Modular Polynomial Multiplication Using RSA/ECC coprocessor

NSS-SocialSec 2023

Aurélien Greuet, Simon Montoya, Clémence Vermeersch

IDEMIA - Crypto & Security Labs

August 16, 2023

Context

1

Post-Quantum Crypto (PQC)

Solution to resist quantum computers

- New public key crypto $\neq$ RSA, Elliptic Curves Cryptography (ECC)
- Can be run on standard devices

Post-Quantum Crypto deployment: When ?

As soon as possible!!

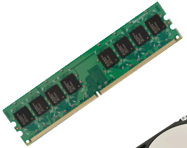
- Crypto replacement is a very long process
- Long-time confidentiality → protection against future attacks

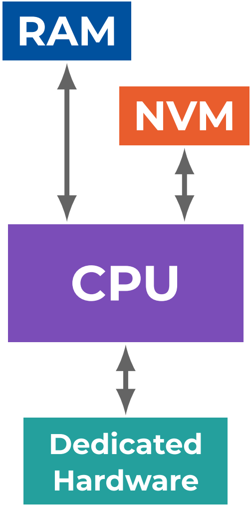
Potential issues

- Slower / more memory consuming → deployment in constrained environments?
- Unmature schemes compared to RSA / ECC → new attacks?
- Standardization still in progress → specification evolutions?

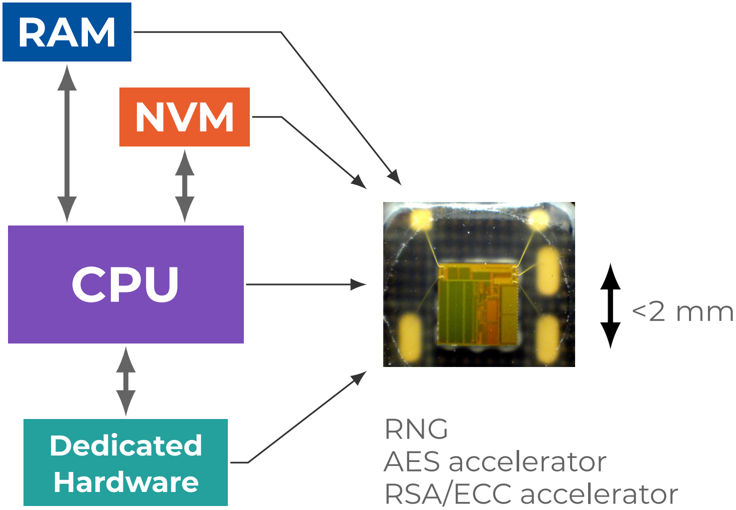
→ **Implementations on smartcards?**

Smartcard Architecture





Smartcard Architecture



Smartcard Specificities

Low Computing Capacity

	400\$ $\simeq$ 300£ PC	High-end Smartcard	
CPU	64-bit, 4 cores @4 GHz	32-bit, 1 core @100 MHz	$\longrightarrow$ $>$ 40$\times$ slower
RAM	8 GB	48 kB	$\longrightarrow$ 170 000$\times$ less

Performance Constraints – Examples

- Contactless banking transaction < 300 ms
- Key Generation performed in factory < 3-4 seconds

Solution – Dedicated Hardware

Example: RSA/ECC coprocessor

$\longrightarrow$ Instructions on large integers / modular integers (256 to 4096 bits)

$\longrightarrow$ Addition, subtraction, multiplication, shift

$\longrightarrow$ At least 20$\times$ faster than CPU for $\geq$ 256-bit numbers

Dedicated Hardware for Post-Quantum Crypto?

