RNS Modular Arithmetic: Introduction and Cryptographic Applications

Karim Bigou

CNRS – IRISA – CAIRN

May 29, 2015

Plan

1. Context
2. RNS for Cryptographic Computations
3. New RNS Modular Multiplication
4. Specific Patterns for Exponentiations

Context

One objective of our research group:

Design efficient hardware implementations of asymmetric cryptography using fast arithmetic techniques

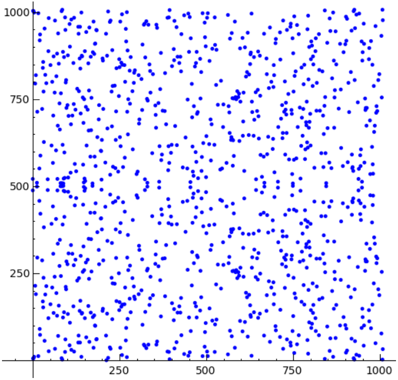
Examples of targetted cryptosystems:

- RSA [RSA78]
- Discrete Logarithm Cryptosystems: Diffie-Hellman [DH76] (DH), ElGamal [Elg85]
- Elliptic Curve Cryptography (ECC) [Mil85] [Kob87]

The **residue number system** (RNS) is a representation which enables fast computations for cryptosystems requiring large integers (or $\mathbb{F}_P$ elements)

ECC Very Short Overview

P large prime of 160–600 bits



$y^2 = x^3 + 4x + 20$ over $\mathbb{F}_{1009}$

Elliptic curve E over $\mathbb{F}_P$:

$$y^2 = x^3 + a\,x + b$$

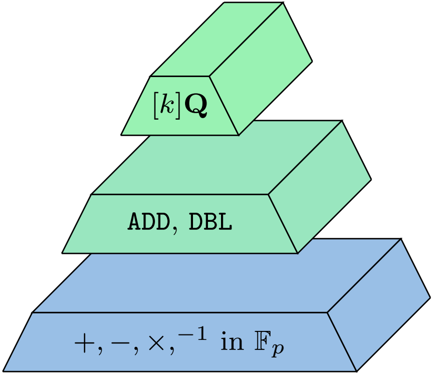
Curve level operations:

- Point addition (ADD): $\mathbf{Q} + \mathbf{Q'}$
- Point doubling (DBL): $\mathbf{Q} + \mathbf{Q}$
- Scalar multiplication: $[k]\mathbf{Q} = \underbrace{\mathbf{Q} + \mathbf{Q} + \ldots + \mathbf{Q}}_{k \text{ times}}$

Security (ECDLP): knowing $\mathbf{Q}$ and $[k]\mathbf{Q}$, k cannot be recovered

ECDLP : Elliptic Curve Discrete Logarithm Problem

Internal Operations of a Scalar Multiplication

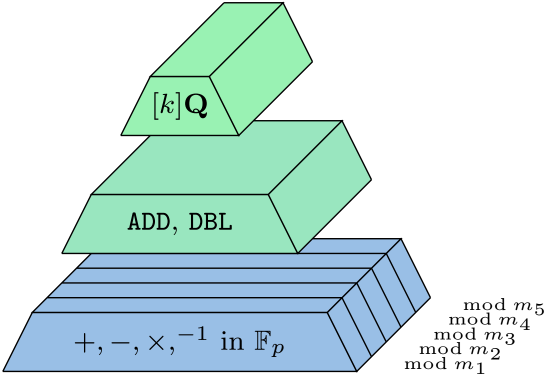


One scalar multiplication requires...

Many curve level operations which require...

MANY $\mathbb{F}_P$ operations

Internal Operations of a Scalar Multiplication



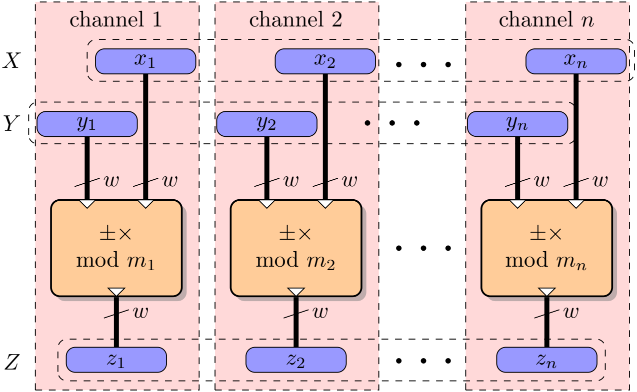
One scalar multiplication requires...

Many curve level operations which require...

MANY $\mathbb{F}_P$ operations which can be performed in a parallel way using RNS

Residue Number System (RNS) [SV55] [Gar59]

X a large integer of ℓ bits ($\ell \approx$ 160–4096) is represented by:
$\overrightarrow{X} = (x_1, \ldots, x_n) = (X \bmod m_1, \ldots, X \bmod m_n)$



RNS base $\mathcal{B} = (m_1, \ldots, m_n)$, n pairwise co-primes of w bits, $n \times w \geqslant \ell$
The Chinese remainder theorem (CRT) is the base of RNS

Note: an EMM is a w-bit elementary modular multiplication (one channel)

RNS Properties

Pros:

- **Carry free** between channels
  - each channel is independant
- **Fast parallel** $+$, $-$, $\times$ and some exact divisions
  - computations over all channels can be performed in parallel
  - an RNS multiplication requires *n* EMMs
- **Non-positional** number system
  - randomization of internal computations (SCA countermeasures)
- **Flexibility** for hardware implementations
  - the number of hardware channels and theoretical channels can be different
  - various area/ time trade-offs and multi-size support

