

RNS-Representation of Numbers for Elliptic Curve Cryptography

Sylvain Duquesne

University of Rennes 1

Workshop on Pairings in Arithmetic Geometry and Cryptography,
Essen, 4-6 May 2009



- Standard prime field arithmetic
- RNS (Residue Number System) arithmetic
- Application to standard elliptic curve arithmetic
- Application to leak-resistant elliptic curve arithmetic
(Joint work with JC. Bajard and M. Ercegovic)
- Application to extension field arithmetic and pairing computations
(Work in progress)

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 $x^5 \bmod p$

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

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 : $q = -R \times p^{-1} \bmod \beta^n$ 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 $m_1, m_2, \dots, m_{n-1}, m_n$ relatively prime numbers and $M = \prod_{i=1}^n m_i$

We can represent $X \in [0, M]$ with (x_1, x_2, \dots, 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, \dots, m_n) is named RNS base, we denote it \mathcal{B}_n

Operations

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

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

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 $AM^{-1} \bmod p$

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 $r < 2p$
 $r = R\beta^{-n} \bmod p$

Data $R = AB$ in \mathcal{B}_n
 $-p^{-1}$ in \mathcal{B}_n

Compute $q = -R \times p^{-1}$ in \mathcal{B}_n
 $r = (R + q \times p)M^{-1}$ in \mathcal{B}_n

Result $r < (n+2)p$
 $r = RM^{-1} \bmod p$

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 $AM^{-1} \bmod p$

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 $r < 2p$
 $r = R\beta^{-n} \bmod p$

Data $R = AB$ in \mathcal{B}_n
 $-p^{-1}$ in \mathcal{B}_n

Compute $q = -R \times p^{-1}$ in \mathcal{B}_n
 $r = (R + q \times p)M^{-1}$ in \mathcal{B}_n

Result $r < (n+2)p$
 $r = RM^{-1} \bmod p$

Computation modulo p replaced by

computation modulo β^n

computation modulo M

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	$r < (n + 2)p$ $r = RM^{-1} \bmod p$

Complexity

After some improvements, 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.

→ Try to optimize ECC formulas to take advantage of this.

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.

Leak-resistance in elliptic curve cryptography

Context : E elliptic curve defined over \mathbb{F}_p , $P \in E(\mathbb{F}_p)$, r an integer.
Central operation in elliptic curve cryptography is the computation of $[r]P$.
It must be fast and secure.

leak-resistance

Leaks coming from a processor (power consumption, electro-magnetic radiations or even noise) can be analyzing.

⇒ The operations performed must be independent of the secret (here r).

With elliptic curves, basic operations (addition and doubling) are different.

⇒ Weakness

Solutions

- Use unified formulas
- Use Montgomery formulas

Principle

- y -coordinate bring only minor information, so forget it
- $P - Q$ must be known to compute $P + Q$
- Use an adapted algorithm
- Compute both $P + Q$ and $[2]P$ at each step \rightarrow leak-resistant

Known formulas

- Montgomery gives formulas for a family of curves (1987)
- Brier and Joye give formulas for any curve (2002)

General formulas required 19 multiplications and 16 reductions

No more exciting than standard elliptic curve arithmetic for RNS.

The Kummer variety

Use a mathematical point of view

- forgotten the y -coordinate \Leftrightarrow Take the quotient of E by the hyperelliptic involution
 \Leftrightarrow Work on the Kummer variety

The biquadratic forms

On the Kummer variety, traces of the group law remain :

- Addition of a 2-torsion point
- Doubling
- Biquadratics forms M_x , M_z and M_{xz} such that for any points given in projective coordinate $P = (X_p, Z_p)$ and $Q = (X_q, Z_q)$

$$\begin{aligned}2X_{p+q}X_{p-q} &= M_x \\X_{p+q}Z_{p-q} + X_{p-q}Z_{p+q} &= M_{xz} \\2Z_{p+q}Z_{p-q} &= M_z\end{aligned}$$

