

## 1.2 Use of Montgomery arithmetic

The Montgomery reduction [38] is used when the modulo does not have a special form allowing fast reduction. This algorithm is regularly used in cryptography, like for RSA [43], or all the approaches based on finite field arithmetic like Elliptic Curve Cryptography [35, 37], pairing-based protocols [32] or more recently isogeny key exchanges [31].

### 1.2.1 Principle of Montgomery reduction

The main idea is to replace divisions by $p$ by much simpler divisions by a power of the radix, in other words to replace divisions by shifts. For example, with radix 2, if $p$ is a $e$-bits number, the reduction $\mathrm{Red}(a)$ of $a < 2^e p$ involves 2 steps:

- $q = -ap^{-1} \bmod 2^e$
- $r = (a + qp)/2^e$

It is then easy to prove that $r < 2p$ and $r = a2^{-e} \bmod p$ which is not exactly the expected reduction of $a \bmod p$. Of course, we can subtract $p$ as we did for Mersenne primes to get $r < p$. On the other hand, the fact that $r = a2^{-e} \bmod p$ instead of $a$ can be handled using the so-called Montgomery representation of numbers defined by $\overline{x} = x\,2^e \bmod p$ for any $0 \le x < p$.
This representation is indeed stable for the addition ($\overline{x} + \overline{y} = \overline{x + y}$) but also for the multiplication with the Montgomery modular reduction Red:

$$\mathrm{Red}(\overline{x}\,\overline{y}) = \overline{x}\,\overline{y}\,2^{-e} \bmod p = x\,y\,2^e \bmod p = \overline{x\,y} \bmod p.$$

The Montgomery representation of any $x \in [0, p[$ can be easily obtained by $\overline{x} = \mathrm{Red}(x\,y)$ if $y = 2^{2e} \bmod p$. Reciprocally, the value of $x \bmod p$ can be recovered with $x = \mathrm{Red}(\overline{x})$.

### 1.2.2 Word version of the Montgomery reduction and complexity

We did not give a precise general algorithm for the Montgomery reduction in the previous section because it is mainly used in its word version. We assume in the following that $p$ fits on $n$ $w$-bits words and denote $\beta = 2^w$. The word version of the Montgomery reduction of any $a < p\beta^n$ modulo $p < \beta^n$ ([38, 13]) is Algorithm 3.
Each step of the **for** loop of this algorithm requires

- 1 word multiplication $M_w$ (to compute $\mu r_0$),
- 1 multiplication of a word ($q$) by a $n$-words number ($p$) for computing $r$. Such a multiplication requires of course $nM_w$ but also $(n-1)A_w$,
- 1 addition with a $(n+1)$-words number ($qp$), namely $(n+1)A_w$.

[1] `https://csrc.nist.gov/publications/detail/fips/186/4/final`

**Algorithm 3:** Word version of the Montgomery reduction

```
Data:
  • A prime $p < \beta^n$ ($\beta$ is the word size)
  • The precomputed value $\mu = -p^{-1} \bmod \beta$
Input: $0 \leq a < p\beta^n$
Result: $r = a\beta^{-n} \bmod p$ and $0 \leq r < p$
  $r \leftarrow a$
  for $i = 0$ to $n - 1$ do
      $r_0 \leftarrow r \bmod \beta$                    // word truncation
      $q \leftarrow \mu \times r_0 \bmod \beta$
      $r \leftarrow (r + q \times p)/\beta$             // $r + qp$ multiple of $\beta$
  end for
  $r' \leftarrow r - p + \beta^n$
  if $r' \geq \beta^n$ then                           // else $r < p$
      $r \leftarrow r' \bmod \beta^n$                   // $r \leftarrow r - p$ if $r \geq p$
  return $r$
```

So, the **for** loop requires $(n^2 + n)M_w + 2n^2 A_w$. Finally, one addition of $\beta^n - p$ is required to ensure that $r < p$. The overall complexity of Algorithm 3 is then

$$(n^2 + n)M_w + (2n^2 + n)A_w$$

**Remark 3** *There also exists an interleaved version of the modular multiplication ([38, 13]) that has some advantages (smaller intermediate results) and some drawbacks (only compatible with schoolbook multiplication). We do not give it here because we are only interested in the reduction step but it can be used with all the Montgomery-like reduction algorithms given in this paper (as long as Karatsuba like technique is not used).*

## 1.3 Use of RNS arithmetic

### 1.3.1 Principle of RNS arithmetic

The Residue Number Systems have their origin in the Chinese Remainder Theorem [24]. We consider a set of co-prime numbers $\{m_1, \ldots, m_n\}$ and $M = \prod_{i=1}^{n} m_i$. Then, an integer $0 \leq a < M$ is fully defined by the set $(a_1, \ldots, a_n)$ with $a_i = a \bmod m_i$ for $i \in \{1, \ldots, n\}$. In practice, the $m_i$ would fit on one or two machine word.
Additions and multiplications modulo $M$ can be performed on each modulo independently:

$$a + b \bmod M = (a_1 + b_1 \bmod m_1, \ldots, a_n + b_n \bmod m_n)$$

$$ab \bmod M = (a_1 b_1 \bmod m_1, \ldots, a_n b_n \bmod m_n)$$

### 1.3.2 RNS Montgomery reduction

When we want to speed up computations in cryptography using RNS, we need to support modular reduction operations related to the finite ring or finite field on which the cryptographic system is defined. This is the case of many cryptographic approaches: RSA [8], ECC [2] or Pairing [17], Homomorphic Encryption [7, 26]... Thus, we consider arithmetics over finite rings like $\mathbb{Z}/p\mathbb{Z}$ (or finite fields if $p$ prime), where $p$ is a huge number, as it occurs in cryptography. In general $p$ is not the product of small numbers, thus the reduction modulo $p$ must use a specific algorithm like the modular reduction algorithm of Montgomery given in Section 1.2.1. Indeed,

this algorithm deals with the least significant digits of the handled values, thus its adaptation to RNS is natural [41, 5, 33].
The main idea, as the set $\mathcal{B} = \{m_1, \ldots, m_n\}$ represents the RNS base, is to use $M = \prod_{i=1}^{n} m_i$ as the Montgomery factor of reduction instead of $2^e$.
If we consider $0 \le a < pM$, the modular reduction can be done with a Montgomery approach assuming $2p < M$ and the use of a second base $\mathcal{B}' = \{m'_1, \ldots, m'_n\}$ such that $2p < M' = \prod_{i=1}^{n} m'_i$. This second base is necessary, because we must divide by $M$ (or multiply by $M^{-1}$) which cannot be done modulo $M$. This algorithm works in four main steps:

```
Algorithm 4: RNS Montgomery Reduction
  Data:
    • $\mathcal{B} = \{m_1, \ldots, m_n\}$ and $\mathcal{B}' = \{m'_1, \ldots, m'_n\}$ 2 RNS
      bases s.t. $2p < M = \prod_{i=1}^{n} m_i$ and $2p < M' = \prod_{i=1}^{n} m'_i$
    • $\mu = -p^{-1} \bmod M$ precomputed in $\mathcal{B}$
  Input: $0 \le a < pM$ given in $\mathcal{B}$ and $\mathcal{B}'$
  Result: $r = aM^{-1} \bmod p$ and $r < 2p$
    $q \leftarrow a \times \mu \bmod M$, in $\mathcal{B}$
    conversion of $q$ from $\mathcal{B}$ to $\mathcal{B}'$
    $r \leftarrow (a + q \times p) \times M^{-1} \bmod M'$, in $\mathcal{B}'$
    conversion of $r$ from $\mathcal{B}'$ to $\mathcal{B}$
  return $r$ given in $\mathcal{B}$ and $\mathcal{B}'$
```

As it was already the case for the classical Montgomery reduction, the result of Algorithm 4 is not exactly $a \bmod p$ (it is multiplied by the constant value $M^{-1} \bmod p$), but the result is reduced ($r < 2p$), and $r \bmod M' = r \bmod M = r$ because $2p < M, M'$. Again, this can be handled using the Montgomery representation defined in this case by $\widehat{a} = aM \bmod p$ and $\widehat{a} < 2p$. It can of course also be easily obtained by applying Algorithm 4 to $aM^2$ and the value of $a \bmod p$ can be recovered by applying Algorithm 4 to $\widehat{a}$.

**Remark 4** *When we want to perform sequences of operations, we need at least $4p^2 < MM'$ to assume the representation of the product of two values smaller than $2p$. The condition $2p < M$ then becomes $4p < M$ so that a product of 2 numbers smaller than $2p$ followed by a Montgomery reduction will give a result smaller than $2p$. Of course operations should be done in Montgomery representation because it is stable for the addition and for the multiplication followed by a Montgomery Reduction.*

### 1.3.3 RNS bases and conversions

General pseudo-Mersenne numbers are massively used in RNS arithmetic. For example, Kawamura et al [33] use bases of $n$ moduli $m_i, = 2^w - c_i$ ($w$ represents the number of bits of the basic words) and $c_i < 2^{w/2}$. In fact this approach works under a condition which is roughly speaking $c_i < \frac{2^w}{n}$, but they use $c_i < 2^{w/2}$ due to the internal modular reduction for each element of the RNS base. In [9], a double Montgomery reduction is suggested to avoid internal modular reduction constraint and the use of pseudo-Mersenne. In 2017, J. van der Hoven suggests in [29] to use *s-gentle moduli*, for example for $s = 2$, $m_i = 2^{2w} - \epsilon_i^2$ with $0 \le \epsilon_i < 2^{(w-1)/2}$.
To convert a RNS representation to a classical representation or to another RNS base, most of the approaches use a Lagrange interpolation:
if $0 \le X < M = \prod_{i=1}^{n} m_i$, and $(x_1, \ldots, x_n)$ is the RNS representation of $X$ in $\mathcal{B} = \{m_1, \ldots, m_n\}$,

then,

$$X = \sum_{i=1}^{n} \left| x_i M_i^{-1} \right|_{m_i} M_i - \alpha M \text{ where } M_i = \frac{M}{m_i}.$$

We note $|.|_{m_i}$ the value modulo $m_i$.
For [33], the challenge is to compute $\alpha$. They show that the first conversion in Algorithm 4 can be an approximation as $q$ is then multiply by $p$ that does not affect the result modulo $p$. The second conversion needs to be exact, that is done if $w < 2p < (1 - \frac{1}{\rho})M'$ for $\rho \geq 2$ and $\frac{c_i}{2^w} < \frac{1}{n}(1 - \frac{1}{\rho})$. In [9] the conditions are a little bit relaxed, but the philosophy stays the same. In [29], the complexity is studied asymptotically so the author uses a binary tree construction with a logarithmic depth with a modulo reduction at each level.
In Section 5 we introduce a new alternative of RNS bases using primes of the form $p = 2^{e_2}\alpha \mp 1$ and which are well adapted to preconfigured words architectures (CPU, GPU, FPGA,...).