Cons:

- comparison, **modular reduction** and division are much harder

A Very Brief and Very Non-Exhaustive History of RNS for Cryptographic Implementations

- First motivation: parallel implementation of RSA
- Need for efficient modular reduction: [PP95, BDK98]
- Lead to RNS implementations of RSA [KKSS00, NMSK01]
- A protection based on randomization is proposed: the Leak Resistant Arithmetic (LRA) [BILT04]
- Ideas are adapted and reused for ECC and Pairings [Gui10, CDF$^+$11, YFCV12]

Now:

- New algorithms, new selection of parameters for RNS arithmetic [GLP$^+$12, BDE13, BT13, YFCV14, BT14]
- New protections based on RNS [Gui11, BEG13, PITM13, NP15]
- New architectures [BM14]
- New applications [BEMP14]

Base Extension [ST67]

Issue:
computing a reduction modulo a large number P from the small residues

- Usual technique for modular reduction:
  Use conversions between 2 bases
- $\mathcal{B} = (m_1, \ldots, m_n)$ and $\mathcal{B}' = (m'_1, \ldots, m'_n)$ are coprime RNS bases
- X is $\overrightarrow{X}$ in $\mathcal{B}$ and $\overrightarrow{X}'$ in $\mathcal{B}'$
- The base extension (`BE`, introduced in [ST67]) is defined by:

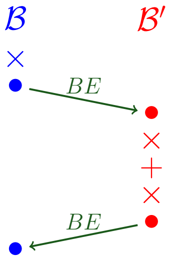
$$\overrightarrow{X}' = \mathtt{BE}(\overrightarrow{X}, \mathcal{B}, \mathcal{B}')$$

- Some operations become possible after a base extension
  - $M = \prod_{i=1}^{n} m_i$ is invertible in $\mathcal{B}'$
  - exact division by M can be done easily
- State-of-art `BE` algorithms cost $n^2 + n$ `EMMs`

RNS Montgomery Reduction (MR) [PP95]

Input: $\overrightarrow{X}$, $\overrightarrow{X}'$ with $X < \alpha P^2 < PM$ and $2P < M'$
Output: $(\overrightarrow{\omega}, \overrightarrow{\omega}')$ with $\omega \equiv X \times M^{-1} \bmod P$
$0 \leqslant \omega < 2P$

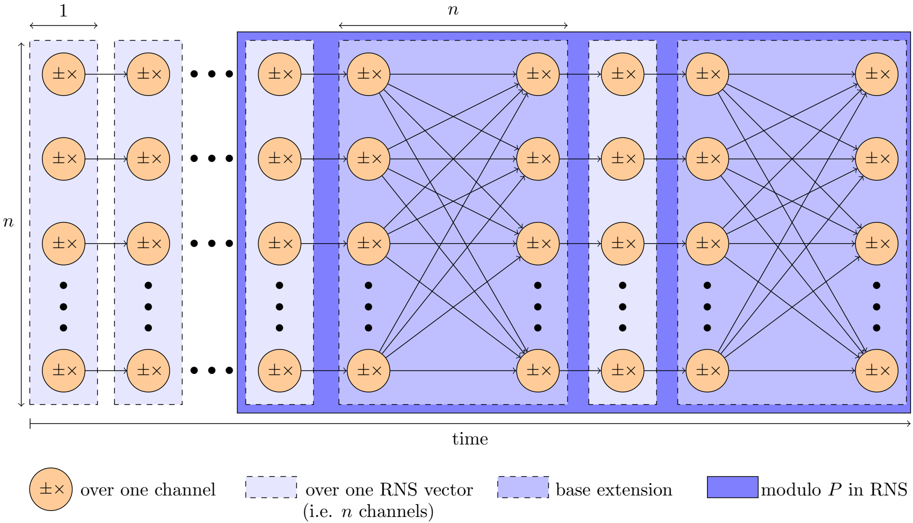
$\overrightarrow{Q} \longleftarrow \overrightarrow{X} \times (-\overrightarrow{P}^{-1})$ (in base $\mathcal{B}$)
$\overrightarrow{Q}' \longleftarrow \mathtt{BE}(\overrightarrow{Q}, \mathcal{B}, \mathcal{B}')$
$\overrightarrow{S}' \longleftarrow \overrightarrow{X}' + \overrightarrow{Q}' \times \overrightarrow{P}'$ (in base $\mathcal{B}'$)
$\overrightarrow{\omega}' \longleftarrow \overrightarrow{S}' \times \overrightarrow{M}^{-1}$ (in base $\mathcal{B}'$)
$\overrightarrow{\omega} \longleftarrow \mathtt{BE}(\overrightarrow{\omega}', \mathcal{B}', \mathcal{B})$



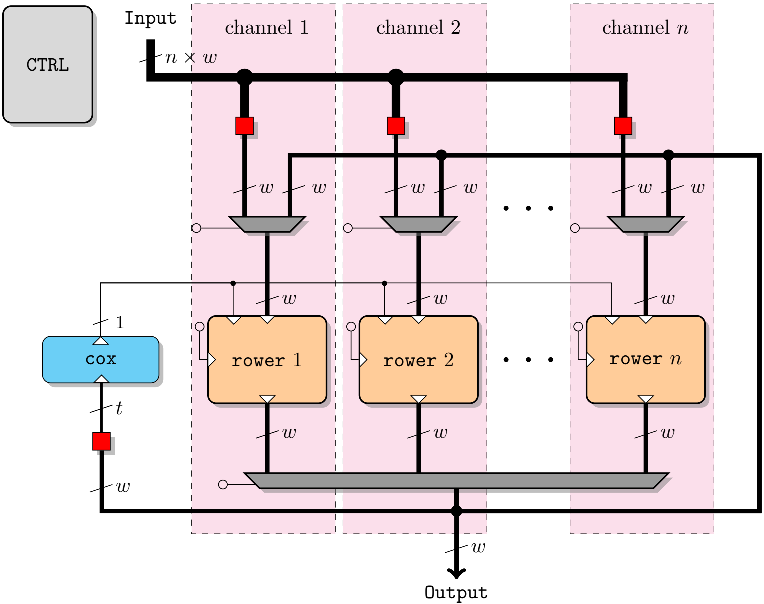
α is a parameter chosen to speed up some computations, $M > \alpha P$ and $M' > 2 \times P$

