High-Speed Implementations of RSA & Elliptic Curve Cryptosystems

Çetin Kaya Koç

Oregon State University

Contents

- RSA arithmetic
- Elliptic curve arithmetic
- Exponentiation heuristics
- Canonical recoding exponentiation
- Montgomery multiplication

RSA Arithmetic

The RSA algorithm uses modular exponentiation for encryption

$$C = M^e \pmod{n}$$

and decryption

$$M = C^d \pmod{n}$$

The computation of $M^e \bmod n$ is performed using exponentiation algorithms (heuristics)

Modular exponentiation requires implementation of three basic modular arithmetic operations: addition, subtraction, and multiplication

The decryption can be decomposed into two half-size modular exponentiations using CRT (*Quisquater & Couvreur 82*)

Elliptic Curve Arithmetic

Elliptic curves defined over $GF(p)$ or $GF(2^k)$ are used in cryptography

The arithmetic of $GF(p)$ is the usual mod p arithmetic

The arithmetic of $GF(2^k)$ is similar to that of $GF(p)$, however, there are some differences

Elliptic curves over $GF(2^k)$ are more popular due to the space and time-efficient algorithms for doing arithmetic in $GF(2^k)$

Elliptic curve cryptosystems based on discrete logarithms seem to provide similar amount of security to that of RSA, but with relatively shorter key sizes

Elliptic Curves over $GF(p)$

Let $p > 3$ be a prime number and $a, b \in GF(p)$ be such that $4a^3 + 27b^2 \neq 0$ in $GF(p)$. An elliptic curve E over $GF(p)$ is defined by the parameters a and b as the set of solutions (x, y) where $x, y \in GF(p)$ to the equation

$$y^2 = x^3 + ax + b$$

together with an extra point O. The set of points E form a group with respect to the addition rules:

- $O + O = O$
- $(x, y) + O = (x, y)$
- $(x, y) + (x, -y) = O$

Elliptic Curves over $GF(p)$

- Addition of two points with $x_1 \neq x_2$

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

$$\begin{aligned} \lambda &= \frac{y_2 - y_1}{x_2 - x_1} \\ x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}$$

- Doubling of a point with $x_1 \neq 0$

$$(x_1, y_1) + (x_1, y_1) = (x_3, y_3)$$

$$\begin{aligned} \lambda &= \frac{3x_1^2 + a}{2y_1} \\ x_3 &= \lambda^2 - 2x_1 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned}$$

Elliptic Curves over $GF(2^k)$

A non-supersingular elliptic curve E over the field $GF(2^k)$ is defined by parameters $a, b \in GF(2^k)$ with $b \neq 0$ is the set of solutions (x, y) where $x, y \in GF(2^k)$, to the equation

$$y^2 + xy = x^3 + ax^2 + b$$

together with an extra point O. The set of points E form a group with respect to the addition rules:

- $O + O = O$
- $(x, y) + O = (x, y)$
- $(x, y) + (x, x + y) = O$

Elliptic Curves over $GF(2^k)$

- Addition of two points with $x_1 \neq x_2$

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

$$\begin{aligned} \lambda &= \frac{y_1 + y_2}{x_1 + x_2} \\ x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1 \end{aligned}$$

- Doubling of a point with $x_1 \neq 0$

$$(x_1, y_1) + (x_1, y_1) = (x_3, y_3)$$

$$\begin{aligned} \lambda &= x_1 + \frac{y_1}{x_1} \\ x_3 &= \lambda^2 + \lambda + a \\ y_3 &= x_1^2 + (\lambda + 1)x_3 \end{aligned}$$

Elliptic Curve Cryptosystems

Based on the difficulty of computing e given eP where P is a point on the curve

Example: Elliptic Curve Diffie-Hellman

Alice and Bob agree on, the elliptic curve E, the underlying field $GF(2^k)$ or $GF(p)$, and the generating point P with order n

