Représentation RNS des nombres et calcul de couplages

Sylvain Duquesne

Université Rennes 1

Séminaire CCIS
Grenoble, 7 Février 2013

Institut de recherche mathématique de Rennes

IRMAR - UMR 6625 du CNRS

Outline

- Standard prime field arithmetic
- RNS (Residue Number System) arithmetic
- Application to standard elliptic curve arithmetic
  (Joint work with JC. Bajard and M. Ercegovac)
- RNS arithmetic in extension fields
- Application to pairing computations
- Practical implementation (FPGA)
  (Joint work with N. Guillermin and Cheung-Fan-Verbauwhede-Yao)

Basic arithmetic over large prime fields

Context

- p a large prime of size β^n with β the size of a word ($\beta = 2^{32}$)
- A, B in $[0, p-1]$
- All numbers are represented in base β

We want to compute $A \times B$ modulo p

Usual method

Multiplication : Schoolbook method (quadratic) or better if n large
Reduction : Euclidean division

Advanced methods for reduction

- Use Mersenne or pseudo-Mersenne primes, $p = \beta^n - c$ with c small
- Use Montgomery reduction

Montgomery reduction algorithm

Assume that integers are given in Montgomery representation, i.e. an integer A is represented by $A\beta^n \bmod p$.

Algorithm

Data	$R = AB < \beta^{2n}$ and $-p^{-1} \bmod \beta^n$ (precomputed)
Compute	$q = -R \times p^{-1} \bmod \beta^n$
	$r = (R + q \times p)/\beta^n$
Result	$r < 2p$ and $r = R\beta^{-n} \bmod p$

Either r or $r - p$ is the reduction of R modulo p in Montgomery representation

Finally, we replace a computation modulo p by a computation modulo β^n

Practical use of Montgomery reduction

Example : compute x^5 mod p

1. Compute $\overline{x} = x\beta^n$ modulo p
2. Compute $\overline{y} = \overline{x}^2$ and "reduce" it modulo β^n
3. Compute $\overline{z} = \overline{y}^2$ and "reduce" it modulo β^n
4. Compute $\overline{t} = \overline{z} \times \overline{x}$ and "reduce" it modulo β^n
5. Recover $x^5 = \overline{t} \times \beta^{-n}$ mod p

Use in cryptography : RSA-1024 decryption requires 1200 to 1500 modular multiplications $\rightarrow$ time for changing the representation is negligible.

Complexity of Montgomery modular multiplication

Operations : $q = -R \times p^{-1}$ mod β^n and $r = (R + q \times p)/\beta^n$.
The cost of each multiplication is $n^2/2$ because we are only interested by a part of the result $\rightarrow n^2 + n$.

Overall cost of modular multiplication : $2n^2 + n$

The Residue Number System representation

First approach

Let $m_1, m_2, \ldots, m_{n-1}, m_n$ relatively prime numbers and $M = \prod_{i=1}^{n} m_i$

We can represent $X \in [0, M]$ with $(x_1, x_2, \ldots, x_n)$ such that :

$$\left| \begin{array}{l} x_1 = X \bmod m_1 \\ \vdots \\ x_n = X \bmod m_n \end{array} \right.$$

$(m_1, m_2, \ldots, m_n)$ is named RNS base, we denote it $\mathcal{B}_n$.

Operations

$$X_{RNS} + Y_{RNS} = ((x_1 + y_1) \bmod m_1, \ldots, (x_n + y_n) \bmod m_n)_{RNS}$$
$$X_{RNS} \times Y_{RNS} = ((x_1 \times y_1) \bmod m_1, \ldots, (x_n \times y_n) \bmod m_n)_{RNS}$$

Advantages

Advantages

- No carry propagation
- Multiplication becomes linear in n
- If the m_i are chosen like pseudo-Mersenne primes i.e. $m_i = \beta - c_i$ with c_i small, reducing modulo m_i is very fast
- Easy parallelization with n processors

Advantages and disadvantages

Advantages

- No carry propagation
- Multiplication becomes linear in n
- If the m_i are chosen like pseudo-Mersenne primes i.e. $m_i = \beta - c_i$ with c_i small, reducing modulo m_i is very fast
- Easy parallelization with n processors

Drawback

- p prime, so $p \neq \prod_{i=1}^{n} m_i$

Question : is it possible to perform prime field arithmetic using RNS ?