MR cost: $\boxed{2\,n^2 + O(n)}$ EMMs

Typical RNS Computation Flow



Cox-Rower RNS Architecture [KKSS00, Gui10]



Some Implementation Results of 256-bit ECC on FPGA

ref.	[GP08]	[MLPJ13]		[Gui10] (RNS)	[BM14] (RNS)
prime	NIST	Any		Any	Any
FPGA	Virtex 4	Virtex 4	Virtex 5	Stratix II	Kintex 7
# Slices	1715	4655	1725	9177*	1630
# DSPs	32	37	37	96*	46
Freq. MHz	490/245**	250	291	157	281
time ms	0.62	0.44	0.38	0.68	0.61
$[k]\mathbf{P}$ Algo.	DBL & ADD	Möller [Mö01]		Mont. ladd. [JY02]	

* : Stratix II FPGA is counted in ALM instead of Slices and 9×9 multiplier instead of Xilinx DSP (18×25)

**: [GP08] uses 2 clock domains: 490 MHz for arithmetic and 245 MHz for control

RNS as a SCA protection

Protections based on randomization

- [CNPQ03] proposes to randomly choose the 2 RNS bases in a large set of moduli (*e.g.* 2 bases of 9 moduli in a set of 69)
- [BILT04] introduces the Leak Resistant Arithmetic (**LRA**):
  - at the beginning both bases are chosen randomly from $2n$ moduli (*i.e* once)
  - Very costly if used at each `MR`
- [Gui11] adapts LRA to the Kawamura *et al.* base extension
- [PITM13] implements LRA and an initial base permutation against EMA attacks
- [NP15] implements a trade-off in LRA usable for each `MR`

Fault detection using redundancy, *e.g.* [WH66, Man72, YL73, CNPQ03] and recently adapted to cryptographic implementations [Gui11, BEG13]

How to Speed up RNS computations for Cryptography?

Two main ideas to reduce the impact of modular reductions:

- Reduce the cost of modular reduction in specific contexts, for instance:
  - rearranging computations in an ECC context [Gui10]
  - rearranging computations in RSA exponentiation context [GLP+12]
  - our proposed modular multiplication algorithms [BT15, BT14] and new exponentiation algorithms for discrete logarithm and RSA
- Reduce the number of modular reductions, for instance:
  - computing pattern of the form $AB + CD \bmod P$ in ECC formulas [BDE13]
  - our proposed modular inversion algorithm `PM-MI` in an ECC context [BT13]

New RNS Modular Multiplication

Improving Modular Multiplication

RNS modular multiplication MM is the most costly operation in RNS cryptographic applications (ECC, RSA, DL)

Two different multiplications:

- simple RNS multiplication : n EMMs
- MM = simple RNS multiplication + MR : $2n^2 + O(n)$ EMMs

Our idea: modify RNS to add some positional information

Let us assume $\mathcal{B}_a$ with $\frac{n}{2}$ moduli of w bits ($\log_2 P \approx n \times w$, $\mathcal{B}_a$ is a "half base"), then $(\overrightarrow{K_x}, \overrightarrow{R_x})$ represents:

$$\overrightarrow{X} = \overrightarrow{K_x}\,\overrightarrow{M_a} + \overrightarrow{R_x}$$

where $M_a = \prod_{i=1}^{n_a} m_{a,i}$

Note: K_x and R_x are $\frac{\log_2 P}{2}$ bits long

Decomposition with Split Algorithm

Input: $\overrightarrow{X_{a|b}}$

Precomp.: $\overrightarrow{\left(M_a^{-1}\right)_b}$

Output: $\overrightarrow{(K_x)_{a|b}}$, $\overrightarrow{(R_x)_{a|b}}$ with $\overrightarrow{X_{a|b}} = \overrightarrow{(K_x)_{a|b}} \times \overrightarrow{(M_a)_{a|b}} + \overrightarrow{(R_x)_{a|b}}$

$\overrightarrow{(R_x)_b} \leftarrow \text{BE}\left(\overrightarrow{(R_x)_a}, \mathcal{B}_a, \mathcal{B}_b\right)$

$\overrightarrow{(K_x)_b} \leftarrow \left(\overrightarrow{X_b} - \overrightarrow{(R_x)_b}\right) \times \overrightarrow{\left(M_a^{-1}\right)_b}$

if $\overrightarrow{(K_x)_b} = \overrightarrow{-1}$ **then**

  $\overrightarrow{(K_x)_b} \leftarrow \overrightarrow{0}$ /* *Kawamura BE correction* */

  $\overrightarrow{(R_x)_b} \leftarrow \overrightarrow{(R_x)_b} - \overrightarrow{(M_a)_b}$