Not yet $\longrightarrow$ too soon because of potential evolutions of PQC algorithms

Problematic – State of the Art

2

Problematic

Problem

Multiply polynomials

- of "large" degree, e.g. 256 or 512
- with "small" modular coeffs in $\mathbb{Z}_q$, $q =$ fixed prime number or 2^k, < 32-bit

→ core operation in lattice-based Post-Quantum schemes

Standard Methods

- Generic: Karatsuba / Toom-Cook
- With assumptions on prime modulus: Number Theoretical Transform (NTT)

→ fast algorithms but rely on multiplication between coefficients

› Problematic

Find faster generic way to multiply polynomials with modular coeffs when

- no / slow small multiplication
- fast coprocessor for large integer arithmetic

State of the Art

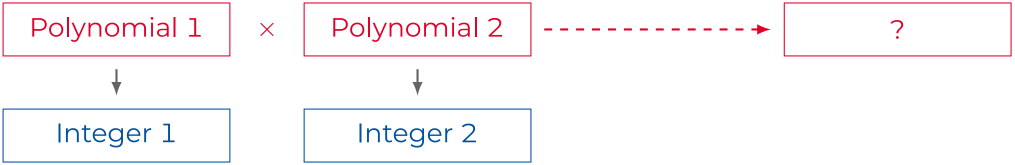
Kronecker Substitution

→ polynomial multiplication as integer multiplication

State of the Art

Kronecker Substitution

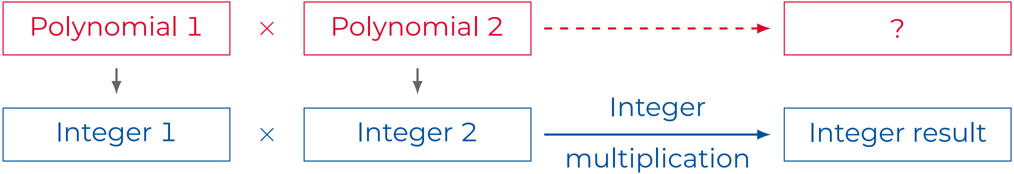
→ polynomial multiplication as integer multiplication



State of the Art

Kronecker Substitution

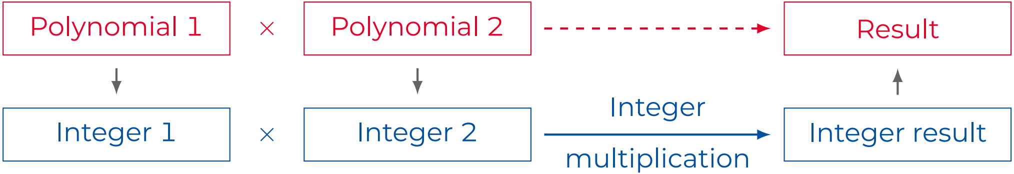
→ polynomial multiplication as integer multiplication



State of the Art

Kronecker Substitution

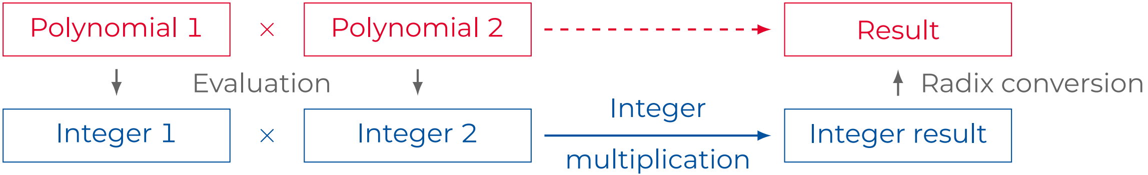
→ polynomial multiplication as integer multiplication



State of the Art

Kronecker Substitution

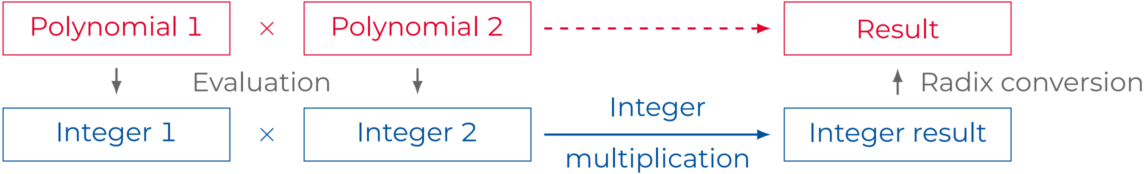
→ polynomial multiplication as integer multiplication