From Montgomery reduction to RNS reduction

Representation

A is represented by $A\beta^n \bmod p$

Algorithm

Data	$R = AB < \beta^{2n}$
	$-p^{-1} \bmod \beta^n$
Compute	$q = -R \times p^{-1} \bmod \beta^n$
	$r = (R + q \times p)/\beta^n$
Result	$r < 2p$
	$r = R\beta^{-n} \bmod p$

Computation modulo p replaced by

computation modulo β^n

From Montgomery reduction to RNS reduction

Representation

A is represented by $A\beta^n \bmod p$	A is represented by $AM \bmod p$

Algorithm

Data	$R = AB < \beta^{2n}$ $-p^{-1} \bmod \beta^n$	Data	$R = AB$ in $\mathcal{B}_n$ $-p^{-1}$ in $\mathcal{B}_n$
Compute	$q{=}-R \times p^{-1} \bmod \beta^n$ $r{=}(R + q \times p)/\beta^n$	Compute	$q{=}-R \times p^{-1}$ in $\mathcal{B}_n$ $r{=}(R + q \times p)M^{-1}$ in $\mathcal{B}_n$
Result	$r < 2p$ $r = R\beta^{-n} \bmod p$	Result	$r < 2p$ $r = RM^{-1} \bmod p$

Computation modulo p replaced by

computation modulo β^n	computation modulo M

From Montgomery reduction to RNS reduction

Representation

A is represented by $A\beta^n \bmod p$	A is represented by $AM \bmod p$

Algorithm

Data	$R = AB < \beta^{2n}$ $-p^{-1} \bmod \beta^n$	Data	$R = AB$ in $\mathcal{B}_n$ $-p^{-1}$ in $\mathcal{B}_n$
Compute	$q{=}-R \times p^{-1} \bmod \beta^n$ $r{=}(R + q \times p)/\beta^n$	Compute	$q{=}-R \times p^{-1}$ in $\mathcal{B}_n$ $r{=}(R + q \times p)M^{-1}$ in $\mathcal{B}_n$
Result	$r < 2p$ $r = R\beta^{-n} \bmod p$	Result	$r < 2p$ $r = RM^{-1} \bmod p$

Computation modulo p replaced by

computation modulo β^n	computation modulo M

RNS reduction

Introduce a new RNS basis to handle M^{-1} (Bajard, Didier, Kornerup, 2001)

Algorithm

Data	Two coprime RNS basis $\mathcal{B}_n$ and $\mathcal{B}'_n$
	$R = AB$ in $\mathcal{B}_n$ and $\mathcal{B}'_n$
	$-p^{-1}$ in $\mathcal{B}_n$ and M^{-1} in $\mathcal{B}'_n$ (precomputations)
Compute	$q{=}-R \times p^{-1}$ in $\mathcal{B}_n$
	q in $\mathcal{B}_n \rightarrow \hat{q}$ in $\mathcal{B}'_n$ (change of basis)
	$\hat{r}{=}(R + \hat{q} \times p)M^{-1}$ in $\mathcal{B}'_n$
	$\hat{r}$ in $\mathcal{B}'_n \rightarrow r$ in $\mathcal{B}_n$ (change of basis)
Result	$r < 2p$
	$r = RM^{-1} \bmod p$

Complexity

The RNS reduction requires $2n^2 + 3n$ small (word size) operations.
Overall cost of modular RNS multiplication : $2n^2 + 5n$

Is RNS really interesting ? ($2n^2 + n$ for Montgomery arithmetic ...)

Other advantages of the RNS

- Easy to parallelize
- High flexibility
- Leak-resistant arithmetic
- $A \times B + C \times D$ require only $2n^2 + 7n$ operations

Gap between complexity of multiplication and complexity of reduction.

$\rightarrow$ Try to optimize ECC formulas to take advantage of this.

Elliptic curve cryptography

Group law

An elliptic curves is defined over $\mathbb{F}_p$ by an equation $y^2 = x^3 + ax + b$.
It has an explicit group law given by the chord and tangent rule

$$\Rightarrow \text{Cryptography based on discrete logarithm}$$

Security parameters

Best discrete log algorithms in $O(\sqrt{p})$.

