Contribution to the Design of RNS Architecture

Benoît Gérard*⊛, Jean-Gabriel Kammerer*†
Nabil Merkiche*‡

*DGA/MI, Rennes, France
⊛IRISA, Rennes, France
†IRMAR, Université de Rennes 1, France
‡Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, France

June 23rd, 2015

Motivation

Context

Elliptic Curve Cryptography or RSA using RNS Architecture

Goal

- Construct a basis producing the largest product of moduli
- Improving frequency of RNS Cox-Rower architecture by removing the final subtraction

Motivation

Context

Elliptic Curve Cryptography or RSA using RNS Architecture

Goal

- Construct a basis producing the largest product of moduli
- Improving frequency of RNS Cox-Rower architecture by removing the final subtraction

Applications

Bank transaction, PayTV...

Motivation

Context

Elliptic Curve Cryptography or RSA using RNS Architecture

Goal

- Construct a basis producing the largest product of moduli
- Improving frequency of RNS Cox-Rower architecture by removing the final subtraction

Applications

Bank transaction, PayTV... Telecommunications !

Motivation

IPsec

- Non-authenticated Key Exchange (ECDH) : 2 scalar multiplications

Motivation

IPsec

- Non-authenticated Key Exchange (ECDH) : 2 scalar multiplications
- Authentication (ECDSA) : 3 scalar multiplications

Motivation

IPsec

- Non-authenticated Key Exchange (ECDH) : 2 scalar multiplications
- Authentication (ECDSA) : 3 scalar multiplications
- VPN (ECDH) : 4 scalar multiplications

Motivation

IPsec

- Non-authenticated Key Exchange (ECDH) : 2 scalar multiplications
- Authentication (ECDSA) : 3 scalar multiplications
- VPN (ECDH) : 4 scalar multiplications

Cryptoequipment with 10000 VPN $\Leftrightarrow$ 90000 scalar multiplications

Motivation

IPsec

- Non-authenticated Key Exchange (ECDH) : 2 scalar multiplications
- Authentication (ECDSA) : 3 scalar multiplications
- VPN (ECDH) : 4 scalar multiplications

Cryptoequipment with 10000 VPN $\Leftrightarrow$ 90000 scalar multiplications
20 ms for scalar multiplication $\Leftrightarrow$ 30 minutes to boot all the VPNs...

Outline

- RNS & Cox-Rower Architecture
- Self-correcting k
- Moduli selection
- Implementation & Results
- Conclusion

RNS : Motivation

Context

Modular arithmetic computations over large numbers

RNS : Motivation

Context

Modular arithmetic computations over large numbers

Classical solution

Multiple-precision algorithms (Handbook of Applied Crypto, GMP, ...)

RNS : Motivation

Context

Modular arithmetic computations over large numbers

Classical solution

Multiple-precision algorithms (Handbook of Applied Crypto, GMP, ...)

multiple-precision → carry propagations

For hardware, this means :

- bad critical path,
- no straightforward parallelism

RNS : formally

Let $M = \prod_{i=1}^{n} m_i$ and $x_i = X \mod m_i$.

$$\begin{aligned} \varphi : \mathbb{Z}/M\mathbb{Z} &\rightarrow \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_n\mathbb{Z} \\ X &\mapsto (x_1, x_2, \ldots, x_n) \end{aligned}$$

Chinese Remainder Theorem

$$\forall 1 \leqslant i < j \leqslant n, \gcd(m_i, m_j) = 1 \Leftrightarrow \varphi \text{ is an isomorphism}$$

RNS : formally

Let $M = \prod_{i=1}^{n} m_i$ and $x_i = X \mod m_i$.

$$\begin{aligned} \varphi : \mathbb{Z}/M\mathbb{Z} &\rightarrow \mathbb{Z}/m_1\mathbb{Z} \times \cdots \times \mathbb{Z}/m_n\mathbb{Z} \\ X &\mapsto (x_1, x_2, \ldots, x_n) = \{X\}_{\mathcal{B}} \end{aligned}$$

Chinese Remainder Theorem

$$\forall 1 \leqslant i < j \leqslant n, \gcd(m_i, m_j) = 1 \Leftrightarrow \varphi \text{ is an isomorphism}$$

Reconstruction

$$X = \sum_i (x_i M_i^{-1} \mod m_i) M_i \mod M,$$

where $M_i = M/m_i$.