State of the Art

Kronecker Substitution

→ polynomial multiplication as integer multiplication



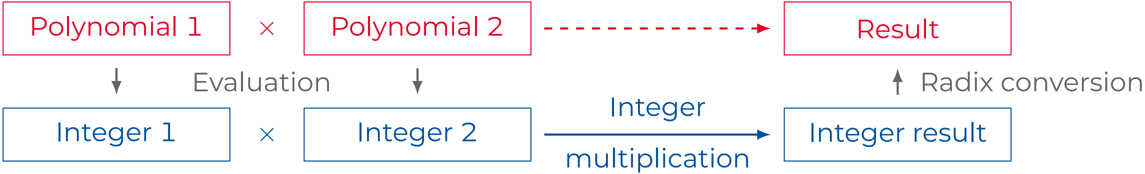
Example

$P_1(x) = 2x^2 + 1x + 3$ × $P_2(x) = 1x^2 + 3x + 1$

State of the Art

Kronecker Substitution

→ polynomial multiplication as integer multiplication



Example

$P_1(x) = 2x^2 + 1x + 3$ × $P_2(x) = 1x^2 + 3x + 1$

↓

$P_1(100) = 020103$ $\quad$ $P_2(100) = 010301$

State of the Art

Kronecker Substitution

→ polynomial multiplication as integer multiplication

Polynomial 1 × Polynomial 2 - - - → Result

↓ Evaluation ↓ ↑ Radix conversion

Integer 1 × Integer 2 —Integer multiplication→ Integer result

Example

$P_1(x) = 2x^2 + 1x + 3$ × $P_2(x) = 1x^2 + 3x + 1$

↓ ↓

$P_1(100) = 020103$ × $P_2(100) = 010301$ —Integer multiplication→ 0207081003

State of the Art

Kronecker Substitution

→ polynomial multiplication as integer multiplication

Polynomial 1 × Polynomial 2 → Result

↓ Evaluation ↓ ↑ Radix conversion

Integer 1 × Integer 2 → (Integer multiplication) → Integer result

Example

$P_1(x) = 2x^2 + 1x + 3$ × $P_2(x) = 1x^2 + 3x + 1$ → $2x^4 + 7x^3 + 8x^2 + 10x + 3$

↓ ↓ ↑

$P_1(100) = 020103$ × $P_2(100) = 010301$ → (Integer multiplication) → 0207081003

State of the Art

Kronecker Substitution

→ polynomial multiplication as integer multiplication

Polynomial 1 × Polynomial 2 - - - → Result

↓ Evaluation ↓ ↑ Radix conversion

Integer 1 × Integer 2 —Integer multiplication→ Integer result

Example

$P_1(x) = 2x^2 + 1x + 3$ × $P_2(x) = 1x^2 + 3x + 1$ - - - → $2x^4 + 7x^3 + 8x^2 + 10x + 3$

↓ ↓ ↑

$P_1(100) = 020103$ × $P_2(100) = 010301$ —Integer multiplication→ 0207081003

It works because

- Evaluation point is large enough → no coeff overlapping in the result
- Non-negative coeffs → evaluation = "concatenation" / radix conv = "splitting"

State of the Art

Modular Polynomial Multiplication with Coprocessor

Input: polynomial with coefficients mod q in $\{-q/2-1,\ldots,q/2\}$

1. Add q to negative coefficients $\longrightarrow$ ensure all coeffs ≥ 0
2. Evaluation $\longrightarrow$ concatenation of coeffs
3. Integer multiplication $\longrightarrow$ done with coprocessor
4. Radix conversion $\longrightarrow$ splitting integer to coeffs
5. Modular reduction of each coefficient $\longrightarrow$ one coeff at a time

[Albrecht et al., *Implementing RLWE-based schemes using an RSA Co-Processor*, TCHES 2019]
[Bos et al., *Post-Quantum Cryptography with Contemporary Co-Processors*, USENIX 2021]
[Wang et al., *Saber on ESP32*, ACNS 2020]