80 bits security $\rightarrow$ 160 bits elliptic curve (RSA 1024)

128 bits security $\rightarrow$ 256 bits elliptic curve (RSA 3072)

Scalar multiplication (computation of nP)

- $T \leftarrow P$
- for each bit of n do
  $T \leftarrow 2T$
  if the bit is 1 do $T \leftarrow T + P$

Application to standard elliptic curve arithmetic

Doubling formulas in Jacobian coordinates

We want to compute $2T$ if $T = (X_T, Y_T, Z_T)$ is a point on the elliptic curve defined by the equation $Y^2 = X^3 + aXZ^4 + bZ^6$.
Let $A = 3X_T^2 + aZ_T^4$ and $C = 4X_T Y_T^2$, then

$$X_{2T} = A^2 - 2C, \quad Y_{2T} = A(C - X_{2T}) - 8Y_T^4, \quad Z_{2T} = 2Y_T Z_T$$

This requires 4M and 6S but 2 reductions can be saved (in A and Y_{2T})

Assuming (for simplicity) that S=M, usual Montgomery arithmetic requires $20n^2 + 10n$ operations.
But RNS arithmetic requires only

$$10\times 2n + 8\times(2n^2 + 3n) = 16n^2 + 44n$$

Better asymptotical complexity but not interesting for cryptographic sizes.

$\mathbb{F}_{p^k}$ arithmetic

Usually quadratic (schoolbook) or subquadratic (Karatsuba,...) in k but

linear in k in RNS representation

Example of $\mathbb{F}_{p^3}$ arithmetic

Assume $\mathbb{F}_{p^3} = \mathbb{F}_p[z]/(z^3 - \beta)$ with β small.
let $f = f_0 + f_1 z + f_2 z^2$ and $g = g_0 + g_1 z + g_2 z^2$ in $\mathbb{F}_{p^3}$, then

$$fg = (f_0 g_0 + f_1 g_2 \beta + f_2 g_1 \beta) + (f_0 g_1 + f_1 g_0 + f_2 g_2 \beta) z + (f_0 g_2 + f_2 g_0 + f_1 g_1) z^2$$

requires 9 multiplications in $\mathbb{F}_p$ but only 3 reductions.
This can be reduced to 6M and 3R using Karatsuba's method.
The gain is less important if computing f^2 since efficient methods (Chung-Hasan) require 5M and 3R.

Devegili, O hEigeartaigh, Scott and Dahab study in full details $\mathbb{F}_{p^k}$ arithmetic for $k \leq 6 \Rightarrow$ detailed comparison.

$\mathbb{F}_{p^6}$ arithmetic

$\mathbb{F}_{p^6}$ is seen as a cubic extension of a quadratic one so that

- $f.g$ requires 18M and 6R
- f^2 requires 12M and 6R

	RNS	Montgomery
Word-complexity for $f.g$	$12n^2+54n$	$36n^2+18n$
n=5	570	990
n=16	3936	9504
Word-complexity for f^2	$12n^2+38n$	$24n^2+12n$
n=5	490	660
n=16	3680	6336

Security level	r	p^k	p	n
80	160	1024	160	5
128	256	3072	512	16

Better asymptotical complexity **and** interesting for cryptographic sizes.

Let us be a little more objective

Lazy reduction

The accumulation of product before reduction can also be done in Montgomery arithmetic.
$f.g$ requires 18 multiplications ($18n^2$) and 6 reductions ($6n^2 + 6n$)
$\Rightarrow$ overall complexity : $24n^2 + 6n$ (630 for $n = 5$)

Cost of basic operation

Most expensive in RNS (word-multiplication and reduction) than in Montgomery (word-multiplication). Additional cost estimated to 10% in the literature $\Rightarrow$ 627 for $n = 5$

More advantageous for RNS if $n = 16$, but less for f^2.

Improvement of RNS complexity

Bajard et al. recently prove that RNS change of basis can be done in only

$$\frac{7}{5}n^2 + \frac{8}{5}n \Rightarrow 9n^2 + 50n \text{ for } f.g \in \mathbb{F}_{p^6} (475 \text{ for } n = 5)$$

Size of the basis

Lazy reduction $\Rightarrow$ larger input in the reduction step.
Example 1 : $AB + CD$ has size $2p^2$.
Example 2 : In $f.g \in \mathbb{F}_{p^6}$, each component has size $372p^2$.

Montgomery reduction

Use an additional word to handle this factor $\rightarrow$ costly especially in cryptography.

RNS reduction

Choose RNS basis such that $\prod m_i$ sufficiently large $\rightarrow$ same remark

In some cases (FPGA), cryptographic sizes are less disadvantaged.