RNS : numbers modulo M using base $\mathcal{B} = \{m_1, m_2, \ldots, m_n\}$.

Montgomery & RNS : Base Extension

Montgomery Reduction

$$X \mod p \equiv \frac{X+(X\cdot(-p^{-1}) \mod M)p}{M}$$

1. $Q_M = X \cdot (-p^{-1}) \mod M$
2. $Q_{M'} = BE(Q_M, \mathcal{B}, \mathcal{B}')$
3. $S_{M'} = (X_{M'} + Q_{M'} \cdot p) \cdot M^{-1}$
4. $S_M = BE(S_{M'}, \mathcal{B}', \mathcal{B})$

Montgomery & RNS : Base Extension

Montgomery Reduction

$$X \mod p \equiv \frac{X+(X\cdot(-p^{-1}) \mod M)p}{M}$$

1. $Q_M = X \cdot (-p^{-1}) \mod M$
2. $Q_{M'} = BE(Q_M, \mathcal{B}, \mathcal{B}')$
3. $S_{M'} = (X_{M'} + Q_{M'} \cdot p) \cdot M^{-1}$
4. $S_M = BE(S_{M'}, \mathcal{B}', \mathcal{B})$

Base Extension

$\{x'_1, \ldots, x'_n\} = BE(X, \mathcal{B}, \mathcal{B}')$

$$X = \sum_i \xi(x_i) M_i \mod M,$$

where $\xi(x_i) = x_i M_i^{-1} \mod m_i$.

Montgomery & RNS : Base Extension

Montgomery Reduction

$$X \mod p \equiv \frac{X+(X\cdot(-p^{-1}) \mod M)p}{M}$$

1. $Q_M = X \cdot (-p^{-1}) \mod M$
2. $Q_{M'} = BE(Q_M, \mathcal{B}, \mathcal{B}')$
3. $S_{M'} = (X_{M'} + Q_{M'} \cdot p) \cdot M^{-1}$
4. $S_M = BE(S_{M'}, \mathcal{B}', \mathcal{B})$

Base Extension

$\{x'_1, \ldots, x'_n\} = BE(X, \mathcal{B}, \mathcal{B}')$

$$X = \sum_i \xi(x_i) M_i \mod M,$$

where $\xi(x_i) = x_i M_i^{-1} \mod m_i$.

$$x'_j = \sum_i \xi(x_i) M_i \mod m'_j$$

Montgomery & RNS : Base Extension

Montgomery Reduction

$$X \mod p \equiv \frac{X+(X\cdot(-p^{-1}) \mod M)p}{M}$$

1. $Q_M = X \cdot (-p^{-1}) \mod M$
2. $Q_{M'} = BE(Q_M, \mathcal{B}, \mathcal{B}')$
3. $S_{M'} = (X_{M'} + Q_{M'} \cdot p) \cdot M^{-1}$
4. $S_M = BE(S_{M'}, \mathcal{B}', \mathcal{B})$

Base Extension

$\{x'_1, \ldots, x'_n\} = BE(X, \mathcal{B}, \mathcal{B}')$

$$X = \sum_i \xi(x_i) M_i \mod M,$$

where $\xi(x_i) = x_i M_i^{-1} \mod m_i$.

$$x'_j = \sum_i \xi(x_i) M_i - k(\xi(x_1), \ldots, \xi(x_n)) M \mod m'_j$$

Why using RNS ?

Advantages

- no carry propagation,
- enable parallelism,
- fast computation ($O(n)$ complexity in RNS vs $O(n^{\log_2(3)})$ in multiprecision for multiplications when using Karatsuba).

Why using RNS ?

Advantages

- no carry propagation,
- enable parallelism,
- fast computation ($O(n)$ complexity in RNS vs $O(n^{\log_2(3)})$ in multiprecision for multiplications when using Karatsuba).

Drawbacks

None

Why using RNS ?

Advantages

- no carry propagation,
- enable parallelism,
- fast computation ($O(n)$ complexity in RNS vs $O(n^{\log_2(3)})$ in multiprecision for multiplications when using Karatsuba).