$\overrightarrow{(K_x)_a} \leftarrow \text{BE}\left(\overrightarrow{(K_x)_b}, \mathcal{B}_b, \mathcal{B}_a\right)$

return $\overrightarrow{(K_x)_{a|b}}$, $\overrightarrow{(R_x)_{a|b}}$

Note: the cost of Split is dominated by the 2 BEs (on half bases) :

$$\frac{n^2}{2} + O(n) \text{ when } n_a = n_b = n/2$$

Use Decomposition

SBMM (Single Base Modular Multiplication) idea:

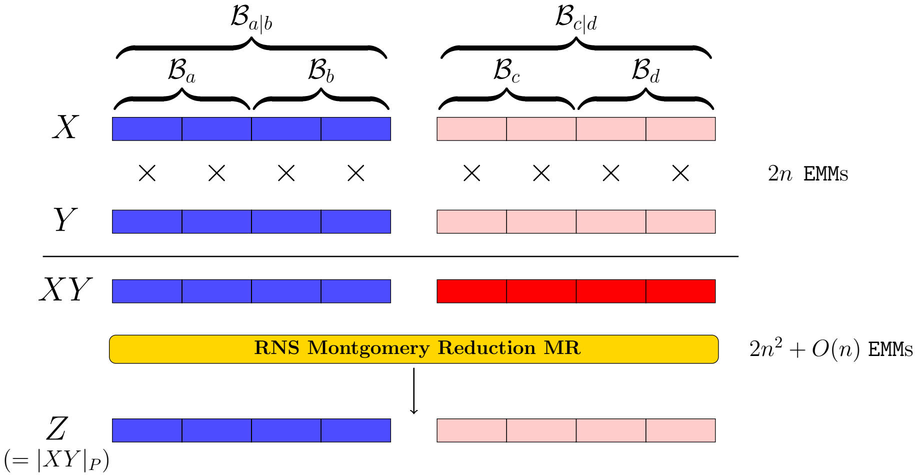
- X is represented by (K_x, R_x)
- $P = M_a^2 - 2$ with P prime and M_a odd

Some remarks

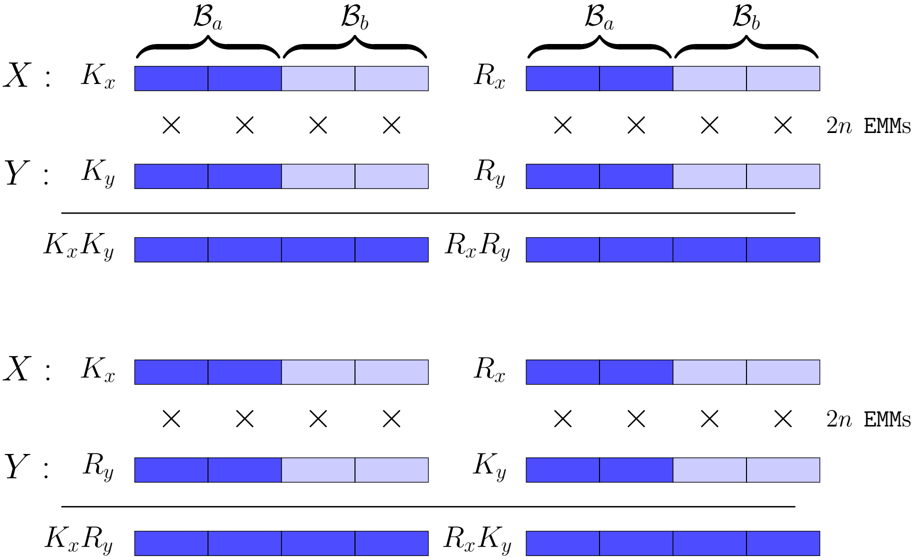
- P is an equivalent for RNS to pseudo-Mersenne numbers for the radix 2 standard representation (for instance $P = 2^{521} - 1$)
- $P = M_a^2 - 1$ is never prime
- One can find a lot of P for a given size (probabilistic primality tests using `isprime` from Maple, for instance generating 10 000 P of 512 bits in 15 s.)

Classical RNS MM principle

$\mathcal{B}_{a|b}$, $\mathcal{B}_{c|d}$: full RNS bases $\mathcal{B}_a$, $\mathcal{B}_b$, $\mathcal{B}_c$, $\mathcal{B}_d$: half bases



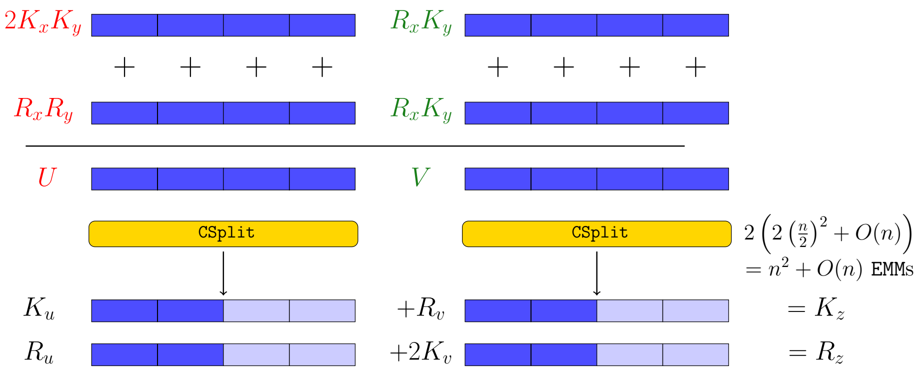
SBMM Principle 1/2



$$XY \equiv 2K_xK_y + (K_xR_y + K_yR_x)M_a + R_xR_y \equiv U + VM_a \bmod P$$

SBMM Principle 2/2

$$XY \equiv U + V M_a \equiv (K_u + R_v) M_a + (R_u + 2K_v) \equiv \boxed{K_z}\ M_a + \boxed{R_z} \bmod P$$