## 2 Primes of the form $p = 2^{e_2}\alpha \mp 1$

In this paper, we focus on so-called Montgomery-friendly prime numbers [27, 14, 15], namely having the form $p = 2^{e_2}\alpha - 1$ with $e_2$ larger than the word size $w$. Similar results are obtained for primes of the form $2^{e_2}\alpha + 1$.

**Remark 5** *These prime numbers do not have significantly better reduction algorithm than the ones of Section 1 but they are attenuating their main default by providing more choices of primes suitable for cryptographic applications.*

### 2.1 General reduction algorithm

The main interest of such primes is their nice behaviour with the word version of the Montgomery reduction (Algorithm 3) because $-p^{-1} = 1 \bmod 2^w$ (because $w \leq e_2$), so that the parameter $\mu$ equals 1.Then, the only operation to perform in the **for** loop is the computation of $(r + qp)/\beta$ where $\beta = 2^w$ and $q$ is simply $r \bmod \beta$. This can be simplified if $p$ is written as $\beta 2^{e_2 - w}\alpha - 1$. Indeed, assuming that $r = \sum r_i \beta^i$ at the beginning of each step of the for loop, then:

$$\begin{aligned} (r + qp)/\beta &= \left(\sum r_i \beta^i + r_0 \beta 2^{e_2 - w}\alpha - r_0\right)/\beta \\ &= \sum_{i \neq 0} r_i \beta^{i-1} + r_0 2^{e_2 - w}\alpha \end{aligned}$$

So the cost of each step of the for loop is reduced to one multiplication of a single word by $\alpha 2^{e_2 \bmod w}$, a shift of $e_2$ div $w$ words which is for free and the addition of the result of this multiplication. The final addition requires $n$ word additions. Assuming $\alpha 2^{e_2 \bmod w}$ fits on $n_\alpha$ words, this gives the overall complexity for Algorithm 5:

$$n n_\alpha M_w + n\left(2n_\alpha + 1\right) A_w$$

**Remark 6** *We get the same result if* $p = 2^{e_2}\alpha + 1$. *The only difference is that Algorithm 5 involves* $-r_0\alpha 2^{e_2 - w}$ *instead of* $+r_0\alpha 2^{e_2 - w}$

### 2.2 Choice of $e_2$ and $\alpha$

The complexity of Algorithm 5 is dominated by multiplications by $\alpha$. As a consequence, when it is possible, good choices of $\alpha$ allow to improve the complexity of the reduction modulo prime

**Algorithm 5:** Word version of the Montgomery reduction if $p = 2^{e_2}\alpha - 1$

```
**Data**: $p < \beta^n$ with $\beta = 2^w$ and $w \leq e_2$
**Input:** $0 \leq a < p\beta^n$
**Result**: $r = a\beta^{-n} \bmod p$ and $0 \leq r < p$
    $r \leftarrow a$
    **for** $i = 0$ **to** $n - 1$ **do**
        $r_0 \leftarrow r \bmod \beta$
        $r \leftarrow (r - r_0)/\beta + r_0 \times \alpha 2^{e_2 - w}$
    **end for**
    $r' \leftarrow r + (\beta^n - p)$
    **if** $r' \geq \beta^n$ **then**
        $r \leftarrow r' - \beta^n$
    **return** $r$
```

numbers $p$ of the form $2^{e_2}\alpha \mp 1$. The best possible choice for $\alpha$ is of course 1 but $p$ would then be a Mersenne prime. In order to get more candidates, we can choose small values of $\alpha$ or use the same process than in Section 1, namely considering $\alpha$ of the form $2^{e'_2} - c$ with $c$ small (as an equivalent of pseudo-Mersenne primes) or sparse involving only coefficients of the word size (as an equivalent of the Solinas primes).
The other way to optimize the complexity of Algorithm 5 is to assume that $e_2$ (and $e'_2$) is a multiple of the word size. This may reduce the word size $n_\alpha$ to the one of $\alpha$.
If both $\alpha$ and $e_2$ can be wisely chosen as above, the cost of each step of the for loop is only one multiplication of a single word by $c$ and the addition of the result of this multiplication. The best choice for $c$ is of course $\pm 1$. However this case is similar to Mersenne primes in the sense that such primes are very rare for real world cryptographic applications. For example, if $w = 32$, there are only 4 of them in the cryptographic range for elliptic and hyperelliptic curve cryptography:

- $p_{192} = 2^{64}(2^{128} - 1) - 1$, used in most standards,
- $p_{448} = 2^{224}(2^{224} - 1) - 1$, used for the curve Ed448 [28],
- $p_{480} = 2^{448}(2^{32} - 1) - 1$,
- $p_{512} = 2^{32}(2^{480} - 1) - 1$,

**Remark 7** *There are all for high security levels and only $p_{192}$ can be used if $w = 64$. Considering primes of the form $2^{e_2}(2^{e'_2} - 1) + 1$ provides 7 additional primes but only one can be used if $w = 64$ and it is 576-bits long.*

If $c$ is assumed to be small, typically if it fits on a word (of size $w$), Algorithm 5 requires only one word multiplication ($r_0c$) as well as 3 word additions (one for adding $r_0 2^{e'_2}$ to $(r - r_0)/\beta$ and 2 for adding $r_0 c 2^{e_2 - w}$ which fits on two words because $e_2 - w$ is a multiple of the word size. The final addition is also simpler than in the general case, because $\beta^n - p = c2^{e_2} + 1$ so that computing $r'$ requires only 2 word additions.
Consequently, if $p = 2^{e_2}(2^{e'_2} - c) - 1$ where $c$ fits on one word and both $e_2$ and $e'_2$ are multiples of the word size $w$, the complexity of Algorithm 5 becomes

$$nM_w + (3n + 2)A_w.$$

Of course, if $c$ is sufficiently small, the $M_w$ can be replaced by some $A_w$.

Let us now assume that $c$ is written in the form $\sum c_i 2^{wi}$ (in the spirit of Solinas primes). The $c_i$ can be chosen of any sign and size, but they should be very small in practice for efficiency reasons. In this case, multiplying $r_0$ by $c$ and adding the result to $(r - r_0)/\beta$ requires at most $\sum |c_i|$ word additions. One more word addition is necessary to add $r_0 2^{e_2'}$. As in the previous case, the final addition is simplified by the form of $\beta^n - p$ and computing $r'$ requires only $1 + \#\{c_i\}_{c_i \neq 0}$ word additions. Then the complexity of Algorithm 5 in this case is at most

$$\left(n + 1 + n \sum |c_i| + \#\{c_i\}_{c_i \neq 0}\right) A_w.$$

## 2.3 The case $e_2 \geq \log_2(p)/2$

In this section, we assume that $e_2 \geq e/2$ where $p$ is $e$-bits long and that $e_2$ is a multiple of the word size, which is $n - n_\alpha$ if $p$ (resp. $\alpha$) fits on $n$ (resp. $n_\alpha$) words. We can then introduce a new version of the Montgomery reduction that is intermediate between the word version and the general case and that is still optimized in the sense that the parameter $\mu$ involved is $\pm 1$. The idea is to perform 2 Montgomery reduction steps modulo $2^{e_2}$ (instead of one step modulo $2^e$ in the general case or $n$ steps modulo $\beta = 2^w$ in the word version) using that $\mu = -p^{-1} = \pm 1 \bmod 2^{e_2}$. More precisely, the first step of such a reduction of some $a < 2^e p \leq 2^{2e_2} p$ will be

1. $q_0 = a(-p^{-1}) \bmod 2^{e_2} = \pm a \bmod 2^{e_2}$
2. $r_0 = (a + q_0 p)/2^{e_2} = (a - a \bmod 2^{e_2})/2^{e_2} + q_0 \alpha$

It is then easy to prove that $r_0$ has been reduced compared to $a$ but it is only smaller than $2^{e_2} p$ so that a second Montgomery reduction step is necessary to get a full reduction of $a$. We then get Algorithm 6 for the modular reduction in this case.

**Algorithm 6:** Intermediate Montgomery reduction for $p = 2^{e_2}\alpha - 1$ with $e_2 \geq \log_2(p)/2$

```
Data: A $e$-bits long prime $p = 2^{e_2}\alpha - 1$ s.t. $e_2 \geq e/2$
Input: $0 \leq a < 2^e p$
Result: $r_1 = a 2^{-2e_2} \bmod p$ and $0 \leq r_1 < p$
    $q_0 \leftarrow a \bmod 2^{e_2}$
    $r_0 \leftarrow (a - q_0)/2^{e_2} + q_0 \times \alpha$                // first reduction
    $q_1 \leftarrow r_0 \bmod 2^{e_2}$
    $r_1 \leftarrow (r_0 - q_1)/2^{e_2} + q_1 \times \alpha$               // second reduction
    $r_1' \leftarrow r_1 - p + 2^e$
    if $r_1' \geq 2^e$ then
        $r_1 \leftarrow r_1' \bmod 2^e$                                   // $r \leftarrow r - p$ if $r \geq p$
    return $r_1$
```

It is obvious that the output of this algorithm is $a 2^{-2e_2} \bmod p$ as it was already the case in classical Montgomery reduction. Moreover, it is less than $p$ because at the end of the 2 reduction steps

$$\begin{aligned} r_1 &= \frac{\frac{(a+q_0 p)}{2^{e_2}} + q_1 p}{2^{e_2}} = \frac{a + (q_0 + q_1 2^{e_2})p}{2^{2e_2}} \\ &< \frac{2^{2e_2} p + (2^{2e_2} - 1)p}{2^{2e_2}} < 2p \end{aligned}$$

The operations involved in Algorithm 6 are the following ones:

- $a \bmod 2^{e_2}$ and $(a - q_0)/2^{e_2}$ is a simple truncation which is for free because $e_2$ is assumed to be a multiple of the word size,

- $q_0\alpha$ and $q_1\alpha$ are 2 multiplications of $n - n_\alpha$ by $n_\alpha$ words numbers,
- 2 additions of two $n$-words numbers ($q_0\alpha$ and $q_1\alpha$),
- a final addition of $2^e - p$ which is less than $n$-words long.

This algorithm is of course optimal when $2^{e_2}$ fits on $n/2$ words, namely $n_\alpha = n/2$. In this case, its complexity is the same as the one of Algorithm 5 if schoolbook method is used for computing the $q_i\alpha$. But thanks to this approach, alternative multiplication algorithms like Karatsuba's one can be used to reduce the global complexity of Algorithm 6. Of course, this will only be interesting if $\alpha$ cannot be chosen with small coefficients.