- Alice sends $Q = aP$ to Bob
- Bob sends $R = bP$ to Alice
- Alice computes $S = a(R) = abP$
- Bob computes $S = b(Q) = abP$

Adversary knows P, and sees Q and R

Computing S seems to require elliptic logarithms (*Miller 85, Koblitz 87, Menezes 93*)

Elliptic Curve Arithmetic

Computation of eP can be performed using exponentiation algorithms

In order to compute e multiple of P we perform elliptic curve additions

An elliptic curve addition is performed by using a few **finite field** operations

Implementation of elliptic curve addition operation requires implementation of four basic finite field operations: addition, subtraction, multiplication, and inversion

Inversion is a relatively expensive operation

Projective coordinates allow us to eliminate the need for performing inversion (*Menezes 93*)

Exponentiation Heuristics

Given the integer e, the computation of M^e or eP is an exponentiation operation

The objective is to use as few multiplications (or elliptic curve additions) as possible for a given integer e

This problem is related to **addition chains**

An addition chain is a sequence of integers

$$a_0 \quad a_1 \quad a_2 \quad \cdots \quad a_r$$

starting from $a_0 = 1$ and ending with $a_r = e$ such that any a_k is the sum of two earlier integers a_i and a_j in the chain:

$$a_k = a_i + a_j \quad \text{for } 0 < i, j < k$$

Addition Chains

Example: $e = 55$

1	2	3	6	12	13	26	27	54	55
1	2	3	6	12	13	26	52	55	
1	2	4	5	10	20	40	50	55	
1	2	3	5	10	11	22	44	55	

An addition chain yields an algorithm for computing M^e or eP given the integer e

$$M^1 \; M^2 \; M^3 \; M^5 \; M^{10} \; M^{11} \; M^{22} \; M^{44} \; M^{55}$$

$$P \quad 2P \quad 3P \quad 5P \quad 10P \quad 11P \quad 22P \quad 44P \quad 55P$$

The length of the chain r gives the number of operations required to compute M^e or eP

Addition Chains

Finding the shortest addition chain is an NP-complete problem (*Downey 81*)

Let $H(e)$ be the Hamming weight of e

Upper bound: $\lfloor \log_2 e \rfloor + H(e) - 1$
(*Knuth 81*)

Lower bound: $\log_2 e + \log_2 H(e) - 2.13$
(*Schönhage 75*)

Heuristics: binary, m-ary, sliding windows

Statistical methods, such as simulated annealing, can be used to produce short addition chains for certain exponents

Binary Method

Scan the bits of e and perform squarings and multiplications to compute $C = M^e$

1. **if** $e_{k-1} = 1$ **then** $C := M$ **else** $C := 1$
2. **for** $i = k-2$ **downto** 0

2a. $\quad C := C \cdot C \pmod{n}$

2b. $\quad$ **if** $e_i = 1$ **then** $C := C \cdot M \pmod{n}$

3. **return** C

Similarly, elliptic curve doublings and additions are performed in order to compute $Q = eP$

1. **if** $e_{k-1} = 1$ **then** $Q := P$ **else** $Q := O$
2. **for** $i = k-2$ **downto** 0

2a. $\quad Q := Q + Q$

2b. $\quad$ **if** $e_i = 1$ **then** $Q := Q + P$

3. **return** Q

Example: $e = 55 = (110111)$

Step 1: $e_5 = 1 \longrightarrow C := M$

i	e_i	Step 2a (C)	Step 2b (C)
4	1	$(M)^2 = M^2$	$M^2 \cdot M = M^3$
3	0	$(M^3)^2 = M^6$	M^6
2	1	$(M^6)^2 = M^{12}$	$M^{12} \cdot M = M^{13}$
1	1	$(M^{13})^2 = M^{26}$	$M^{26} \cdot M = M^{27}$
0	1	$(M^{27})^2 = M^{54}$	$M^{54} \cdot M = M^{55}$