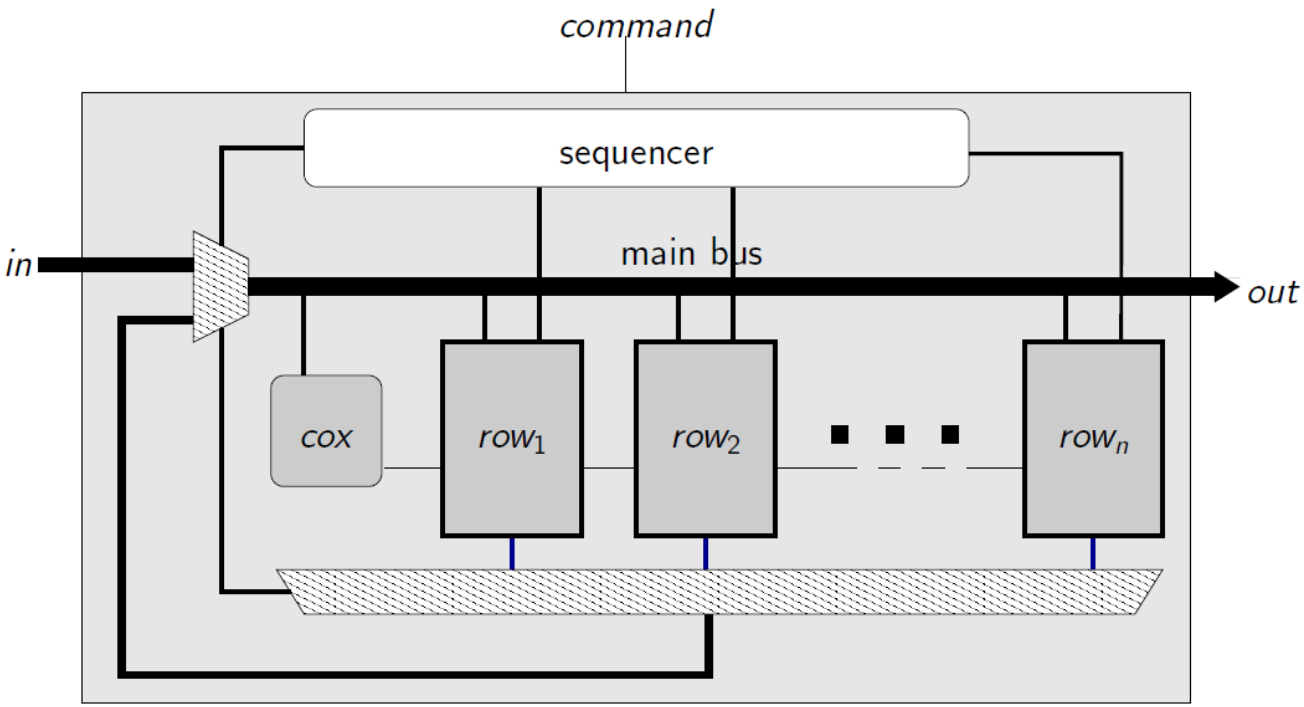
Drawbacks

- reduction over prime p is complex ($O(n^2)$ complexity in RNS),
- comparison is difficult,
- division modulo p is only possible for co-primes with M.

Cox-Rower Architecture

Features

- SIMD,
- Generic (ECC independent),
- Pipelinable,
- Suitable for lazy reduction ($\sum_{i=1}^{n} A_i B_i$).



Self-correcting k

Inner reduction without final subtraction

$\{x'_1, \ldots, x'_n\} = BE(X, \mathcal{B}, \mathcal{B}')$

$$X = \sum_i \xi_i M_i \mod M,$$

where $\xi_i \equiv x_i M_i^{-1} \mod m_i < 2m_i$.

Self-correcting k

Inner reduction without final subtraction

$\{x'_1, \ldots, x'_n\} = BE(X, \mathcal{B}, \mathcal{B}')$

$$X = \sum_i \xi_i M_i \mod M,$$

where $\xi_i \equiv x_i M_i^{-1} \mod m_i < 2m_i$.