State of the Art

Modular Polynomial Multiplication with Coprocessor

Input: polynomial with coefficients mod q in $\{-q/2-1, \ldots, q/2\}$

1. Add q to negative coefficients $\longrightarrow$ ensure all coeffs ≥ 0
2. Evaluation $\longrightarrow$ concatenation of coeffs
3. Integer multiplication $\longrightarrow$ done with coprocessor
4. Radix conversion $\longrightarrow$ splitting integer to coeffs
5. Modular reduction of each coefficient $\longrightarrow$ one coeff at a time

[Albrecht et al., *Implementing RLWE-based schemes using an RSA Co-Processor*, TCHES 2019]
[Bos et al., *Post-Quantum Cryptography with Contemporary Co-Processors*, USENIX 2021]
[Wang et al., *Saber on ESP32*, ACNS 2020]

Limitations

- Make input coefficients ≥ 0
- Modular reduction of each coefficients, one by one

Contributions
3

Contributions

State of the Art Limitations

- Make input coefficients ≥ 0
- Modular reduction of each coefficients

Contributions

- Slight modification of coprocessor → two new simple instructions
- With coprocessor modification, new Modular Polynomial Multiplication:
  1. Sign-independent evaluation
  2. Integer multiplication
  3. Simultaneous modular reduction done with coprocessor
  4. Radix conversion
- Implementation for several PQC algos
- Comparison with standard methods

Contributions

State of the Art Limitations

- Make input coefficients ≥ 0
- Modular reduction of each coefficients

Contributions

- Slight modification of coprocessor $\longrightarrow$ two new simple instructions
- With coprocessor modification, new Modular Polynomial Multiplication:
  1. Sign-independent evaluation
  2. Integer multiplication
  3. Simultaneous modular reduction done with coprocessor
  4. Radix conversion
- Implementation for several PQC algos
- Comparison with standard methods

Contribution – Simultaneous Modular Reduction

Standard Barrett Modular Reduction

Goal Fixed modulus q, compute r s.t. $a = \lfloor a/q \rfloor \cdot q + r$ without division

Contribution – Simultaneous Modular Reduction

Standard Barrett Modular Reduction

Goal Fixed modulus q, compute r s.t. $a = \lfloor a/q \rfloor \cdot q + r$ without division

Idea Approximate $\lfloor a/q \rfloor$ with $\left\lfloor \frac{a}{2^\ell} \cdot \left\lfloor \frac{2^\ell}{q} \right\rfloor \right\rfloor$

→ $m = \lfloor 2^\ell / q \rfloor$ can be precomputed

→ $\left\lfloor \frac{a}{2^\ell} \cdot m \right\rfloor = (a \cdot m) >> \ell$ → (fast) shift instead of (slow) division

Contribution – Simultaneous Modular Reduction

Standard Barrett Modular Reduction

Goal Fixed modulus q, compute r s.t. $a = \lfloor a/q \rfloor \cdot q + r$ without division

Idea Approximate $\lfloor a/q \rfloor$ with $\left\lfloor \frac{a}{2^\ell} \cdot \left\lfloor \frac{2^\ell}{q} \right\rfloor \right\rfloor$

→ $m = \lfloor 2^\ell / q \rfloor$ can be precomputed

→ $\left\lfloor \frac{a}{2^\ell} \cdot m \right\rfloor = (a \cdot m) >> \ell$ → (fast) shift instead of (slow) division

Algo
- Precompute $m = \lfloor 2^\ell / q \rfloor$
- $t \leftarrow (a \cdot m) >> \ell$
- $r \leftarrow a - t \cdot q$ (→ $0 \leq r < 2q$)
- if $r \geq q$ then $r \leftarrow r - q$

Contribution – Simultaneous Modular Reduction

Standard Barrett Modular Reduction

Goal Fixed modulus q, compute r s.t. $a = \lfloor a/q \rfloor \cdot q + r$ without division