$\texttt{SBMM}$ Algorithm

Parameters: $\mathcal{B}_a$ such that $M_a^2 = P + 2$ and $\mathcal{B}_b$ such that $M_b > 6M_a$

Input: $\overrightarrow{(K_x)_{a|b}}$, $\overrightarrow{(R_x)_{a|b}}$, $\overrightarrow{(K_y)_{a|b}}$, $\overrightarrow{(R_y)_{a|b}}$ with K_x, R_x, K_y, $R_y < M_a$

Output: $\overrightarrow{(K_z)_{a|b}}$, $\overrightarrow{(R_z)_{a|b}}$ with $K_z < 5M_a$ and $R_z < 6M_a$

$\overrightarrow{U_{a|b}} \leftarrow \overrightarrow{2K_xK_y + R_xR_y}$

$\overrightarrow{V_{a|b}} \leftarrow \overrightarrow{K_xR_y + R_xK_y}$

$\left(\overrightarrow{(K_u)_{a|b}}, \overrightarrow{(R_u)_{a|b}}\right) \leftarrow \texttt{Split}(\overrightarrow{U_{a|b}})$

$\left(\overrightarrow{(K_v)_{a|b}}, \overrightarrow{(R_v)_{a|b}}\right) \leftarrow \texttt{Split}(\overrightarrow{V_{a|b}})$

$\left(\overrightarrow{(K_z)_{a|b}}, \overrightarrow{(R_z)_{a|b}}\right) \leftarrow \left(\overrightarrow{(K_u + R_v)_{a|b}}, \overrightarrow{(2 \cdot K_v + R_u)_{a|b}}\right)$

return $\left(\overrightarrow{(K_z)_{a|b}}, \overrightarrow{(R_z)_{a|b}}\right)$

M_b is a few bits larger than M_a because outputs K_z and R_z are larger than inputs K_x, K_y, R_x, R_y

SBMM Algorithm

Using an extra modulo m_γ in $\mathcal{B}_b$:

- one can have $M_b > 6M_a$
- it enables to compress output values from SBMM
- it can be chosen small (e.g. $m_\gamma = 2^6$)

Algo.	MM [GLP+12]	SBMM	SBMM + Compress
EMM	$\mathbf{2n^2} + 4n$	$\mathbf{n^2} + 5n$	$(\mathbf{n^2} + 7n)$ EMM $+ (n+2)$ GMM
Precomp. EMW	$\mathbf{2n^2} + 10n$	$\mathbf{\frac{n^2}{2}} + 3n$	$\mathbf{\frac{n^2}{2}} + 4n + 2$

EMM is a w-bit modular multiplication

GMM is a one multiplication modulo m_γ (6 bits in practice)

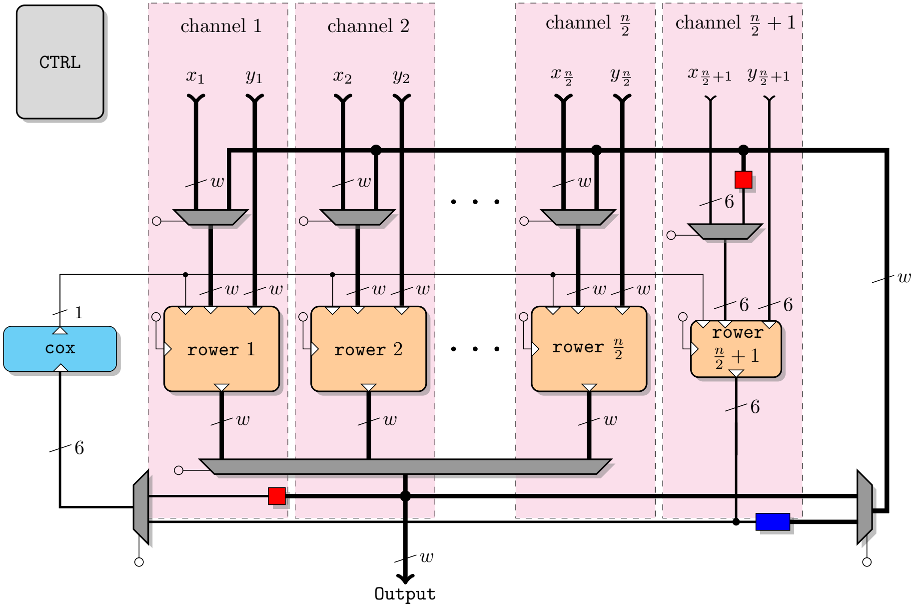
EMW is a w-bit word stored as a precomputation

Implementations

FPGA implementations:

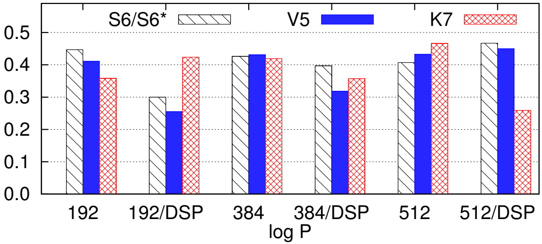
- `MM` and `SBMM` have been implemented
- n `Rowers` for `MM` and $n/2$ `Rowers` for `SBMM`
- 3 field lengths implemented: 192, 384 and 512 bits
- $w = 16$ bits for 192 and 32 for 384 and 512
- on various FPGAs
  - high performance Virtex 5 (LX220)
  - low cost Spartan 6 (LX45/LX100)
  - recent mid-range Kintex 7 (70T)
- (parallel) compression not implemented yet