Pairings in cryptography

Definition

Let $(G_1, +), (G_2, +)$ and $(G_3, *)$ be 3 groups. A pairing is an application $e : G_1 \times G_2 \to G_3$ which is

- bilinear, ie $e(P + P', Q) = e(P, Q)e(P', Q)$
- non degenerate, ie $\forall P \in G_1, \exists Q \in G_2$ s.t. $e(P, Q) \neq 1$
- easily computable

Use in cryptography

- Destructive : Decisionnal Diffie-Hellman is easy, transfer of discrete log
- Constructive : Tripartite key-exchange, short signature, ID-based cryptography

Realization of pairings

Context

- E elliptic curve defined over $\mathbb{F}_p$ (p prime).
- $P \in E(\mathbb{F}_p)$ of prime order r.
- $k = 2d$ the embedding degree (smallest integer such that $r | p^k - 1$).
- $Q = (x, y\alpha) \in E(\mathbb{F}_{p^k})$ with $x, y \in \mathbb{F}_{p^d}$ and $\mathbb{F}_{p^k} = \mathbb{F}_{p^d}[\alpha]$

Definition

Let f_P be the function on the curve such that $\mathrm{Div}(f_P) = rP - r\infty$.

$$e(P, Q) = f_P(Q)^{\frac{p^k - 1}{r}} \in \mathbb{F}_{p^k}$$

Examples

- Supersingular curves ($k \leq 2$ in large characteristic)
- MNT curves ($k = 6$), optimal for 80 bits security
- Barreto-Naherig curves ($k = 12$), optimal for 128 bits security

Fast Tate pairing computation

The Miller loop (computation of $f_P(Q)$)

- $T \leftarrow P, f \leftarrow 1$
- for each bit of r do
  $f \leftarrow f^2.\ell_{T,T}(Q)$ and $T \leftarrow 2T$
  if the bit is 1 do $f \leftarrow f.\ell_{T,P}(Q)$ and $T \leftarrow T + P$

where $\ell_{A,B}$ is the equation of the line passing through A and B.

The final exponentiation (computation of $f^{\frac{p^k-1}{r}}$)

Split in an easy part (use of Frobenius) and a difficult part.
Difficult part is roughtly f^s with $s \approx p$ and even $p^{\frac{1}{2}}$ (MNT) or $p^{\frac{3}{4}}$ (BN).

Computation of $\ell_{T,T}(Q)$ in Jacobian coordinates

Let $A = 3X_T^2 + aZ_T^4$ and $C = 4X_T Y_T^2$.
Computation of $2T$ requires 10M and 8R.

$$X_{2T} = A^2 - 2C, \quad Y_{2T} = A(C - X_{2T}) - 8Y_T^4, \quad Z_{2T} = 2Y_T Z_T$$

It is easy to prove that

$$\ell_{T,T}(Q) = 2Y_T Z_T . Z_T^2 . y_Q \alpha - A . Z_T^2 . x_Q + A . X_T - 2Y_T^2$$

and that its computation requires

$k + 3$ multiplications in $\mathbb{F}_p$

$k + 2$ reductions (accumulate AX_T and the constant term of $AZ_T^2 x_Q$ before reducing)

No more exciting than standard elliptic curve arithmetic for RNS.

Application to MNT curves with $k = 6$

Miller loop : step complexity if the bit of r is zero

- 10M and 8R for the computation of $2T$
- 9M and 8R for the computation of $\ell_{T,T}(Q)$
- 12M and 6R for the squaring of f
- 18M and 6R for the multiplication of f^2 and $\ell_{T,T}(Q)$

	RNS	Montgomery	Gain
Word-complexity	$43n^2 + 147n$	$77n^2 + 28n$	
n=5	1810	2065	13%
n=16	13360	20160	34%

Final exponentiation : step complexity if the bit is zero

	RNS	Montgomery	Gain
Word-complexity	$9n^2 + 34.5n$	$18n^2 + 6n$	
n=5	397	480	17%
n=16	2856	4704	39%

$\mathbb{F}_{p^{12}}$ arithmetic

$\mathbb{F}_{p^{12}}$ is seen as a quadratic extension of a cubic extension of a quadratic one so that

- $f.g$ requires 54M and 12R
- f^2 requires 36M and 12R

	RNS	Montgomery
Word-complexity for $f.g$	$18.5n^2 + 129n$	$66n^2 + 12n$
n=8	2216	4320
Word-complexity for f^2	$18.5n^2 + 93n$	$48n^2 + 12n$
n=8	1928	3168

Security level	r	p^k	p	n
128	256	3072	256	8

Better asymptotical complexity **and** interesting for cryptographic sizes.