Proof (works also for Cox-Rower) :

$$\text{Let } \xi'_i = \xi_i + \beta_i m_i \text{ with } \beta_i = \{0,1\} \text{ and } k' = \left\lfloor \sum_{i=1}^{n} \frac{\xi'_i}{m_i} \right\rfloor = k + \sum_{i=1}^{n} \beta_i$$

$$\text{Then } X' = \sum_{i=1}^{n} \xi'_i M_i - k'M = \sum_{i=1}^{n} (\xi_i M_i + \beta_i M) - kM - \sum_{i=1}^{n} \beta_i M$$

Self-correcting k

Cox-Rower impact

Condition on moduli selection was : $2^r - \alpha 2^r \leqslant m_i \leqslant 2^r$,
New condition on moduli selection : $2^r - \alpha 2^{r-1} \leqslant m_i \leqslant 2^r$.

- No need to test for final subtract with m_i
- No need to modify the truncation ($\hat{k}$ evaluation) : error induced is cancelled

Moduli selection

Context

Cox-Rower RNS needs a set of coprime moduli in a certain range given a radix size

Moduli selection

Context

Cox-Rower RNS needs a set of coprime moduli in a certain range given a radix size

Problems

How to find a maximal cardinality basis in given range ?

Moduli selection

Context

Cox-Rower RNS needs a set of coprime moduli in a certain range given a radix size

Problems

How to find a maximal cardinality basis in given range ?

First come, first selected strategy

- Add 2^r, $2^r - 1$ since coprime, $2^r - 3$ if coprime, and so on...
- Can be very bad : $2^{16} - 1 = 65535 = 3 \times 5 \times 17 \times 257$

Goal : do not consume small prime factors too fast.

Moduli selection

Proposed strategy

Examine factorization of all numbers in the range :

- Doable in practice : range has 2^{16} for $r = 32$ bits
- Consider the small prime factors
- Can add numbers with exactly one directly :
  - Consumes exactly one small prime
  - Big primes appear only once

Moduli selection

Proposed strategy

Examine factorization of all numbers in the range :

- Doable in practice : range has 2^{16} for $r = 32$ bits
- Consider the small prime factors
- Can add numbers with exactly one directly :
  - Consumes exactly one small prime
  - Big primes appear only once

Prime factors size

Prime factor size in the range $[\![2^r - \mu_{max}, 2^r]\!]$:

- Any prime greater than μ_{max} appear at most once as a factor in the range
- Especially true when $\mu_{max} = \sqrt{2^r}$ (each number has at most one prime factor greater than its square root)

Moduli selection

Proposed strategy : algorithms

Consume the least possible small primes :

- First add prime number in the range
- Add compatible numbers with exactly 1 small prime factor, greatest first :
  - There always exists a maximal basis containing these numbers
  - Remove non-coprime others candidates
  - Increase the big prime bound accordingly (as for 2^r where it is increased from μ_{max} to $2\mu_{max}$
- Compute a basis from the remaining numbers (using MAX-Clique solvers for example)