SBMM Architecture with $n/2$ Rowers

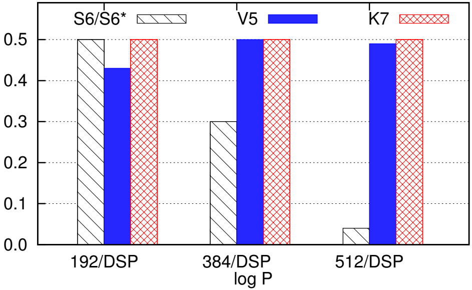


FPGA Implementation Results

Reduction in Slices
(e.g. 0.4 is -40%)

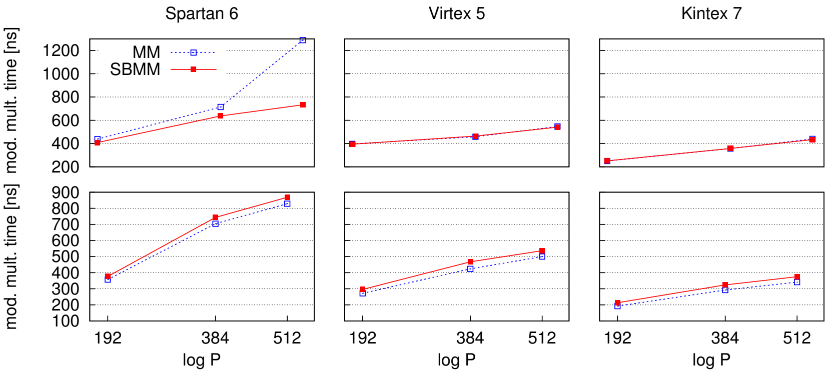


Reduction in DSP blocks



FPGA Implementation Results

Timing results for a single modular multiplication with (top) and without (bottom) DSP blocks



Conclusion on SBMM

Theoretical conclusions:

- # EMMs / 2
- # precomputations / 4
- # moduli / 2
- the architecture is still flexible

First implementations conclusions:

- the area is almost divided by 2 for a small time overhead ($< 10\,\%$)

Further implementation works:

- n Rowers for SBMM (full parallel implementation)
- integration in a full scalar multiplication

This work will be presented at CHES 2015 (September in Saint-Malo)

Specific Patterns for Exponentiations

Idea

Goal: accelerate some specific, but usual, computation patterns which uses RNS modular multiplications

Examples:

- modular squares
- modular multiplication by constants
- more complex patterns with operands reuse

In state-of-the-art, RNS does not support accelerations for these patterns (except accelerations inside channels)

A Specific Fast Pattern

The cost of some patterns can be reduced without constraint on the field characteristic, for instance in the following algorithm [Gor98] :

Input: $k = (k_{\ell-1}, \ldots, k_1, k_0)_2,\ G \in \mathbb{Z}/P\mathbb{Z}$
Output: $G^k \bmod P$
$S \leftarrow 1$
for i **from** $\ell - 1$ **to** *0* **do**
  $S \leftarrow S^2 \bmod P$
  if $k_i = 1$ **then** $S \leftarrow S \cdot G \bmod P$
return S

One can observe:

$$\begin{aligned} S^2 G &\equiv \left(K_s^2 M_a^2 + 2K_s R_s M_a + R_s^2\right) G \bmod P \\ &\equiv K_s^2 |M_a^2 G|_P + K_s R_s |2 M_a G|_P + R_s^2 |G|_P \bmod P \\ &\equiv K_s \left(K_s |M_a^2 G|_P + R_s |2 M_a G|_P\right) + R_s^2 |G|_P \bmod P \end{aligned}$$

A Specific Fast Pattern

Values $|M_a^2 G|_P$, $|2M_a G|_P$ and $|G|_P$ can be precomputed

We choose $\mathcal{B}_a$ with $n/2$ moduli of w bits then K_s and R_s are $\ell/2$-bit values (*i.e.* the same size as $\sqrt{P}$)

If $U_2 = K_s \left(K_s|M_a^2 G|_P + R_s|2M_a G|_P\right) + R_s^2|G|_P$ then $\log_2 U_2 \approx 2\ell$ *i.e.* U_2 is a partially reduced value

Finally, we use the state-of-the-art `MR` to finish the modular reduction

The total cost of $|S^2 G|_P$:

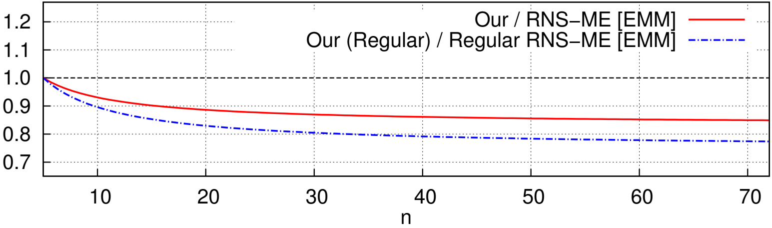
- the `Split` mainly costs n^2 `EMMs` ...
- and the final `MR` mainly costs $2n^2$ `EMMs` ...
- leading to $3n^2$ `EMMs`

The same pattern is computed with $4n^2$ `EMMs` in state-of-the-art of RNS

Exponentiation Algorithm

Average values for 2 bits of key (one 1 and one 0):

Algo.	EMM	EMW
Our algorithm	$\mathbf{5n^2 + 17n}$	$\mathbf{3n^2 + 20n}$
RNS-ME [GLP+12]	$6n^2 + 12n$	$2n^2 + 10n$
Our algorithm (regular)	$\mathbf{6n^2 + 26n}$	$\mathbf{3n^2 + 26n}$
Regular RNS-ME [GLP+12]	$8n^2 + 16n$	$2n^2 + 10n$