Application to BN curves ($k = 12$)

Miller loop : step complexity if the bit of r is zero

- 10M and 8R for the computation of $2T$
- 9M and 8R for the computation of $\ell_{T,T}(Q)$
- 36M and 12R for the squaring of f
- 39M and 12R for the multiplication of f^2 and $\ell_{T,T}(Q)$

	RNS	Montgomery	Gain
Word-complexity	$61.5n^2 + 258n$	$134n^2 + 40n$	
n=8	6000	8896	32.5%

Final exponentiation : step complexity if the bit is zero

	RNS	Montgomery	Gain
Word-complexity	$18.5n^2 + 93n$	$48n^2 + 12n$	
n=8	1928	3168	39%

Some remarks

- RNS gain is essentially on $\mathbb{F}_{p^k}$ arithmetic, then
  - Choosing other systems of coordinate (affine, projective,...)
  - Using Ate pairing or other way to have shorter Miller loop

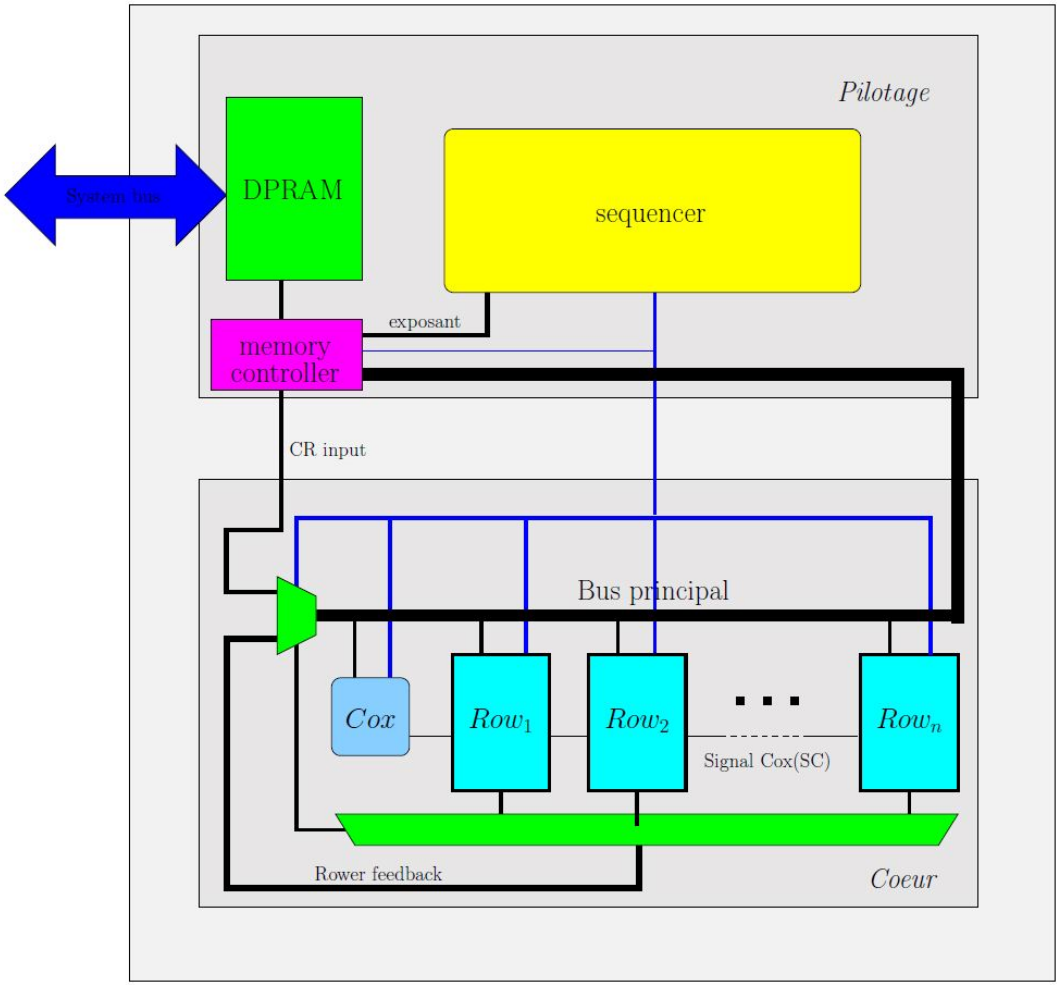
  will not change our conclusion that RNS arithmetic is interesting for pairing computations.
- Using curves with $\rho > 1$ will benefit to RNS since n takes larger values.
- Similar results are expected with Freeman curves ($k = 10$).
- Supersingular curves ($k = 2$) take also advantage of RNS arithmetic since $n = 16$ for 80 bits security level.
- Remember advantages of the RNS arithmetic become evident when a parallel architecture is used.
- A practical implementation is missing
  - to take into account the neglected operations (important because n an k are small),
  - to effectively compare with other implementations in the BN case.