New formulas for Montgomery arithmetic

Use equations 1 and 3

$$X_{p+q} = (2X_{p-q})^{-1} M_x$$

$$Z_{p+q} = (2Z_{p-q})^{-1} M_z$$

⇒ Find again Brier and Joye formulas

Use equations 2 and 3

$$X_{p+q} = M_{xz} - X_{p-q} Z_{p+q}$$

$$Z_{p+q} = (2Z_{p-q})^{-1} M_z$$

⇒ Find new formulas requiring

13 reductions and 20 multiplications

Word-complexity comparisons

Usual Montgomery arithmetic requires $38n^2 + 19n$ operations.
But RNS arithmetic on new formulas requires only

$$20 \times 2n + 13 \times (2n^2 + 3n) = 26n^2 + 79n$$

bit size of p	n	RNS complexity	Montgomery complexity
160	5	1045	1045
192	6	1410	1482
256	8	2296	2584
512	16	7920	10032

For unified formulas, the result obtained is comparable.

Finally, using RNS in leak-resistant elliptic curve cryptography is

- interesting especially for high security level,
- very interesting if parallel architecture is used.

Context

- E elliptic curve defined over \mathbb{F}_p (p prime).
- $P \in E(\mathbb{F}_p)$ of prime order r (as sparse as possible or replace it by sparse mr).
- $k = 2d$ 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 $\mathbb{F}_{p^k} = \mathbb{F}_{p^d}[\alpha]$ (use twist of order 2).

Optimised Tate pairing

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

The Miller loop (computation of $f_P(Q)$ up to \mathbb{F}_{p^d} factors)

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

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

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 roughly 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 $C = 4X_TY_T^2$.

Computation of $2T$ requires 10M and 8R.

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

It is easy to prove that

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

and that its computation requires

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

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

No more exciting than standard elliptic curve arithmetic for RNS.

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 $f = f_0 + f_1z + f_2z^2$ and $g = g_0 + g_1z + g_2z^2$ in \mathbb{F}_{p^3} , then

$$fg = (f_0g_0 + f_1g_2\beta + f_2g_1\beta) + (f_0g_1 + f_1g_0 + f_2g_2\beta)z + (f_0g_2 + f_2g_0 + f_1g_1)z^2$$

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

\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 10M 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	$12n^2 + 38n$	$20n^2 + 10n$
n=5	490	550
n=16	3680	5280

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.

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

	RNS	Montgomery
Word-complexity	$56n^2 + 175n$	$94n^2 + 28n$
n=5	2275	2490
n=16	17136	24512

Final exponentiation : step complexity if the bit is zero

	RNS	Montgomery
Word-complexity	$12n^2 + 38n$	$20n^2 + 10n$
n=5	490	550
n=16	3680	5280

$\mathbb{F}_{p^{12}}$ 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 20M and 12R

	RNS	Montgomery
Word-complexity for $f.g$ n=8	$24n^2 + 144n$ 2688	$108n^2 + 54n$ 7344
Word-complexity for f^2 n=8	$24n^2 + 76n$ 2144	$40n^2 + 20n$ 2720

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 $\ell_{T,T}(Q)$
- 20M and 12R for the squaring of f
- **27M** and 12R for the multiplication of f^2 and $\ell_{T,T}(Q)$

	RNS	Montgomery
Word-complexity	$80n^2 + 252n$	$132n^2 + 66n$
n=8	7136	8976

Final exponentiation : step complexity if the bit is zero

	RNS	Montgomery
Word-complexity	$24n^2 + 76n$	$40n^2 + 20n$
n=8	2144	2720

Some concluding 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 loopwill not change our conclusion that RNS arithmetic is interesting for pairing computations.
- Using curves with $\rho > 1$ will benefit to RNS since n take 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 and k are small),
 - to precise the situation with $n = 5$ and $k = 6$,
 - to effectively compare with Devegili, Scott and Dahab implementation in the BN case.

Thank you

Thank you for your attention