Conclusion on Fast Patterns

Our proposed modular exponentiation:

- reduces the number of `EMMs` up to 15 % for the non regular algorithm
- reduces the number of `EMMs` up to 22 % for the regular version
- can be easily adapted into a windowed version

Future works:

- implementations of the propositions in full cryptosystems
- time$\times$area trade-off explorations
- analysis of other patterns
- analysis of the use of this pattern in other cryptosystems (e.g. ECC)

Other Published Works on RNS

Proposition SPRR (presented at ASAP 2014) :

Combines Split and MR on reduced bases

- gain in EMMs depends on the reuse of operands in operation sequences (up to **10%** less EMMs)
- gain in precomputations of **25 %**
- works for discrete logarithm and ECC

Proposition PM-MI (presented at CHES 2013):

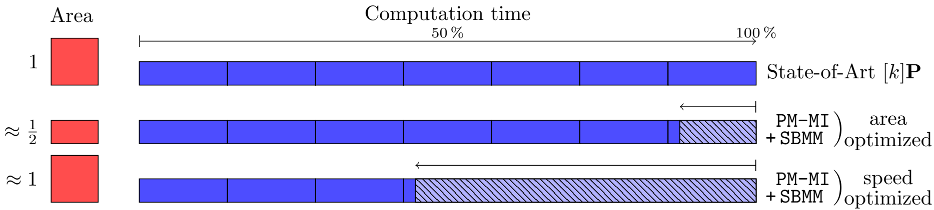
Adapts the binary extended Euclidean algorithm for RNS using the plus-minus trick

- it does not require BE
- it significantly reduces the number of EMM: # EMMs divided **by 10–20**
- PM-MI and state-of-art algorithm have been implemented on FPGA
  - PM-MI is **5–12 times faster**
  - with a **small** area overhead on RNS operator for ECC

Conclusion

Conclusion of our contributions

Objective for a full RNS ECC implementation:



Several aspects of our propositions still have to be studied:

- a complete ECC cryptoprocessor in RNS implementation
- flexibility of the Cox-Rower architecture
- compatibility with the countermeasures based on RNS

General Conclusion

- RNS is interesting thanks to several natural properties (e.g. parallelism, randomization)
- the relative costs between the different operations are not the same in RNS compared to the usual binary system
  - We have to count differently: *e.g.* in [BDE13] one has point ADD faster than DBL!
- there is a lot of lines of research to improve the use of RNS for cryptographic applications
  - choice of parameters (e.g. moduli, curve parameters . . .)
  - new algorithms
  - new architectures

Thank you for your attention

References I

[BDE13] J.-C. Bajard, S. Duquesne, and M. D. Ercegovac.
Combining leak-resistant arithmetic for elliptic curves defined over Fp and RNS representation.
Publications Mathématiques UFR Sciences Techniques Besançon, pages 67–87, 2013.

[BDK98] J.-C. Bajard, L.-S. Didier, and P. Kornerup.
An RNS Montgomery modular multiplication algorithm.
IEEE Transactions on Computers, 47(7):766–776, July 1998.

[BEG13] J.-C. Bajard, J. Eynard, and F. Gandino.
Fault detection in RNS Montgomery modular multiplication.
In *Proc. 21th Symposium on Computer Arithmetic (ARITH)*, pages 119–126. IEEE, April 2013.

[BEMP14] J.-C. Bajard, J. Eynard, N. Merkiche, and T. Plantard.
Babaï round-off CVP method in RNS: Application to lattice based cryptographic protocols.
In *Proc. 14th International Symposium on Integrated Circuits (ISIC)*, pages 440–443. IEEE, December 2014.

References II

[BILT04] J.-C. Bajard, L. Imbert, P.-Y. Liardet, and Y. Teglia.
Leak resistant arithmetic.
In *Proc. Cryptographic Hardware and Embedded Systems (CHES)*, volume 3156 of *LNCS*, pages 62–75. Springer, 2004.

[BM14] J.-C. Bajard and N. Merkiche.
Double level Montgomery Cox-Rower architecture, new bounds.
In *Proc. 13th Smart Card Research and Advanced Application Conference (CARDIS)*, LNCS. Springer, November 2014.

[BT13] K. Bigou and A. Tisserand.
Improving modular inversion in RNS using the plus-minus method.
In *Proc. 15th Cryptographic Hardware and Embedded Systems (CHES)*, volume 8086 of *LNCS*, pages 233–249. Springer, August 2013.

[BT14] K. Bigou and A. Tisserand.
RNS modular multiplication through reduced base extensions.
In *Proc. 25th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 57–62. IEEE, June 2014.

References III

[BT15] K. Bigou and A. Tisserand.
Single base modular multiplication for efficient hardware rns implementations of ecc.
In *Proc. 17th Cryptographic Hardware and Embedded Systems (CHES)*, LNCS. Springer, 2015.

[CDF+11] R. C. C. Cheung, S. Duquesne, J. Fan, N. Guillermin, I. Verbauwhede, and G. X. Yao.
FPGA implementation of pairings using residue number system and lazy reduction.
In *Proc. 13th Cryptographic Hardware and Embedded Systems (CHES)*, volume 6917 of *LNCS*, pages 421–441. Springer, September 2011.