Idea Approximate $\lfloor a/q \rfloor$ with $\left\lfloor \frac{a}{2^\ell} \cdot \left\lfloor \frac{2^\ell}{q} \right\rfloor \right\rfloor$

→ $m = \lfloor 2^\ell / q \rfloor$ can be precomputed

→ $\left\lfloor \frac{a}{2^\ell} \cdot m \right\rfloor = (a \cdot m) >> \ell$ → (fast) shift instead of (slow) division

Algo
- Precompute $m = \lfloor 2^\ell / q \rfloor$
- $t \leftarrow (a \cdot m) >> \ell$
- $r \leftarrow a - t \cdot q$ $\quad$ ($\rightarrow 0 \leq r < 2q$)
- if $r \geq q$ then $r \leftarrow r - q$

Our Idea – Barrett generalization

Contribution – Simultaneous Modular Reduction

Example

Modulus $q = 7, \ell = 4, m = \lfloor 2^{\ell}/q \rfloor = 2$

0 4	0 1	1 B	0 D	1 3	0 7	0 8

Standard Barrett

- $t \leftarrow a \cdot m$
- $t \leftarrow t >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Contribution – Simultaneous Modular Reduction

Example

Modulus $q = 7, \ell = 4, m = \lfloor 2^\ell / q \rfloor = 2$

04	01	1B	0D	13	07	08
08	02	36	1A	26	0E	10

$a \cdot m$

Standard Barrett

- $t \leftarrow a \cdot m$
- $t \leftarrow t >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Contribution – Simultaneous Modular Reduction

Example

Modulus $q = 7$, $\ell = 4$, $m = \lfloor 2^\ell / q \rfloor = 2$

0 4	0 1	1 B	0 D	1 3	0 7	0 8	
0 8	0 2	3 6	1 A	2 6	0 E	1 0	$a \cdot m$
0 0	8 0	2 3	6 1	A 2	6 0	E 1	$(a \cdot m) >> \ell$

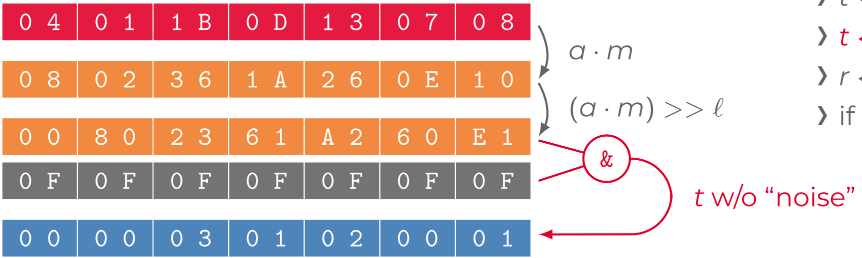
Standard Barrett

- $t \leftarrow a \cdot m$
- $t \leftarrow t >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Contribution – Simultaneous Modular Reduction

Example

Modulus $q = 7$, $\ell = 4$, $m = \lfloor 2^\ell / q \rfloor = 2$



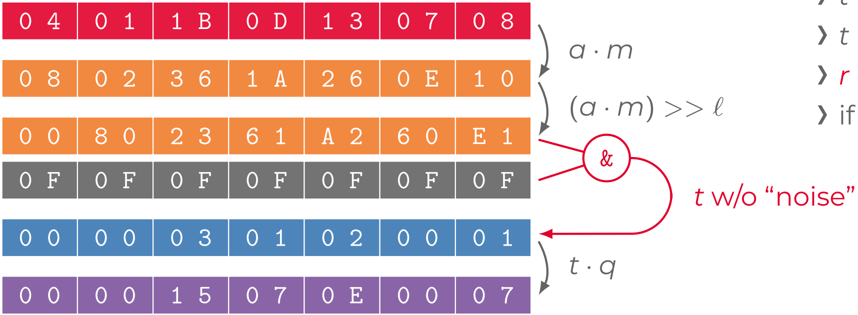
Standard Barrett

- $t \leftarrow a \cdot m$
- $t \leftarrow t >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Contribution – Simultaneous Modular Reduction