**Remark 8** *We get the same result if $p = 2^{e_2}\alpha + 1$. The only difference is that Algorithm 6 involves $-q_i\alpha$ instead of $+q_i\alpha$.*

## 2.4 Comparison with other methods

The complexities we obtain have the same order than the one for pseudo-Mersenne or Solinas primes. We will give more precise comparison for each application in the following. But the main interest of this new form of primes is that we get new prime numbers having an efficient reduction algorithm that can be used when pseudo-Mersenne or Solinas primes are too rare. The situation is in fact even better with the new prime numbers we introduced because, contrary to pseudo-Mersenne primes, the value of $c$ can be even in our case. As a consequence, for the same size of primes we will be able to find roughly twice more candidates. This will for example be very interesting for generating RNS bases as we will see in Section 5.

# 3 New Primes for Isogeny-based cryptography

Isogeny-based protocols such as SIDH are based on a shared secret which is the $j$-invariant of some elliptic curve. They have been introduced in cryptography by Couveignes in 1997 [20] and have been recently developed as an alternative for quantum resistant cryptography [30, 19, 18, 36]. SIKE is now a second-round candidate for the NIST post-quantum cryptography standardization process [40]. We will present it here briefly in a simplified form.

## 3.1 The SIKE protocol

The public parameters are

- a prime number $p = 2^{e_2}3^{e_3} - 1$ such that $2^{e_2} \simeq 3^{e_3}$,
- a supersingular elliptic curve $E_0$ defined over $\mathbb{F}_{p^2}$,
- two points of order $2^{e_2}$, $P_2 \in E_0\left(\mathbb{F}_{p^2}\right) \backslash E_0\left(\mathbb{F}_p\right)$ and $Q_2 \in E_0\left(\mathbb{F}_p\right)$,
- two points of order $3^{e_3}$, $P_3 \in E_0\left(\mathbb{F}_{p^2}\right) \backslash E_0\left(\mathbb{F}_p\right)$ and $Q_3 \in E_0\left(\mathbb{F}_p\right)$.

The points $P_2$ and $Q_2$ (resp. $P_3$ and $Q_3$) then form a base for the full $2^{e_2}$ (resp. $3^{e_3}$) torsion of $E_0$. The key exchange process is as follows

- Alice chooses $\ell = 2$ or $3$ (let $m$ be the other one) and a secret key $\mathrm{sk}_\ell < \ell^{e_\ell}$. She computes a point $S_\ell = P_\ell + \mathrm{sk}_\ell\, Q_\ell \in E_0\left(\mathbb{F}_{p^2}\right)$ which defines an isogeny $\phi_\ell$ of degree $\ell^{e_\ell}$ as its kernel. This isogeny can be easily computed as a sequence of $\ell$-isogenies and Alice's public key is given by $\phi_\ell(P_m)$ and $\phi_\ell(Q_m)$. The elliptic curve $E_\ell = \phi_\ell\left(E_0\right)$ is also part of the public key but it is deduced from $\phi_\ell(P_m)$, $\phi_\ell(Q_m)$ and $\phi_\ell(P_m - Q_m)$ in practice.

- Bob follows the same process exchanging $\ell$ and $m$. His public key is then $E_m$, $\phi_m(P_\ell)$ and $\phi_m(Q_\ell)$.
- Alice repeats the process with $E_m$ and $S'_\ell = \phi_m(P_\ell)+\mathrm{sk}_\ell\phi_m(Q_\ell)$. She then gets an isogeny $\phi'_\ell$ with kernel generated by $S'_\ell = \phi_m(S_\ell)$ so that $\phi'_\ell \circ \phi_m$ has kernel generated by $S_\ell$ and $S_m$.
- Again, Bob do the same exchanging $\ell$ and $m$ and gets $\phi'_m$ such that $\phi'_m \circ \phi_\ell$ has the same kernel.
- Allice and Bob can then both compute their share secret, the $j$-invariant of the target elliptic curve of $\phi'_\ell \circ \phi_m$ and $\phi'_m \circ \phi_\ell$ which are the same.

Most of the computations involved are then elliptic curves operations and 2 or 3-isogenies computation over $\mathbb{F}_p$ or $\mathbb{F}_{p^2}$. Modular reductions modulo $p$ then have a key role in the computational cost of the SIKE protocol.
The computational security of this protocol is based on the degrees of the isogenies involved and then on $2^{e_2}$ and $3^{e_3}$. Known classical attacks are in $O\left(\sqrt{\ell^{e_\ell}}\right)$ and quantum once are in $O\left(\sqrt[3]{\ell^{e_\ell}}\right)$. In order to achieve a close balance on both sides of the protocol, $e_2$ and $e_3$ are chosen such that $2^{e_2} \simeq 3^{e_3}$.

## 3.2 Primes of the form $2^{e_2}3^{e_3} - 1$

According to the constraints above, the original SIKE proposal [4] recommended to use respectively the prime numbers p503, p751 and p964 (given in Table 1) to meet the security requirements of NIST categories 1, 3 and 5. However, it has been recently shown that the hardness of computing isogenies on quantum computers were overestimated [1]. As a consequence the second round proposal of SIKE [3] provides new primes p434 and p610 suitable for NIST categories 1 and 3. Note that the reference implementation of SIKE (that we will use for comparisons) only uses p503 and p751.

| prime | $e_2$ | $e_3$ | $\log_2(3^{e_3})$ | security category | symmetric key size |
|---|---|---|---|---|---|
| p434 | 216 | 137 | 218 | 1 | 128 bits |
| p503 | 250 | 159 | 253 | 2 | |
| p610 | 305 | 192 | 305 | 3 | 192 bits |
| p751 | 372 | 239 | 379 | 4 | |
| p964 | 486 | 301 | 478 | 5 | 256 bits |

Table 1: Primes of the SIKE NIST proposals [4, 3]

Reductions are done using the word version of the Montgomery reduction algorithm (or its interleaved version, see Remark 3) for this specific form of primes, namely Algorithm 5 with $\alpha = 3^{e_3}$. Assuming $p$ fits on $n$ $w$-bits words, and that $\alpha 2^{e_2 \bmod w}$ fits on $n_\alpha$ words, the cost of the reduction is then

$$nn_\alpha M_w + n\left(2n_\alpha + 1\right) A_w$$

**Remark 9** *Remember that we did not consider the carry propagation cost for simplicity but also because it will not significantly change the cost evaluation and it will occur in the same way in our proposal. Then this assumption will not affect the results of this work.*

| prime | | $\min(e_2, \log_2(3^{e_3}))$ | security category | symmetric key size |
|---|---|---|---|---|
| p448 | $262.2^{224}\,3^{136} - 1$ | 216 | 1 | 128 bits |
| p477 | $37.2^{256}\,3^{136} - 1$ | 216 | 1 | |
| p512 | $31.2^{256}\,3^{158} - 1$ | 251 | 2 | |
| p630 | $39.2^{320}\,3^{192} - 1$ | 305 | 3 | 192 bits |
| p765 | $214.2^{384}\,3^{235} - 1$ | 373 | 4 | |
| p996 | $69.2^{512}\,3^{301} - 1$ | 478 | 5 | 256 bits |

Table 2: New primes proposed for SIKE-like protocols

For example, for $p_{503}$ we have $n = 8$ if $w = 64$ and $\alpha = 3^{159}2^{58}$ so that $n_\alpha = 5$ and the overall cost of the reduction algorithm is then $40M_{64} + 88A_{64}$ (which fits with the reference implementation of SIKE). The complexities for reducing modulo the SIKE primes are given in Table 3.
We explained in Section 2.2 that it would be interesting to choose $e_2$ to be a multiple of the word-size in order to reduce the cost of Algorithm 5. This is not the case for the primes given in Table 1. That is the reason why we propose here new primes for the SIKE protocol satisfying this property.

## 3.3 New primes proposed

The only condition on the form of $p$ for SIKE-like protocols is that $p+1$ is divisible by sufficiently large powers of 2 and 3 which ensures the existence of the isogenies $\phi_2$ and $\phi_3$. It is then not restrictive to consider primes $p$ with a small cofactor $f$, namely of the form $p = f.2^{e_2}3^{e_3} - 1$. This is besides mentioned in the SIKE proposals [4, 3] but discarded because they was enough candidates satisfying all the conditions with $f = 1$. For efficiency reasons, we consider here the new condition that $e_2$ is a multiple of the word size. This is of course quite restrictive so we do need to consider non-trivial cofactors in this case. We give in Table 2 the new values we propose for SIKE primes such that $e_2$ is a multiple of 64 (so that $e_2$ is always a multiple of the word size in real world applications). These primes provide at least the same security level that the ones of Table 1 (because $\min(e_2, \log_2(3^{e_3}))$ is always larger in our case) and $p$ requires the same number of words. For the security category 1, considering 32-bits word may be more appropriate. Indeed 64-bits word implies that the $e_2$ and $e_3$ parts of $p$ both fit on 4 words while they can fit on 7 32-bits words. We then give the prime p448 for which $e_2$ is only a multiple of 32.
Because of our new constraint, both sides of the protocol are not balanced anymore at first glance. This can be solved by shortening the longest loop considering public points of lower order. For example, with p630, $2^{e_2}$ is 320-bits long while $3^{e_3}$ is only 305. So the 2-side of the protocol should have 320 steps while only 305 are necessary for this security level. In this case, we only have to choose $P_2$ and $Q_2$ having order $2^{305}$ instead of $2^{320}$ to shorten the 2-side loop. The security level remains the same because it was determined by the 3-side of the protocol.
The value of $\alpha$ is $f.3^{e_3}$ and is smaller than $2^{e_2}$ by construction. As a consequence, $e_2 \geq \log_2(p)/2$ so that Algorithm 6 can be used for reducing modulo these primes. Of course, Algorithm 5 can also be used with the same complexity if schoolbook method is used in Algorithm 6 because in any case, $p$ fits in $n$ words and $\alpha$ in $n/2$ words. This complexity is given by

$$\frac{n^2}{2}M_w + n(n+1)A_w.$$

| Symmetric key size | Security category | NIST primes | Complexity Algorithm 5 | New primes | Complexity: Algorithm 5 or 6 with Schoolbook | Complexity: Algorithm 6 with one Karatsuba step |
|---|---|---|---|---|---|---|
| 128 bits | 1 | p434 | $112\,M_{32}+238\,A_{32}$<br>$28\,M_{64}+63\,A_{64}$ | p448<br>p477 | $98\,M_{32}+196\,A_{32}$<br>$32\,M_{64}+72\,A_{64}$ | $82\,M_{32}+210\,A_{32}$<br>$24\,M_{64}+76\,A_{64}$ |
|  | 2 | p503 | $40\,M_{64}+88\,A_{64}$ | p512 | $32\,M_{64}+72\,A_{64}$ | $24\,M_{64}+76\,A_{64}$ |
| 192 bits | 3 | p610 | $60\,M_{64}+130\,A_{64}$ | p630 | $50\,M_{64}+110\,A_{64}$ | $44\,M_{64}+120\,A_{64}$ |
|  | 4 | p751 | $84\,M_{64}+180\,A_{64}$ | p765 | $72\,M_{64}+156\,A_{64}$ | $54\,M_{64}+150\,A_{64}$ |
| 256 bits | 5 | p964 | $144\,M_{64}+304\,A_{64}$ | p996 | $128\,M_{64}+272\,A_{64}$ | $96\,M_{64}+248\,A_{64}$ |