Moduli selection

In practice

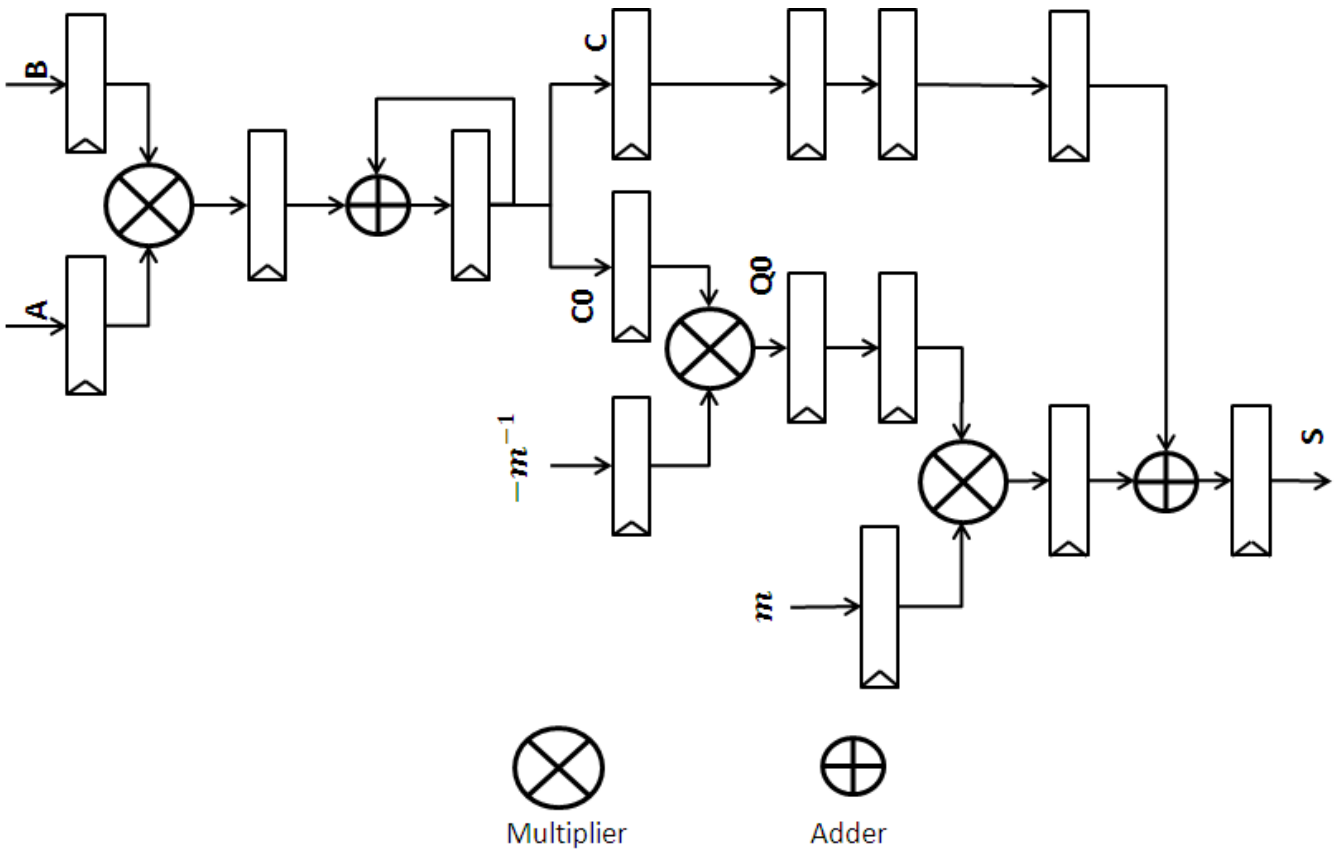
$r = 16$ bits, $\mu_{max} = 2^{r/2}$:

- First come first selected strategy : 43 elements, curve order security max about 165.5 bits,
- Proposed strategy : 48 elements, curve security max 189.5 bits,
- 24 bits security improvement, very short of attaining 192-bits security level

Implementation

Cox-Rower ALU

- No final reduction
- Adding pipeline and lazy reduction to improve frequency
- For Xilinx FPGA : fits entirely in DSP blocks



Results

Frequency limitation

In the sequencer block which decodes and translates instructions

Curve	n	Cycles	Slices (DSP/BRAM)	Fmax	Latency
160	11	88536	1168 (45/13)	400	0,221 ms
192	13	117732	1316 (53/15)	400	0,297 ms
224	15	150795	1542 (61/17)	400	0,377 ms
256	17	187828	1658 (69/19)	400	0,472 ms
384	25	373946	2630 (101/27)	400	0,935 ms
512	33	622053	3405 (133/33)	400	1,555 ms
521	33	632588	3435 (133/33)	400	1,581 ms

TABLE: P&R performances (Xilinx Kintex-7)

Results

Attainable security

Moduli range is large enough with inner Montgomery without final reduction with 16-bits multiplier

	With reduction		Without reduction
Mult. bitsize	Kawamura *et al.*	Inner Montgomery	Inner Montgomery
15	266	536	375
16	380	782	536
17	506	1118	782

TABLE: Attainable security for ALUs

Conclusion & Future works

Conclusion

- Algorithm to find the largest product of coprime moduli given a range
- Critical path moved from datapath to sequencer

Conclusion & Future works

Conclusion

- Algorithm to find the largest product of coprime moduli given a range
- Critical path moved from datapath to sequencer

Future works

- Architecture optimization for pairing computation
- Scalability to GPGPU

Conclusion & Future works

Conclusion

- Algorithm to find the largest product of coprime moduli given a range
- Critical path moved from datapath to sequencer

Future works

- Architecture optimization for pairing computation
- Scalability to GPGPU

Finally

Now, our cryptoequipment boots in 3 minutes !

Thank You !

Questions ?

benoit.gerard@irisa.fr
jean-gabriel.kammerer@m4x.org
nabil.merkiche@lip6.fr