Example

Modulus $q = 7, \ell = 4, m = \lfloor 2^\ell / q \rfloor = 2$



Standard Barrett

- $t \leftarrow a \cdot m$
- $t \leftarrow t >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Contribution – Simultaneous Modular Reduction

Example

Modulus $q = 7, \ell = 4, m = \lfloor 2^\ell / q \rfloor = 2$

0 4	0 1	1 B	0 D	1 3	0 7	0 8
0 8	0 2	3 6	1 A	2 6	0 E	1 0
0 0	8 0	2 3	6 1	A 2	6 0	E 1
0 F	0 F	0 F	0 F	0 F	0 F	0 F
0 0	0 0	0 3	0 1	0 2	0 0	0 1
0 0	0 0	1 5	0 7	0 E	0 0	0 7
0 4	0 1	0 6	0 6	0 5	0 7	0 1

$a \cdot m$

$(a \cdot m) >> \ell$

&

t w/o "noise"

$t \cdot q$

$a - t \cdot q$

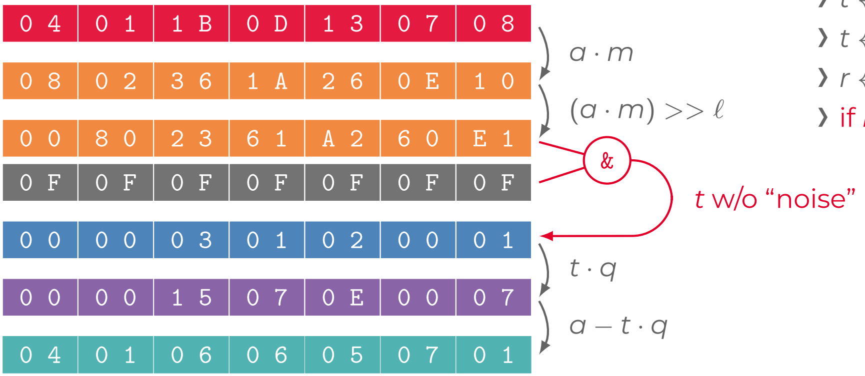
Standard Barrett

- $t \leftarrow a \cdot m$
- $t \leftarrow t >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Contribution – Simultaneous Modular Reduction

Example

Modulus $q = 7, \ell = 4, m = \lfloor 2^{\ell}/q \rfloor = 2$



Standard Barrett

- $t \leftarrow a \cdot m$
- $t \leftarrow t >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Last step: with same techniques, build and subtract

00	00	00	00	00	07	00

Practical Results

Setup

- 32-bit smartcard component, standard instructions
- No CPU multiplication / no CPU division
- Coprocessor: (modular) add, sub, shift, mult on $\leq$ 4096-bit operands
- Additional copro instructions: bitwise and, duplication $x \mapsto x||\cdots||x$

› Performance of polynomial multiplication in PQC schemes

Versus NTT with copro for coeffs mult

Kyber 1.3 to 2.9$\times$ faster

Dilithium 1.25 to 2.3$\times$ faster

Versus Toom-Cook / Karatsuba (power of 2)

Saber 20 to 30$\times$ faster

NTRU 10$\times$ faster

Questions?

IDEMIA

Join us on

www.idemia.com

Practical Results

More detailed setup

- 32-bit smartcard component, standard instructions
  → add, sub, shift, and, xor in 1 cycle, data transfert in 2 cycles
- No CPU multiplication / no CPU division
- Coprocessor: (modular) add, sub, shift, mult on $\leq$ 4096-bit operands
- Additional copro instructions: bitwise and, duplication $x \mapsto x||\cdots||x$
  → (modular) multiplication in $15 + \text{wlen}(op_1) \cdot \text{wlen}(op_2)/4$ cycles
  → other operations in $15 + \max(\text{wlen}(ops))/2$

Performance of polynomial multiplication in PQC schemes

Versus NTT with copro for coeffs mult

Kyber	1.3 to 2.9× faster
Dilithium	1.25 to 2.3× faster

Versus Toom-Cook / Karatsuba (power of 2)