Table 3: Complexity comparison for modular reduction in SIKE-like protocols

For example, using $p_{512} = 31.2^{256}3^{158} - 1$ instead of $p_{503} = 2^{250}3^{159} - 1$ provides a reduction that costs $32M_{64} + 72A_{64}$ instead of $40M_{64} + 88A_{64}$. This is substantially better and will give a speed-up around 20% of the reference implementation of SIKE. Moreover, as explained in Section 2.3, contrary to Algorithm 5, large multiplications are directly involved in Algorithm 6, so that Karatsuba method can be used to compute $q_0 \times \alpha$ and $q_1 \times \alpha$. In our case, $\alpha$ and the $q_i$ are all $n/2$-words numbers. For example if we perform one Karatsuba step in the case of $p512$, the cost of $q_i \times \alpha$ is $3M_{128} + 7A_{128}$ instead of $4M_{128} + 4A_{128}$ so that only $24\,M_{64}$ are necessary instead of 32 (but of course, more additions).
We give in Table 3 a reduction complexity comparison between SIKE primes of Table 1 and our new proposals given in Table 2. This table clearly shows that the new primes we propose may have an great interest for efficient implementations of SIKE-like protocols.

**Remark 10** *Complexities involving Karatsuba should be considered carefully because practice and theory may differ. That is the reason why we did not give the complexities for all the possibilities of using this method in Table 3 (one could for example make more Karatsuba steps to get a better theoretical complexity). Note also that $\alpha$ is constant and used 2 times in Algorithm 6 so that some precomputations could be done to reduce the number of additions involved in a Karatsuba implementation of the multiplication by $\alpha$.*

## 4 Application to Elliptic curve cryptography

Because of their efficient reduction algorithm, particular forms of prime are used in elliptic curve cryptography for a long time. For example, Solinas primes are used in the FIPS 186-3 standards [22] and the prime used for Ed25519 curve is the pseudo-Mersenne prime $2^{255}-19$ [11]. Several authors already proposed to use Montgomery-friendly primes of the form $2^{e_2}(2^{e_2'}-c)\pm 1$ and provide elliptic curves defined over such prime fields which are suitable for cryptographic use [27, 15, 16]. They are for example recommending the primes $2^{240}(2^{16}-88)-1$ and $2^{240}(2^{14}-127)-1$ for the 128-bits security level. However these proposals are not satisfying what we identified in Section 2.2 as good choice for the parameters $e_2$ and $e_2'$. In particular there are not multiples of the word size. We propose in this paper new primes satisfying this condition and then providing a more efficient reduction step.
We will first focus on the 128-bits security level because it is the most commonly used in practice and we will then give the results we obtained for higher security levels.

### 4.1 The 128-bits security level

The first step is to find a prime $p$ of the form $2^{e_2}(2^{e_2'}-c)\pm 1$ such that multiplications by $c$ is as inexpensive as possible and such that $e_2$ and $e_2'$ are mutliples of the word size. We assume that

the word size is 64 to cover as many use cases as possible. For each security level, we made an exhaustive search of such $p$ with $c$ small or sparse.
The best result we get for the 128-bits security level is given by $e_2 = 192, e_2' = 64$ and $c = 4$

$$p_{256} = 2^{192}(2^{64} - 4) - 1$$

In this case, each of the 4 step of the `for` loop of Algorithm 5 consists in the computation of $\frac{r-r_0}{2^{64}} + r_0 2^{64} - 4r_0$ which requires only 4 word-additions. The final addition step of Algorithm 5 is the computation of $r - p + 2^{256} = r - 4.2^{192} + 1$ which is a very fast operation. Finally, the word version of the Montgomery reduction algorithm requires approximately $17A_{64}$. Algorithm 6 can be also used for this prime if one wants to use some Karatsuba steps into the reduction algorithm. Moreover, $p = 3 \bmod 4$ which allows efficient modular square root computations.
As a comparison, with the prime $2^{240}(2^{16} - 88) - 1$, each step of the `for` loop of Algorithm 5 would require $M_{64} + 2A_{64}$ because $\alpha$ fits on one word in this case. Again, the final addition can be simplified in the same way, so that the complexity of Algorithm 5 is approximately $4M_{64} + 9A_{64}$. As a consequence, assuming we are working on a plateform for which additions of 64-bits words are at least twice faster than multiplications, the reduction step is always faster with the prime $p_{256}$ introduced in this paper. Of course, this result is also valid for any platform using less than 64-bits words.
Reducing modulo $2^{240}(2^{14} - 127) - 1$ or $2^{255} - 19$ will also require some multiplications (so we get a similar result) but allow to avoid the final addition step.

**Remark 11** *Choosing $e_2$ and $e_2'$ multiples of the word size implies that the final addition of the Montgomery reduction step cannot be avoided assuming $4p < 2^{256}$ as it is the case for $2^{240}(2^{14} - 127) - 1$. But, the gain obtained thanks to our choice outreaches this additional addition, especially because it is not a full addition due to the form of $p$.*

The second step is to generate an elliptic curve satisfying all security requirements. We follow the process explained in [15] to generate parameters $A$ and $d_0$ defining the isogeneous elliptic curves in Montgomery and twisted Edwards forms

$$\begin{aligned} M_A &: \; y^2 = x^3 + Ax^2 + x \\ E_{d_0} &: \; -x + y^2 = 1 + d_0 x^2 y^2 \end{aligned}$$

These parameters are chosen such that the cardinalities of the curve and its twist are almost prime with the minimal cofactor (which is 4) in order to guarantee the expected security level both on the curve and its twist. For efficiency reasons, $A$ and $d_0$ have to be chosen as small as possible. It is explain in [15] that under some conditions (that will be satisfied in our cases), $A^2 - 4$ is not a square (which simplifies notions of completeness [11]) and $d_0$ can be chosen equal to $-(A+2)/4$. By the way, this value is the one involved in the Montgomery ladder formulas. So, in any case, it is necessary to select the parameter $A$ such that $(A+2)/4$ is as small as possible in order to improve the global efficiency. This is what is done in [15]. However, using Montgomery-friendly primes implies that the operands have to be in Montgomery representation. So the value that has to be small is not $(A + 2)/4$ or $d_0$ but its Montgomery representation $\overline{d_0} = -(A + 2)2^{256}/4 \bmod p$. Of course, we also checked that the curves obtained are satisfying the security requirements given in [10]. For the 128-bits security level, the smallest values we get for the Montgomery representation of $-(A + 2)/4$ are $\overline{d_0} = 5919$ and $-9869$ which both fit on 16 bits contrary to the values given in [15] or Curve25519. This may be an advantage for small devices.
Finally, using the prime $p_{256} = 2^{192}(2^{64} - 4) - 1$ for elliptic curve cryptography seems to be a better choice than the other ones given in the literature for efficiency reasons. This should of course be verified by a practical implementation but in any case, it provides an interesting alternative to Curve25519. More generally considering primes of the form $2^{e_2}(2^{e_2'} - c) \pm 1$ allows to get more choice for efficient elliptic curve cryptography than using only Bernstein-like curves.

### 4.2 Higher security levels

At the 192-bits security level, we found the prime

$$p_{384} = 2^{128}(2^{256} - 2^{192} + 2^{64} + 1) - 1$$

and the elliptic curve defined by the parameter $A$ or $d_0$ such that the Montgomery representation $\overline{d_0}$ of $-\frac{A+2}{4}$ equals $-10498.2^{128}$ which still fits on one (shifted) 16-bits word.
The prime $p = 2^{192}(2^{192} - 2^{128} + 2^{64} - 2) - 1$ could also be considered with $\overline{d_0} = 39737$. One more addition is required for each step of Algorithm 5 but, contrary to $p_{384}$, it is compatible with Algorithm 6.
Concerning the 256-bits security level, we found three primes ensuring the same complexity for Algorithm 5 and well suited for 64-bits architectures. One can for example choose

$$p_{512} = 2^{128}(2^{384} - 2^{128} + 2^{64} - 1) - 1$$

and the elliptic curve defined over $\mathbb{F}_{p_{512}}$ by the parameter $A$ or $d_0$ such that $\overline{d_0} = -60237$ which also fits on one 16-bits word. The two other primes are

- $2^{64}(2^{448} - 2^{192} - 2^{128} + 1) - 1$ and
- $2^{192}(2^{320} - 2^{128} + 2) - 1$

## 5 Alternative RNS bases with $m_i = 2^{e_2}(2^{e_2'} - c_i) \mp 1$

We consider in this section RNS bases made of co-prime numbers $\{m_1, \dots, m_n\}$ such that for $i \in \{1, \dots, n\}$, $m_i = 2^{e_2}(2^{e_2'} - c_i) \mp 1$ with $0 < c_i < 2^{e_2'}$ (when $c_i = 0$ then $m_i = 2^{e_2+e_2'} - 1$). In this case, the reduction modulo each $m_i$ uses Algorithm 6 for the internal modular reduction. If such number system is implemented on a word-architecture ($w-$bits unit) like GPU or FPGA (DSP-unit), a good choice is to have $e_2 = e_2' = w$ so that each $m_i$ is coded on two words to fit well Algorithm 6. We consider in this part that $e_2 - 1 \le e_2' \le e_2$ which seems the most adapted to our purpose, but a generalisation is always possible.

**RNS addition**
The addition of $a = \{a_1, \dots, a_n\}$ and $b = \{b_1, \dots, b_n\}$ is of course done by computing $r_i = a_i + b_i \bmod m_i$ for each modulo $m_i$. The reduction can then be obtained by computing $r_i' = r_i + (2^{e_2}c_i \pm 1)$: if $r_i' \ge 2^{e_2+e_2'}$ then $r_i$ is replaced by $r_i' \bmod 2^{e_2+e_2'}$.