Practical implementation

FPGA

- Programmable hardware device
- Xilinx and Altera
- 2 categories : "low cost" and "high end"
- Includes many logic modules containing LUT (Look-Up-Table) and some additional functions (allowing fast carry propagation or shift register for instance)
- An interconnecting array between logic modules
- DSP block for multiplication (9 bit words)

The Cox-Rower architecture



- Used in mot of RNS FPGA implementation
- based on parallelism
- Well adapted to RNS change of basis
- Need adjustments depending on the goal (RSA, ECC, pairings)
- The Cox computes λ
- A Rower computes modulo m_i

Our choices

- Optimal Ate pairing on BN curves defined by

$$\begin{array}{lcll} x & = & -(2^{62}+2^{55}+1) & \text{126 bits of security} \\ x & = & -(2^{63}+2^{22}+2^{18}+2^{7}+1) & \text{128 bits of security} \\ x & = & -(2^{160}+2^{74}+2^{12}+1) & \text{192 bits of security} \end{array}$$

  with almost all recent algorithmic improvements.
- Projective coordinates
- 36 bits words (allowing more than $192p^2$ as input of the reduction)
- No Karatsuba methods (addition have essentially the same cost as multiplications)

Results

contribution	Couplage	Securité [bit]	Plate-forme	Algorithme	taille	Freq. [MHz]	Cycles [$\times 10^3$]	latence [ms]
Design gén.	optimal ate	126	Altera FPGA (Cyclone II)	RNS Montgomery	14274 LE 35 mult.	91	176	1.93
			Altera FPGA (Stratix III)		4233 A 72 DSPs	165	176	1.07
		192	Altera FPGA (Stratix III)		9910 A 171 DSPs	131	790	6.03
Design opt.	optimal ate	126	Xilinx FPGA (Virtex-6)	RNS Montgomery	7032 slices 32 DSPs	250	143	0.573
[46]	ate	128	Xilinx FPGA (Virtex-6)	Hybrid Montgomery	4014 slices 42 DSPs	210	336	1.60
	optimal ate						245	1.17
[54]	Tate	128	Xilinx FPGA (Virtex-4)	Blakley	52k Slices	50	1,730	34.6
	ate						1,207	24.2
	optimal ate						821	16.4
[80]	Tate	128	ASIC (130 nm)	Montgomery	97 kGates	338	11,627*	34.4
	ate						7,706*	22.8
	optimal ate						5,340*	15.8
[45]	Tate over $\mathbb{F}_{3^{5 \cdot 97}}$	128	Xilinx FPGA (Virtex-4)	-	4755 Slices 7 BRAMs	192	429	2.23
[10]	optimal Eta over $\mathbb{F}_{2^{367}}$	128	Xilinx Virtex-4	-	4518 Slices	220	774*	3.52
[55]	η_T over $\mathbb{F}_{2^{1223}}$	128	Xilinx Virtex-6	-	15167 Slices	250	49	190
[77]	ate	128	64-bit Core2	Montgomery	-	2400	15,000	6.25
	optimal ate						10,000	4.17
[59]	ate	128	64-bit Core2	Montgomery		2400	14,429	6.01
[97]	optimal ate	128	Core2 Quad	Hybrid Mult.	-	2394	4,470	1.86
[25]	optimal ate	126	Core i7	Montgomery	-	2800	2,330	0.83
[12]	optimal ate	126	Phenom II	Montgomery	-	3000	1,562	0.52
[11]	η_T over $\mathbb{F}_{2^{1223}}$	128	Xeon	-	-	2000	3,020	1.51
[23]	η_T over $\mathbb{F}_{3^{509}}$	128	Core i7	-	-	2900	5,423	1.87
[10]	opt. Eta $\mathbb{F}_{2^{367}}$	128	Core i5	-	-	2530	2,440	0.96

Thank you

Thank you for your attention