Saber	20 to 30× faster
NTRU	10× faster

Last Step of Simultaneous Modular Reduction

Example – last step = conditional subtraction

0 4	0 1	1 B	0 D	1 3	0 7	0 8

0 4	0 1	0 6	0 6	0 5	0 7	0 1

Standard Barrett

- $t \leftarrow (a \cdot m) >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Last Step of Simultaneous Modular Reduction

Example – last step = conditional subtraction

0 4	0 1	1 B	0 D	1 3	0 7	0 8

0 4	0 1	0 6	0 6	0 5	0 7	0 1

Standard Barrett

- $t \leftarrow (a \cdot m) >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Lemma

Let c and k s.t. $q = 2^k - c$, let $0 \leq a < 2q$. Then

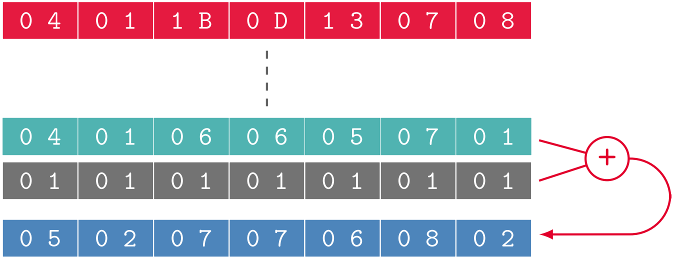
$$a + c \geq 2^k \iff a \geq q$$

→ $(k+1)$-th bit of $a+c$ is 1 iff a needs subtraction

→ here $k = 3$

Last Step of Simultaneous Modular Reduction

Example – last step = conditional subtraction



Standard Barrett

- $t \leftarrow (a \cdot m) >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Lemma

Let c and k s.t. $q = 2^k - c$, let $0 \leq a < 2q$. Then

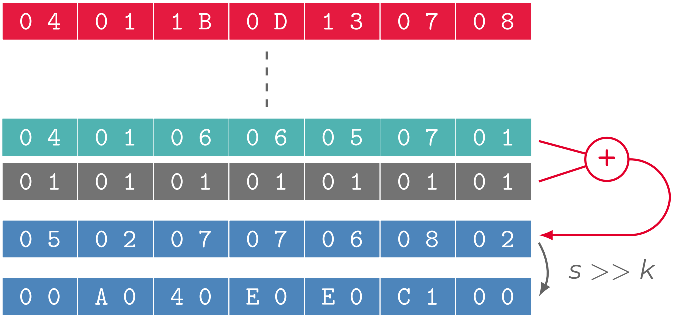
$$a + c \geq 2^k \iff a \geq q$$

→ $(k+1)$-th bit of $a+c$ is 1 iff a needs subtraction

→ here $k = 3$

Last Step of Simultaneous Modular Reduction

Example – last step = conditional subtraction



Standard Barrett

- $t \leftarrow (a \cdot m) >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Lemma

Let c and k s.t. $q = 2^k - c$, let $0 \leq a < 2q$. Then

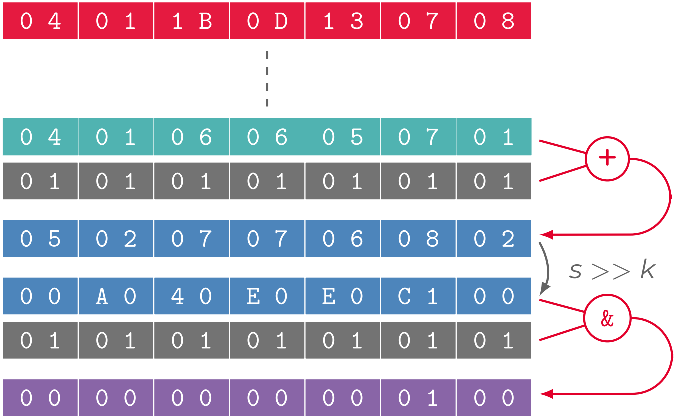
$$a + c \geq 2^k \iff a \geq q$$

→ $(k+1)$-th bit of $a+c$ is 1 iff a needs subtraction

→ here $k = 3$

Last Step of Simultaneous Modular Reduction

Example – last step = conditional subtraction



Standard Barrett

- $t \leftarrow (a \cdot m) >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Lemma