**RNS multiplication**
The multiplication is also done component by component. But, in this case, a stronger modular reduction, like the one of Montgomery (Algorithm 6) must be applied. The result of each reduction will then be $r_i = a_i b_i 2^{-2e_2} \bmod m_i$ with $r_i < 2m_i$.

**Montgomery RNS representation**
Even at the moduli level it is then interesting to use a Montgomery representation: $\overline{a} = a2^{2e_2} \bmod M$, in other words $(\overline{a}_1, \dots, \overline{a}_n)$, with $\overline{a}_i = a_i 2^{2e_2} \bmod m_i$ for $i \in \{1, \dots, n\}$. We obviously have $\overline{a} + \overline{b} = \overline{a+b} \bmod M$ so the addition is stable for the Montgomery representation. But the multiplication is not.

**The operator $\otimes$**
We then introduce the operator $\otimes$ defined by $a \otimes b = ab2^{-2e_2}$. It is said stable for the Montgomery representation with reduction factor equals to $2^{-2e_2}$ because $\overline{a}\otimes\overline{b} = ab2^{2e_2} = \overline{ab} \bmod M$. The operator $\otimes$ also has the following properties modulo $M$ that can be used for switching between representations:

$$\left| \begin{array}{ll} a \otimes 2^{4e_2} & = \overline{a} \\ \overline{a} \otimes 1 & = a \\ a \otimes \overline{b} & = ab \end{array} \right.$$

## 5.1 Modular reduction modulo a given $p$ in RNS with bases of the form $2^{e_2}(2^{e'_2} - c_i) \mp 1$

In this section, we consider bases $\mathcal{B}$ and $\mathcal{B}'$ such that $m_i = 2^{e_2}(2^{e'_2} - c_i) \mp 1$ and $m'_i = 2^{e_2}(2^{e'_2} - c'_i) \mp 1$ with $0 < c_i, c'_i < 2^{e'_2}$ (we can have $c_i$ or $c'_i = 0$ then $m_i$ or $m'_i = 2^{e_2+e'_2} - 1$), and the Montgomery reduction modulo each element of the bases.

The global RNS Montgomery reduction algorithm is Algorithm 7. The only difference with Algorithm 4 is that the precomputed values are affected by the Montgomery representation on each modulo.

**Algorithm 7:** RNS Montgomery reduction with specific RNS bases

**Data**:
- 2 co-prime RNS bases $\mathcal{B} = (m_1, \dots, m_n)$ and $\mathcal{B}' = (m'_1, \dots, m'_n)$ s.t. $m_i = 2^{e_2}(2^{e'_2} - c_i) \mp 1$ and $m'_i = 2^{e_2}(2^{e'_2} - c'_i) \mp 1$ with $0 \le c_i, c'_i < 2^{e'_2}$
- a prime $p$ s.t. $2p < M = \prod m_i$ and $2p < M' = \prod m'_i$
- the precomputed values $\overline{\mu} = -p^{-1}\, 2^{2e_2} \bmod M$, $\overline{\lambda} = M^{-1}2^{2e_2} \bmod M'$ and $\overline{\gamma} = pM^{-1}2^{2e_2} \bmod M'$

**Input:** $a$ given in $\mathcal{B}$ and $\mathcal{B}'$ with $0 \le a < pM$
**Result**: $r = aM^{-1} \bmod p$ and $r < 2p$

```
    $q \leftarrow a \otimes \overline{\mu}$ in $\mathcal{B}$                 // $q = a\left(-p^{-1}\right) \bmod M$ (1)
    conversion of $q$ from $\mathcal{B}$ to $\mathcal{B}'$
    $r \leftarrow a \otimes \overline{\lambda} + q \otimes \overline{\gamma}$ in $\mathcal{B}'$     // $r = (a + qp)M^{-1} \bmod M'$ (2)
    conversion of $r$ from $\mathcal{B}'$ to $\mathcal{B}$
return $r$ in $\mathcal{B}$ and $\mathcal{B}'$
```

As we have, $a \otimes b = ab2^{-2e_2}$, then

$$\begin{aligned} q = a \otimes \overline{\mu} &= a\left(-p^{-1}\, 2^{2e_2}\right) 2^{-2e_2} \\ &= a\left(-p^{-1}\right) \bmod M \qquad (1) \\ r = a \otimes \overline{\lambda} &+ q \otimes \overline{\gamma} = aM^{-1} + qpM^{-1} \\ &= (a + qp)M^{-1} \bmod M'. \qquad (2) \end{aligned}$$

As $0 \le a < pM$, then $(a + qp) < 2pM$. Thanks to (1), $a + qp$ is a multiple of $M$, so $(a+qp)M^{-1} < 2p < M'$, and Equation (2) is in fact not true only modulo $M'$. As a consequence, the result of Algorithm 7 satisfies $r = aM^{-1} \bmod p$ and $r < 2p$.
In most cases, the input of Algorithm 7 will be given in Montgomery representation for the moduli $m_i$ and $m'_i$ as the result of a product of two values in Montgomery representation $\overline{a} = \overline{b} \otimes \overline{d} = bd2^{2e_2} \bmod MM'$ in $\mathcal{B}$ and $\mathcal{B}'$. If the input is $\overline{a}$ with $a < pM$, then Algorithm 7 will compute

$$\overline{q} = \overline{a} \otimes \overline{\mu} = \overline{a\mu} = a\left(-p^{-1}\right) 2^{2e_2} \bmod M \text{ in } \mathcal{B}$$

instead of $q = a\left(-p^{-1}\right) \bmod M$, thus we cannot ensure that $q < M$ and then that $(a+qp)\,M^{-1} < 2p$. Finally, the result will not be less than $2p$ as expected.
Hence, we must adapt this algorithm in order to have $q = a\left(-p^{-1}\right) \bmod M$. For this we will compute $q = \overline{a} \otimes \mu$ in $\mathcal{B}$ and then $\overline{r}$ with $\overline{r} \leftarrow \overline{a} \otimes \overline{\lambda} + q \otimes \overline{\overline{\gamma}} = \overline{(a+qp)\,M^{-1}}$ in $\mathcal{B}'$, and finally we obtain $r = (a+qp)\,M^{-1} \bmod M' < 2p$. This adaptation is presented in Algorithm 8.

**Algorithm 8:** RNS reduction with specific bases in Montgomery representation

```
Data:
  • 2 co-prime RNS bases $\mathcal{B} = (m_1, \dots, m_n)$ and $\mathcal{B}' = (m'_1, \dots, m'_n)$ s.t.
    $m_i = 2^{e_2}(2^{e'_2} - c_i) \mp 1$ and $m'_i = 2^{e_2}(2^{e'_2} - c'_i) \mp 1$ with $0 \le c_i, c'_i < 2^{e'_2}$
  • a prime $p$ s.t. $2p < M = \prod m_i$ and $2p < M' = \prod m'_i$
  • the precomputed values $\mu = -p^{-1} \bmod M$ $\overline{\lambda} = M^{-1}2^{2e_2} \bmod M'$ and
    $\overline{\overline{\gamma}} = pM^{-1}2^{4e_2} \bmod M'$
Input: $\overline{a} = a2^{2e_2}$ given in $\mathcal{B}$ and $\mathcal{B}'$ with $0 \le a < pM$
Result: $\overline{r} = r2^{2e_2}$ with $r = aM^{-1} \bmod p$ and $r < 2p$
    $q \leftarrow \overline{a} \otimes \mu$ in $\mathcal{B}$                              // $q = a(-p^{-1}) \bmod M$
    conversion of $q$ from $\mathcal{B}$ to $\mathcal{B}'$
    $\overline{r} \leftarrow \overline{a} \otimes \overline{\lambda} + q \otimes \overline{\overline{\gamma}}$ in $\mathcal{B}'$                  // $r = (a+qp)M^{-1} \bmod M'$
    conversion of $\overline{r}$ from $\mathcal{B}'$ to $\mathcal{B}$
return $\overline{r}$ in $\mathcal{B}$ and $\mathcal{B}'$
```

**Remark 12** *Two levels of Montgomery representations can occur, one due to the moduli operator $\otimes$ and one due to the RNS Montgomery reduction: we note $\overline{\widehat{a}} = (aM \bmod p)\,2^{2e_2} \bmod M$ this double level of Montgomery representation. As in the previous cases of Montgomery representation, $\overline{\widehat{a}}$ can be easily obtained: let $\tau = M^2 \bmod p$, if we compute $a \otimes \overline{\overline{\tau}} = \overline{a\tau}$ with $a\tau < pM$, then Algorithm 8 returns $\overline{\widehat{a}}$.*

**Remark 13** *As usual, if we consider a sequence of operations, the input data could be the product of two previous results lower than $2p$. Then, we need to have $4p^2 < pM$ (in other words $4p < M$ instead of $2p < M$) to ensure that the input is lower than $pM$.*

## 5.2 Bases conversion with specific RNS bases

Algorithms like the Cox-Rover approach [33] are dependent of the RNS bases elements seen as pseudo-Mersenne close to a power of two. So they will clearly not work directly with $m_i = 2^{e_2}(2^{e'_2} - c_i) \mp 1$.
Hence, we consider approaches like the one of [6] where the first conversion is a partial Chinese remainder Theorem method without a final reduction (only the summation is done). The second conversion is exact using an auxiliary modulo [45].
Let $0 \le X < M$ given by its residues $x_i$ modulo the $m_i$. The CRT construction gives

$$X = \sum_{i=1}^{n} \frac{\xi_i}{m_i} M - \alpha M \text{ with } \xi_i = \left| x_i M_i^{-1} \right|_{m_i}.$$

Thus, the integer part $\left\lfloor \sum_{i=1}^{n} \frac{\xi_i}{m_i} \right\rfloor$ equals $\alpha$. With our specific bases we can have an approach which looks like the Kawamura et al one [33] to recover $\alpha$.