[CNPQ03] M. Ciet, M. Neve, E. Peeters, and J.-J. Quisquater.
Parallel FPGA implementation of RSA with residue number systems - can side-channel threats be avoided?
In *Proc. 46th Midwest Symposium on Circuits and Systems (MWSCAS)*, volume 2, pages 806–810. IEEE, December 2003.

[DH76] W. Diffie and M. E. Hellman.
New directions in cryptography.
IEEE Transactions on Information Theory, 22(6):644–654, November 1976.

[Elg85] T. Elgamal.
A public key cryptosystem and a signature scheme based on discrete logarithms.
IEEE Transactions on Information Theory, 31(4):469–472, July 1985.

References IV

[Gar59] H. L. Garner.
The residue number system.
IRE Transactions on Electronic Computers, EC-8(2):140–147, June 1959.

[GLP+12] F. Gandino, F. Lamberti, G. Paravati, J.-C. Bajard, and P. Montuschi.
An algorithmic and architectural study on Montgomery exponentiation in RNS.
IEEE Transactions on Computers, 61(8):1071–1083, August 2012.

[Gor98] D. M. Gordon.
A survey of fast exponentiation methods.
Journal of algorithms, 27(1):129–146, 1998.

[GP08] T. Güneysu and C. Paar.
Ultra high performance ECC over NIST primes on commercial FPGAs.
In *Proc. Cryptographic Hardware and Embedded Systems (CHES)*, volume 5154 of *LNCS*, pages 62–78. Springer, 2008.

[Gui10] N. Guillermin.
A high speed coprocessor for elliptic curve scalar multiplications over $\mathbb{F}_p$.
In *Proc. 12th Cryptographic Hardware and Embedded Systems (CHES)*, volume 6225 of *LNCS*, pages 48–64. Springer, August 2010.

References V

[Gui11] N. Guillermin.
A coprocessor for secure and high speed modular arithmetic.
Technical Report 354, IACR Cryptology ePrint Archive, 2011.

[JY02] M. Joye and S.-M. Yen.
The Montgomery powering ladder.
In *Proc. 4th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, volume 2523 of *LNCS*, pages 291–302. Springer, August 2002.

[KKSS00] S. Kawamura, M. Koike, F. Sano, and A. Shimbo.
Cox-Rower architecture for fast parallel Montgomery multiplication.
In *Proc. 19th International Conference on the Theory and Application of Cryptographic (EUROCRYPT)*, volume 1807 of *LNCS*, pages 523–538. Springer, May 2000.

[Kob87] N. Koblitz.
Elliptic curve cryptosystems.
Mathematics of computation, 48(177):203–209, 1987.

[Man72] D. Mandelbaum.
Error correction in residue arithmetic.
IEEE Transactions on Computers, 100(6):538–545, 1972.

References VI

[Mil85] V. Miller.
Use of elliptic curves in cryptography.
In *Proc. 5th International Cryptology Conference (CRYPTO)*, volume 218 of *LNCS*, pages 417–426. Springer, 1985.

[MLPJ13] Y. Ma, Z. Liu, W. Pan, and J. Jing.
A high-speed elliptic curve cryptographic processor for generic curves over GF(p).
In *Proc. 20th Selected Areas in Cryptography(SAC)*, LNCS, pages 421–437. Springer, 2013.

[Mö01] B. Möller.
Securing elliptic curve point multiplication against side-channel attacks.
In *Information Security*, volume 2200 of *LNCS*, pages 324–334. Springer, 2001.

[NMSK01] H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura.
Implementation of RSA algorithm based on RNS Montgomery multiplication.
In *Proc. 3rd Cryptographic Hardware and Embedded Systems (CHES)*, volume 2162 of *LNCS*, pages 364–376. Springer, May 2001.

[NP15] C. Negre and G. Perin.
Trade-off approaches for leak resistant modular arithmetic in RNS.
In *Proc. 20th Australasian Conference on Information Security and Privacy (ACISP, to appear)*, June 2015.

References VII

[PITM13] G. Perin, L. Imbert, L. Torres, and P. Maurine.
Electromagnetic analysis on RSA algorithm based on RNS.
In *Proc. 16th Euromicro Conference on Digital System Design (DSD)*, pages 345–352. IEEE, September 2013.

[PP95] K. C. Posch and R. Posch.
Modulo reduction in residue number systems.
IEEE Transactions on Parallel and Distributed Systems, 6(5):449–454, May 1995.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman.
A method for obtaining digital signatures and public-key cryptosystems.
Communications of the ACM, 21(2):120–126, February 1978.

[ST67] N. S. Szabo and R. I. Tanaka.
Residue arithmetic and its applications to computer technology.
McGraw-Hill, 1967.

[SV55] A. Svoboda and M. Valach.
Operátorové obvody (operator circuits in czech).
Stroje na Zpracování Informací (Information Processing Machines), 3:247–296, 1955.

References VIII

[WH66] R. W. Watson and C. W. Hastings.
Self-checked computation using residue arithmetic.
Proceedings of the IEEE, 54(12):1920–1931, 1966.

[YFCV12] G. X. Yao, J. Fan, R. C. C. Cheung, and I. Verbauwhede.
Faster pairing coprocessor architecture.
In *Proc. 5th Pairing-Based Cryptography (Pairing)*, volume 7708 of *LNCS*, pages 160–176. Springer, May 2012.

[YFCV14] G. Yao, J. Fan, R. Cheung, and I. Verbauwhede.
Novel RNS parameter selection for fast modular multiplication.
IEEE Transactions on Computers, 63(8):2099–2105, Aug 2014.

[YL73] S. S-S Yau and Y.-C. Liu.
Error correction in redundant residue number systems.
IEEE Transactions on Computers, 100(1):5–11, 1973.