Let c and k s.t. $q = 2^k - c$, let $0 \leq a < 2q$. Then

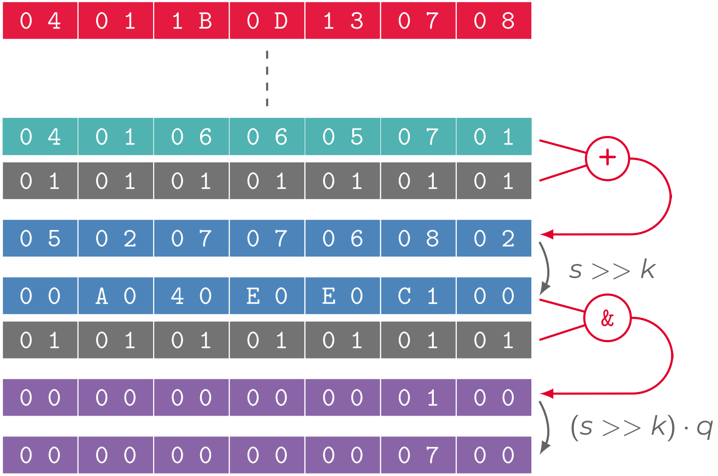
$$a + c \geq 2^k \iff a \geq q$$

→ $(k{+}1)$-th bit of $a{+}c$ is 1 iff a needs subtraction

→ here $k = 3$

Last Step of Simultaneous Modular Reduction

Example – last step = conditional subtraction



Standard Barrett

- $t \leftarrow (a \cdot m) >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Lemma

Let c and k s.t. $q = 2^k - c$, let $0 \leq a < 2q$. Then

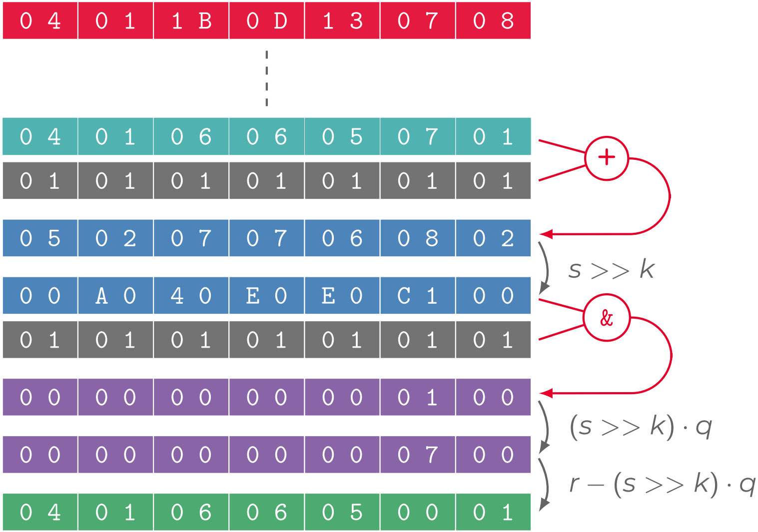
$$a + c \geq 2^k \iff a \geq q$$

→ $(k+1)$-th bit of $a+c$ is 1 iff a needs subtraction

→ here $k = 3$

Last Step of Simultaneous Modular Reduction

Example – last step = conditional subtraction



Standard Barrett

- $t \leftarrow (a \cdot m) >> \ell$
- $r \leftarrow a - t \cdot q$
- if $r \geq q$ then $r \leftarrow r - q$

Lemma

Let c and k s.t. $q = 2^k - c$, let $0 \leq a < 2q$. Then

$$a + c \geq 2^k \iff a \geq q$$

→ $(k+1)$-th bit of $a+c$ is 1 iff a needs subtraction

→ here $k = 3$