### 5.2.1 A general approach

We develop $\left\lfloor \sum_{i=1}^{n} \frac{\xi_i}{m_i} \right\rfloor$ using $m_i = 2^{e_2}(2^{e'_2} - c_i) \mp 1$ ,

$$\begin{aligned} \sum_{i=1}^{n} \frac{\xi_i}{m_i} &= \sum_{i=1}^{n} \frac{\xi_i}{2^{e_2}(2^{e'_2} - c_i) \mp 1} \\ &= \sum_{i=1}^{n} \left( \frac{\xi_i}{2^{e_2+e'_2}} + \frac{\xi_i c_i 2^{e_2} \pm \xi_i}{2^{e_2+e'_2}\left(2^{e_2}(2^{e'_2} - c_i) \mp 1\right)} \right) \end{aligned}$$

And, because $0 \leq \xi_i < 2^{e_2}(2^{e_2} - c_i) \mp 1$ and $c_i = 0$ only when $m_i = 2^{e_2+e'_2} - 1$, we have

$$\begin{aligned} \sum_{i=1}^{n} \frac{\xi_i}{m_i} &< \sum_{i=1}^{n} \left( \frac{\xi_i}{2^{e_2+e'_2}} + \frac{c_i 2^{e_2} \pm 1}{2^{e_2+e'_2}} \right) \\ &< \sum_{i=1}^{n} \left( \frac{\xi_i}{2^{e_2+e'_2}} + \frac{c_i}{2^{e'_2}} + \frac{1}{2^{e_2+e'_2}} \right) \end{aligned}$$

Now, if we consider that $\sum_{i=1}^{n} \left( c_i + \frac{1}{2^{e_2}} \right) < 2^{e'_2}$, then

$$\sum_{i=1}^{n} \left( \frac{\xi_i}{2^{e_2+e'_2}} \right) < \sum_{i=1}^{n} \frac{\xi_i}{m_i} < \sum_{i=1}^{n} \left( \frac{\xi_i}{2^{e_2+e'_2}} \right) + 1$$

and we are in a case similar to Kawamura et al one [33]. We can conclude that, when $\sum_{i=1}^{n} \left( c_i + \frac{1}{2^{e_2}} \right) < 2^{e'_2}$,

$$\left\lfloor \sum_{i=1}^{n} \frac{\xi_i}{2^{e_2+e'_2}} \right\rfloor \leq \left\lfloor \sum_{i=1}^{n} \frac{\xi_i}{m_i} \right\rfloor \leq \left\lfloor \sum_{i=1}^{n} \frac{\xi_i}{2^{e_2+e'_2}} \right\rfloor + 1.$$

In Algorithm 7, the first conversion "of $q$ from $\mathcal{B}$ to $\mathcal{B}'$ " can then be done with an algorithm similar to the one of [33]. It will compute $q$ or $q + M$ in $\mathcal{B}'$. As this value is multiplied by $\overline{\gamma} = pM^{-1}2^{2e_2} \bmod M'$, that will not affect the result modulo $p$.

**Example 1** *We consider $2^{e_2} = 2^{e'_2} = 2^{16}$, for example in a case of a 16-bits architecture. With a trivial algorithm it is possible to construct a set of 180 pairwise 32-bits co-primes satisfying $\sum_{i=1}^{n} \left(c_i + \frac{1}{2^{e_2}}\right) < 2^{e'_2}$ with a maximal value of $c_i$ equal to $360 < \frac{2^{16}}{180} - \frac{1}{2^{16}}$ and we have $2^{5759} \leq \prod_{i=1}^{180} m_i < 2^{5760}$. This means that one can perform RNS arithmetic with p up to 2878 bits on a 16 bits plateform.*
*If we want the $m_i$ to be prime, for example in an homomorphic context, we can obtain 106 primes with $c_i \leq 616 < \frac{2^{16}}{106} - \frac{1}{2^{16}}$ and $2^{3391} \leq \prod m_i < 2^{3392}$. If we impose small values for the $c_i$, e.g. $c_i < 2^8$, we can obtain 136 co-primes moduli with $c_i \leq 253 < \frac{2^{16}}{136} - \frac{1}{2^{16}}$ and $2^{4351} \leq \prod m_i < 2^{4352}$. We can also easily obtain 45 primes with $c_i \leq 255 < \frac{2^{16}}{45} - \frac{1}{2^{16}}$ and $2^{1439} \leq \prod m_i < 2^{1440}$ .*

### 5.2.2 An exact conversion

Now, we consider $\rho \geq 2$ such that $X < \frac{1}{\rho}M$ as for the second conversion in Kawamura et al [33].

If $\sum_{i=1}^{n}\left(c_i+\frac{1}{2^{e_2}}\right) < 2^{e'_2}\left(1-\frac{1}{\rho}\right)$, then

$$\sum_{i=1}^{n}\left(\frac{\xi_i}{2^{e_2+e'_2}}\right) < \sum_{i=1}^{n}\frac{\xi_i}{m_i} < \sum_{i=1}^{n}\left(\frac{\xi_i}{2^{e_2+e'_2}}\right) + \left(1-\frac{1}{\rho}\right).$$

As $X < \frac{1}{\rho}M$, $\left\lfloor \sum_{i=1}^{n}\frac{\xi_i}{m_i}\right\rfloor$ equals $\left\lfloor \sum_{i=1}^{n}\frac{\xi_i}{m_i} + \left(1-\frac{1}{\rho}\right)\right\rfloor$, so we have

$$\left\lfloor \sum_{i=1}^{n}\frac{\xi_i}{m_i}\right\rfloor = \left\lfloor \sum_{i=1}^{n}\frac{\xi_i}{2^{e_2+e'_2}} + \left(1-\frac{1}{\rho}\right)\right\rfloor.$$

That means that in Algorithm 7, the second "conversion of $r$ from $\mathcal{B}'$ to $\mathcal{B}$" can be done exactly as in [33].

**Example 2** *As in Example 1, we consider $2^{e_2} = 2^{e'_2} = 2^{16}$, adding $\rho = 2$. In this case, we can construct a set of 132 co-primes with $2^{4223} \le \prod_{i=1}^{132} m_i < 2^{4224}$ and a maximal value of $c_i$ equals to $243 < \frac{2^{15}}{132} - \frac{1}{2^{16}}$. But also 75 primes with $2^{2399} \le \prod m_i < 2^{2400}$ and $c_i \le 436 < \frac{2^{16}}{106} - \frac{1}{2^{16}}$ or 45 if the $c_i$ are chosen less than $2^8$ and then $2^{1439} \le \prod m_i < 2^{1440}$ and $c_i \le 255 < \frac{2^{16}}{45} - \frac{1}{2^{16}}$.*

### 5.2.3 A conversion with truncated values

The two previous conversions require the calculation of $\sum_{i=1}^{n}\frac{\xi_i}{2^{e_2+e'_2}} = \frac{1}{2^{e_2+e'_2}}\sum_{i=1}^{n}\xi_i$. Suppose our architecture is on $(e_2+e'_2)$-bits. Since we are only interested in the most significant digits of the $\sum_{i=1}^{n}\xi_i$, it might be interesting to consider a truncated summation that stays on less than $(e_2+e'_2)$-bits. So, we can, as in Kawamura et al [33], compute instead the sum $\sum_{i=1}^{n}\left\lfloor\frac{\xi_i}{2^t}\right\rfloor$, with $2^t \ge n$.
In this case we have

$$\begin{aligned}
\sum_{i=1}^{n}\frac{\xi_i}{m_i} &< \sum_{i=1}^{n}\left(\frac{\xi_i}{2^{e_2+e'_2}} + \frac{c_i}{2^{e'_2}} + \frac{1}{2^{e_2+e'_2}}\right)\\
&< \frac{1}{2^{e_2+e'_2}}\sum_{i=1}^{n}\left(\left\lfloor\frac{\xi_i}{2^t}\right\rfloor 2^t + |\xi_i|_{2^t}\right) + \sum_{i=1}^{n}\left(\frac{c_i}{2^{e'_2}} + \frac{1}{2^{e_2+e'_2}}\right)\\
&< \frac{2^t}{2^{e_2+e'_2}}\sum_{i=1}^{n}\left\lfloor\frac{\xi_i}{2^t}\right\rfloor + \frac{n2^t}{2^{e_2+e'_2}} + \frac{1}{2^{e'_2}}\sum_{i=1}^{n}\left(c_i + \frac{1}{2^{e_2}}\right).
\end{aligned}$$

Thus, if $\sum_{i=1}^{n}\left(c_i+\frac{1}{2^{e_2}}\right) < 2^{e'_2} - \frac{n2^t}{2^{e_2}}$, we have

$$\frac{2^t}{2^{e_2+e'_2}}\sum_{i=1}^{n}\left\lfloor\frac{\xi_i}{2^t}\right\rfloor \le \sum_{i=1}^{n}\frac{\xi_i}{m_i} < \frac{2^t}{2^{e_2+e'_2}}\sum_{i=1}^{n}\left\lfloor\frac{\xi_i}{2^t}\right\rfloor + 1,$$

$$\text{so that } \left\lfloor\sum_{i=1}^{n}\frac{\xi_i}{m_i}\right\rfloor = \begin{cases} \left\lfloor\frac{2^t}{2^{e_2+e'_2}}\sum_{i=1}^{n}\left\lfloor\frac{\xi_i}{2^t}\right\rfloor\right\rfloor & \text{or}\\ \left\lfloor\frac{2^t}{2^{e_2+e'_2}}\sum_{i=1}^{n}\left\lfloor\frac{\xi_i}{2^t}\right\rfloor\right\rfloor + 1. \end{cases}$$

If we consider $\rho \geq 2$ such that $X < \frac{1}{\rho}M$ and if we assume $\sum_{i=1}^{n}\left(c_i + \frac{1}{2^{e_2}}\right) < 2^{e'_2}\left(1 - \frac{1}{\rho}\right) - \frac{n2^t}{2^{e_2}}$, then

$$\left\lfloor \sum_{i=1}^{n} \frac{\xi_i}{m_i} \right\rfloor = \left\lfloor \frac{2^t}{2^{e_2+e'_2}} \sum_{i=1}^{n} \left\lfloor \frac{\xi_i}{2^t} \right\rfloor + 1 - \frac{1}{\rho} \right\rfloor.$$

Note that the larger $t$ is, the smaller the architecture needed to calculate this summation is.

### 5.2.4 Summary

In this work, we introduced the RNS bases of coprimes $m_i = 2^{e_2}(2^{e'_2} - c_i) \mp 1$ with $0 \leq c_i < 2^{e'_2}$ for $i = 1 \dots n$. If $0 \leq X < M$ is represented in this base by $(x_1, \dots, x_n)$, $\xi_i = \left|x_i M_i^{-1}\right|_{m_i}$ and $\alpha = \left\lfloor \sum_{i=1}^{n} \frac{\xi_i}{m_i} \right\rfloor$, the CRT construction is given by

$$X = \sum_{i=1}^{n} \frac{\xi_i}{m_i} M - \alpha M.$$

Let us summarize how to find $\alpha$ in the general case:

- Assuming $\sum_{i=1}^{n}\left(c_i + \frac{1}{2^{e_2}}\right) < 2^{e'_2}$, we have
$$\alpha = \left\lfloor \frac{1}{2^{e_2+e'_2}} \sum_{i=1}^{n} \xi_i \right\rfloor \text{ or } \left\lfloor \frac{1}{2^{e_2+e'_2}} \sum_{i=1}^{n} \xi_i \right\rfloor + 1.$$

- If $2^t \geq n$ and $\sum_{i=1}^{n}\left(c_i + \frac{1}{2^{e_2}}\right) < 2^{e'_2} - \frac{n2^t}{2^{e_2}}$,
$$\alpha = \left\lfloor \frac{2^t}{2^{e_2+e'_2}} \sum_{i=1}^{n} \left\lfloor \frac{\xi_i}{2^t} \right\rfloor \right\rfloor \text{ or } \left\lfloor \frac{2^t}{2^{e_2+e'_2}} \sum_{i=1}^{n} \left\lfloor \frac{\xi_i}{2^t} \right\rfloor \right\rfloor + 1.$$

If $\rho \geq 2$ such that $X < \frac{1}{\rho}M$, we get an exact result:

- Assuming $\sum_{i=1}^{n}\left(c_i + \frac{1}{2^{e_2}}\right) < 2^{e'_2}\left(1 - \frac{1}{\rho}\right)$, we have
$$\alpha = \left\lfloor \frac{1}{2^{2e_2}} \sum_{i=1}^{n} \xi_i + \left(1 - \frac{1}{\rho}\right) \right\rfloor.$$

- If $2^t \geq n$ and $\sum_{i=1}^{n}\left(c_i + \frac{1}{2^{e_2}}\right) < 2^{e'_2}\left(1 - \frac{1}{\rho}\right) - \frac{n2^t}{2^{e_2}}$,
$$\alpha = \left\lfloor \frac{2^t}{2^{e_2+e'_2}} \sum_{i=1}^{n} \left\lfloor \frac{\xi_i}{2^t} \right\rfloor + \left(1 - \frac{1}{\rho}\right) \right\rfloor.$$

## 5.3 Some comparisons

The main difference between our approach with Montgomery's RNS bases and pseudo-Mersenne bases, and the approach of Kawamura et al [33] lies in the internal modular reduction modulo $m_i$, the elements of the RNS bases. In the first one, we use Algorithm 6 which uses two multiplications of a value on $e_2$-bits by $c_i$, then in the second one we use Algorithm 2 with a multiplication by $c_i$ of a value on $(e_2 + e_2')$-bits and one by $c_i$ of a value on $e_2$-bits. Depending on the architecture, size or even shape, the $c_i$ values are therefore important.
Another difference is the upper bound for $c = \max c_i$. In the general case we need that $c + \frac{1}{2^{e_2}} < \frac{2^{e_2'}}{n} - \frac{2^t}{2^{e_2}}$ while Kawamura et al requires $c < \frac{2^{e_2+e_2'}}{n} - 2^t$. Our upper bound is then smaller than the one given by Kawamura et al [33]. More precisely it is essentially $2^{e_2}$ times smaller and the situation is similar for the exact conversion with $\rho \geq 2$. But, on the other hand, for the same selection criterion of the $c_i$, we have at least 4 times more candidates for the same bound on the $c_i$ (2 times because of the $\pm 1$ concerned and 2 times because we can consider even values of $c_i$ contrary to [33]). Consequently, we are able to build bases with smaller $c_i$. Similarly, if the maximum size of $c_i$ is supposed to be small, we can build bases larger than [33].

| $e = e_2 + e_2'$ ($m_i < 2^e$) | $ec$ ($c_i < 2^{ec}$) | $\rho$ | $t$ | $n$ (Kawamura et al [33]) ($m_i = 2^{e_2+e_2'} - c_i$) | $n$ (this paper) ($m_i = 2^{e_2}(2^{e_2'} - c_i) \mp 1$) |
|---|---|---|---|---|---|
| 16 | 4 | 2 | 10 | 2 | 7 |
| 16 | 6 | 2 | 10 | 7 | 7 |
| 16 | 7 | 2 | 10 | 11 | 7 |
| 16 | 8 | 2 | 10 | 21 | 7 |
| 16 | 6 | 2 | 7 | 7 | 8 |
| 16 | 8 | 2 | 7 | 21 | 8 |
| 32 | 4 | 2 | 24 | 2 | 7 |
| 32 | 6 | 2 | 24 | 7 | 21 |
| 32 | 8 | 2 | 24 | 23 | 65 |
| 32 | 9 | 2 | 24 | 36 | 65 |
| 32 | 10 | 2 | 24 | 66 | 65 |
| 32 | 8 | 2 | 20 | 23 | 68 |
| 32 | 10 | 2 | 20 | 66 | 89 |
| 32 | 11 | 2 | 20 | 121 | 89 |
| 64 | 4 | 2 | 56 | 2 | 8 |
| 64 | 6 | 2 | 56 | 7 | 20 |
| 64 | 8 | 2 | 56 | 21 | 62 |
| 64 | 10 | 2 | 56 | 67 | 127 |
| 64 | 12 | 2 | 56 | 127 | 127 |
| 64 | 13 | 2 | 56 | 127 | 127 |
| 64 | 14 | 2 | 56 | 127 | 127 |
| 64 | 10 | 2 | 48 | 67 | 205 |
| 64 | 12 | 2 | 48 | 215 | 688 |
| 64 | 13 | 2 | 48 | 388 | 1295 |
| 64 | 14 | 2 | 48 | 710 | 2365 |

Table 4: Largest $n$ reached for [33] and our approach with a greedy search algorithm based on increasing the $c_i$

Using a greedy algorithm based on increasing the $c_i$, we give in Table 4 the largest size $n$ of the RNS bases that can be reached by Kawamura et al [33] approach and by ours. It is given in function of the parameters $e = e_2 + e_2'$ (the bitsize of the moduli), $ec$ (the maximal bitsize expected for the $c_i$), $\rho$ and $t$ (the truncation size), knowing that at least $2n$ base elements are required.
We note that, as expected, our approach reaches bigger RNS bases than Kawamura et al for small $c_i$ (which is important in term of efficiency because the internal modular reduction modulo $m_i$ uses multiplications by $c_i$). We remark also that, by changing the size of the truncation,

our approach becomes a little bit better, see for exemple $e = 16$ or $e = 32$. These phenomena correspond to the bounds given on $c$ which are smaller in our case so that $t$ may have greater influence.
Another interesting point comes from the use of Algorithm 6. This algorithm has the advantage of being able to use basic RNS elements of twice the size of the architecture because the multiplications are on $e_2$-bit values instead of $(e_2 + e_2')$-bits. For example, for an architecture of 16-bits, we can consider $e = e_2 + e_2' = 32$ and $e_2 = 16$. As a consequence RNS-based arithmetic can be considered for larger fields or rings on such architectures. For example, a RNS-based implementation of RSA-1024 can be done on a 16-bits architecture (we need 64 moduli with $e_2 = 16$ for this) while it cannot be done directly with pseudo-Mersenne 16-bits moduli (128 are required with $e = 16$) and are more costly with 32-bits moduli due to Algorithm 2.
In a recent paper, Kawamura et al [34] give an interesting survey of different approaches [33, 25, 17, 23] about RNS ECC implementations. They also suggest an original approach using bases where each element is a square modulo the others and $p$ is a square modulo each element of the bases (the elliptic curve is defined over $\mathbb{F}_p$). The bases proposed by our approach are compatible with all the previous approaches and also with the one of [34]. Moreover, one can see in Table 5 that our approach offers smaller $c_i$ than [34].

| Method | (p, e, ec, $\rho$, t, 2n) | [ $c_i$] | size |
|---|---|---|---|
| This paper | (p192,50,12,2,42,8) | [2, 68, 98, 134, 158, 206, 1902, 2903], | 199 |
| [34] | (p192,50,14,2,42,8) | [27, 117, 351, 951, 1163, 2567, 2855, 8543] | 199 |
| This paper | (p224,58,11,2,53,8) | [32, 67, 122, 123, 413, 1653, 1751, 1943] | 231 |
| [34] | (p224,58,13,2,53,8) | [27, 57, 63, 147, 447, 731, 3807, 7403] | 231 |
| This paper | (p256,65,12,2,58,8) | [49, 73, 391, 512, 578, 1630, 2212, 3812] | 259 |
| [34] | (p256,65,14,2,58,8) | [49, 535, 751, 979, 2191, 3219, 8031, 11335] | 260 |
| This paper | (p384,98,14,2,90,8) | [7, 21, 44, 792, 2301, 3251, 5286, 8601] | 391 |
| [34] | (p384,98,16,2,90,8) | [51, 117, 831, 855, 1571, 1827, 4343, 52155] | 392 |
| This paper | (p521,132,14,2,124,8) | [4, 43, 252, 315, 906, 3010, 7696, 9022] | 528 |
| [34] | (p521,132,16,2,124,8) | [347, 363, 527, 725, 6647, 11535, 38679, 38835] | 528 |
| This paper | (pCurve25519,64,12,2,57,8) | [1, 4, 92, 241, 483, 1230, 2410, 3645] | 255 |
| [34] | (pCurve25519,64,14,2,57,8) | [59, 83, 323, 899, 3263, 6983, 8547, 13515] | 256 |

Table 5: Comparison of our approach with the one of Kawamura et al [34]. $p$ defines the finite field, $e = e_2 + e_2'$, $ec > \log_2(c_i)$, $t$ is the truncation size, and $2n$ the number of $m_i$ for the two bases needed. We return the set of $c_i$ defining the RNS bases.

In [9], the authors propose a double Montgomery reduction in a Cox-Rover like architecture which is more general than our current approach (which also involves a double level of Montgomery reduction). In their approach, the Kawamura et al. condition $c < \frac{2^{e_2+e_2'}}{n}(1 - \frac{1}{\rho}) - 2^t$ becomes $c < \frac{2^{e_2+e_2'}}{n}(1 - \frac{1}{\rho}) - 2^{e_2+e_2'-t} + 1$. As a consequence, they can generate more bases than [33] as they allow bigger $c_i$. This could be very interesting when $e_2 + e_2'$ is small, but the cost of the multiplication by $c_i$ can become costly depending of the architecture and this approach has the same drawback than [33] compared to ours. When $e_2 + e_2' \geq 32$ our approach is clearly very attractive.

## 6 Conclusions

Montgomery-friendly prime numbers, which are particularly suitable for Montgomery modular reductions, form a large family. They offer new prime numbers that can be used in many cryptographic contexts such as elliptic curve or isogeny based cryptography or RNS arithmetic for large numbers. For isogenies, we have proposed new primes that significantly improve the complexity of the modular reduction step for each of the NIST security categories. For ECC, we provide new primes for base fields for the 3 standard security levels as alternatives to the pseudo-Mersenne primes like $2^{255} - 19$, or other Montgomery-friendly primes like $2^{240}(2^{16} - 88) - 1$. Again, the modular reduction is expected to better depend on the relative costs of word additions versus word multiplications. Note that we get curves with smaller coefficients than the existing ones which can be an advantage for small devices. The use of Montgomery Friendly moduli for RNS implementations offers several advantages. It is indeed possible to obtain larger sets of pairwise co-prime numbers with very small $c_i$ parameters. The size of this parameter is a very important criterion for the cost of the modular reduction at the moduli level. Moreover, Algorithm 6 allows the use of moduli with a size that is twice the size of the operators of the targeted architecture.

## References

[1] Adj, G., Cervantes-Vázquez, D., Chi-Domínguez, J.J., Menezes, A., Rodríguez-Henríquez, F.: On the cost of computing isogenies between supersingular elliptic curves. In: C. Cid, M.J. Jacobson Jr. (eds.) Selected Areas in Cryptography – SAC 2018, *Lecture notes in computer science*, vol. 11349, pp. 322–343 (2019)

[2] Antao, S., Bajard, J.C., Sousa, L.: Elliptic Curve point multiplication on GPUs. In: 21st IEEE International Conference on Application-specific Systems Architectures and Processors, ASAP (2010). DOI 10.1109/ASAP.2010.5541000

[3] Azarderakhsh, R., Campagna, M., Costello, C., Feo, L.D., Hess, B., Jalali, A., Jao, D., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Pereira, G., Renes, J., Soukharev, V., Urbanik, D.: Supersingular isogeny key encapsulation. Tech. rep., Submission to the NIST's post-quantum cryptography standardization process (2019). `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions/SIKE.zip`

[4] Azarderakhsh, R., Campagna, M., Costello, C., Feo, L.D., Hess, B., Jalali, A., Jao, D., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Renes, J., Soukharev, V., Urbanik, D.: Supersingular isogeny key encapsulation. Tech. rep., Submission to the NIST's post-quantum cryptography standardization process (2017). `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/SIKE.zip`

[5] Bajard, J., Didier, L., Kornerup, P.: An RNS montgomery modular multiplication algorithm. In: 13th Symposium on Computer Arithmetic (ARITH-13 '97), 6-9 July 1997, Asilomar, CA, USA, pp. 234–239 (1997)

[6] Bajard, J., Didier, L., Kornerup, P.: Modular multiplication and base extensions in residue number systems. In: 15th IEEE Symposium on Computer Arithmetic (Arith-15 2001), 11-17 June 2001, Vail, CO, USA, pp. 59–65 (2001)

[7] Bajard, J.C., Eynard, J., Hasan, M.A., Zucca, V.: A Full RNS Variant of FV Like Somewhat Homomorphic Encryption Schemes. In: 23rd International Conference on Selected Areas in Cryptography - SAC, *Lecture Notes in Computer Science*, vol. 10532, pp. 423–442. Springer Berlin Heidelberg (2016). DOI {10.1007/978-3-319-69453-5_23}

[8] Bajard, J.C., Imbert, L.: A Full RNS Implementation of RSA. IEEE Trans. Computers **53**(6) (2004). DOI https://doi.org/10.1109/TC.2004.2

[9] Bajard, J.C., Merkiche, N.: Double level montgomery cox-rower architecture, new bounds. In: Smart Card Research and Advanced Applications. CARDIS 2014, vol. Lecture Notes in Computer Science, vol 8968. Springer Berlin Heidelberg (2014)

[10] Bernstein, D., Lange, T.: Safecurves: choosing safe curves for elliptic-curve cryptography. `http://safecurves.cr.yp.to`

[11] Bernstein, D.J.: Curve25519: New diffie-hellman speed records. In: Public Key Cryptography - PKC 2006, Lecture notes in computer science, pp. 207–228 (2006)

[12] Bernstein, D.J., Chuengsatiansup, C., Lange, T., Schwabe, P.: Kummer strikes back: New dh speed records. In: Advances in Cryptology – ASIACRYPT 2014, *Lecture notes in computer science*, vol. 8873, pp. 317–337 (2014)

[13] Bos, J., Lenstra, A. (eds.): Topics in Computational Number Theory Inspired by Peter L. Montgomery. Cambridge University Press (2017). DOI 10.1017/9781316271575

[14] Bos, J.W., Costello, C., Hisil, H., Lauter, K.: Fast cryptography in genus 2. In: T. Johansson, P.Q. Nguyen (eds.) Advances in Cryptology – EUROCRYPT 2013, pp. 194–210. Springer Berlin Heidelberg (2013)

[15] Bos, J.W., Costello, C., Longa, P., Naehrig, M.: Selecting elliptic curves for cryptography: an efficiency and security analysis. Journal of Cryptographic Engineering **6**(4), 259–286 (2016)

[16] Bos, J.W., Montgomery, P.L.: Montgomery Arithmetic from a Software Perspective, pp. 10–39. Cambridge University Press (2017). DOI 10.1017/9781316271575.003

[17] Cheung, R.C.C., Duquesne, S., Fan, J., Guillermin, N., Verbauwhede, I., Yao, G.X.: FPGA Implementation of Pairings Using Residue Number System and Lazy Reduction. In: Cryptographic Hardware and Embedded Systems - CHES 2011 , *Lecture Notes in Computer Science*, vol. 6917, pp. 421–441. Springer Berlin Heidelberg (2011). DOI 10.1007/978-3-642-23951-9_28

[18] Costello, C., Jao, D., Longa, P., Naehrig, M., Renes, J., Urbanik, D.: Efficient compression of sidh public keys. In: J.S. Coron, J.B. Nielsen (eds.) Advances in Cryptology – EUROCRYPT 2017, *Lecture notes in computer science*, vol. 10210, pp. 679–706 (2017)

[19] Costello, C., Longa, P., Naehrig, M.: Efficient algorithms for supersingular isogeny diffie-hellman. In: M. Robshaw, J. Katz (eds.) Advances in Cryptology – CRYPTO 2016, *Lecture notes in computer science*, vol. 9814, pp. 572–601 (2016)

[20] Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006). `https://eprint.iacr.org/2006/291`

[21] Crandall, R.: Method and apparatus for public key exchange in a cryptographic system (1992). U.S. Patent #5159632

[22] Gallagher, P., Foreword, D.D., Director, C.F.: Fips pub 186-3 federal information processing standards publication digital signature standard (dss) (2009). U.S.Department of Commerce/National Institute of Standards and Technology

[23] Gandino, F., Lamberti, F., Paravati, G., Bajard, J.C., Montuschi, P.: An algorithmic and architectural study of montgomery exponentiation in rns. IEEE Trans. Compu **61**(8) (2012)

[24] Garner, H.L.: The Residue Number System. IRE Transactions on Electronic Computers **EC-8**(2), 140–147 (1959). DOI {10.1109/TEC.1959.5219515}

[25] Guillermin, N.: A high speed coprocessor for elliptic curve scalar multiplications over fp. In: CHES2010, *LNCS*, vol. 6225. Springer (2010)

[26] Halevi, S., Polyakov, Y., Shoup, V.: An Improved RNS Variant of the BFV Homomorphic Encryption Scheme. In: Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference, *Lecture Notes in Computer Science*, vol. 11405, pp. 83–105. Springer Berlin Heidelberg (2019). DOI {10.1007/978-3-030-12612-4_5}

[27] Hamburg, M.: Fast and compact elliptic-curve cryptography. IACR Cryptology ePrint Archive p. 309 (2012). URL `http://eprint.iacr.org/2012/309`

[28] Hamburg, M.: Ed448-goldilocks, a new elliptic curve. Cryptology ePrint Archive, Report 2015/625 (2015). `https://eprint.iacr.org/2015/625`

[29] van der Hoven, J.: Fast chinese remaindering in practice. In: Mathematical Aspects of Computer and Information Sciences. MACIS 2017, vol. Lecture Notes in Computer Science, vol 10693. Springer Berlin Heidelberg (2017)

[30] Jao, D., De Feo, L., Plut, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Journal of Mathematical Cryptology **8**(3), 209–247 (2014)

[31] Jao, D., Feo, L.D.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: PQCrypto 2011: Post-Quantum Cryptography, vol. Lecture Notes in Computer, vol.7071, pp. 19–34. Springer Berlin Heidelberg (2011)

[32] Joux, A.: A one round protocol for tripartite diffie–hellman. Journal of Cryptology **17**(4), 263–276 (2004). DOI https://doi.org/10.1007/s00145-004-0312-y

[33] Kawamura, S., Koike, M., Sano, F., Shimbo, A.: Cox-Rower Architecture for Fast Parallel Montgomery Multiplication. In: Proc. EUROCRYPT 2000, *LNCS*, vol. 1807, pp. 523–538. Springer (2000)

[34] Kawamura, S., Komano, Y., Shimizu, H., Yonemura, T.: Rns montgomery reduction algorithms using quadratic residuosity. Journal of Cryptographic Engineering **9**(4) (2019)

[35] Koblitz, N.: Elliptic curve cryptosystems. Math. Comp. **48**, 203–209 (1987). DOI https://doi.org/10.1090/S0025-5718-1987-0866109-5

[36] Koziel, B., Azarderakhsh, R., Mozaffari Kermani, M., Jao, D.: Post-quantum cryptography on fpga based on isogenies on elliptic curves. IEEE Transactions on Circuits and Systems I: Regular Papers **64**(1), 86–99 (2017)

[37] Miller, V.S.: Use of elliptic curves in cryptography. In: Advances in Cryptology — CRYPTO '85 Proceedings, vol. Lecture Notes in Computer Science, vol. 218, pp. 417–426. Springer Berlin Heidelberg (1985)

[38] Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation **44**, 519–521 (1985)

[39] Murty, R.: Prime numbers and irreducible polynomials. The American Mathematical Monthly **109**(5) (2002)

[40] NIST: Post-quantum cryptography standardization (2017). `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization`

[41] Posch, K., Posch, R.: Modulo Reduction in Residue Number Systems. IEEE Trans. Parallel Distrib. Syst. **6**(5), 449–454 (1995)

[42] Renes, J., Schwabe, P., Smith, B., Batina, L.: $\mu$Kummer: efficient hyperelliptic signatures and key exchange on microcontrollers. In: Cryptographic Hardware and Embedded Systems – CHES 2016, *Lecture notes in computer science*, vol. 9813, p. 20 (2016)

[43] Rivest, R. L. and Shamir, A. and Adleman, L.: A Method for Obtaining Digital Signatures and Public-key Cryptosystems. Commun. ACM **21**(2), 120–126 (1978). DOI {10.1145/359340.359342}

[44] Robinson, R.M.: Mersenne and fermat numbers. Proc. Amer. Math. Soc. **5**, 842–846 (1954)

[45] Shenoy, A.P., Kumaresan, R.: Fast base extension using a redundant modulus in RNS. IEEE Trans. Computers **38**(2), 292–297 (1989)

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