Step 1: $e_5 = 1 \longrightarrow Q := P$

i	e_i	Step 2a (Q)	Step 2b (Q)
4	1	$P + P = 2P$	$2P + P = 3P$
3	0	$3P + 3P = 6P$	$6P$
2	1	$6P + 6P = 12P$	$12P + P = 13P$
1	1	$13P + 13P = 26P$	$26P + P = 27P$
0	1	$27P + 27P = 54P$	$54P + P = 55P$

The m-ary Method

Scan d-bit words in e, where $2^d = m$

Example: Quaternary Method

$$e = 250 = \underline{11}\ \underline{11}\ \underline{10}\ \underline{10}$$

Pre-processing:

- $00 \rightarrow M^0 = 1$
- $01 \rightarrow M^1 = M$
- $10 \rightarrow M \cdot M = M^2$
- $11 \rightarrow M^2 \cdot M = M^3$

bits	Step 2a	Step 2b
11	M^3	M^3
11	$(M^3)^4 = M^{12}$	$M^{12} \cdot M^3 = M^{15}$
10	$(M^{15})^4 = M^{60}$	$M^{60} \cdot M^2 = M^{62}$
10	$(M^{62})^4 = M^{248}$	$M^{248} \cdot M^2 = M^{250}$

Quaternary method: $2 + 6 + 3 = 11$
Binary method: $7 + 5 = 12$

Analysis of the m-ary Method

The average number of multiplications plus squarings required by the m-ary method:

- Pre-processing multiplications: $2^d - 2$

- Squarings: $(\frac{k}{d} - 1) \cdot d = k - d$

Probability that a length d bit-section has all bits equal to zero: 2^{-d}

- Multiplications: $(1 - 2^{-d}) \cdot (\frac{k}{d} - 1)$

- There is an optimal d for every k

Addition-Subtraction Chains

An addition-subtraction chain is a sequence of integers

$$a_0 \quad a_1 \quad a_2 \quad \cdots \quad a_r$$

starting from $a_0 = \pm 1$ and ending with $a_r = e$ such that any a_k is the sum or the difference of two earlier integers a_i and a_j in the chain:

$$a_k = a_i \pm a_j \quad \text{for } 0 < i, j < k$$

Example: $e = 55$

$$\pm 1 \quad 2 \quad 4 \quad 8 \quad 7 \quad 14 \quad 28 \quad 56 \quad 55$$

An addition-subtraction chain is an algorithm for computing M^e or eP given the integer e

However, it requires negative powers of M or negative multiples of P

Recoding Technique

We obtain a sparse signed-digit representation of the exponent with digits $\{0, 1, -1\}$, e.g.,

$$
\begin{aligned}
30 = (011110) &= 2^4 + 2^3 + 2^2 + 2^1 \\
30 = (1000\bar{1}0) &= 2^5 - 2^1
\end{aligned}
$$

This method needs $M^{-1} \pmod{n}$ as input

Recoding Binary Method

Input: M, M^{-1}, e, n

Output: $C := M^e \bmod n$

0. Obtain a signed-digit recoding f of e
1. **if** $f_k = 1$ **then** $C := M$ **else** $C := 1$
2. **for** $i = k - 1$ **downto** 0

2a. $C := C \cdot C \pmod{n}$

2b. **if** $f_i = 1$ **then** $C := C \cdot M \pmod{n}$

if $f_i = \bar{1}$ **then** $C := C \cdot M^{-1} \pmod{n}$

3. **return** C

Recoding Technique Example

Example: $e = 119 = (1110111)$

Binary method: $6 + 5 = 11$ multiplications

Exponent:	01110111
Recoded exponent:	$1000\bar{1}00\bar{1}$

f_i	Step 2a (C)	Step 2b (C)
1	M	M
0	$(M)^2 = M^2$	M^2
0	$(M^2)^2 = M^4$	M^4
0	$(M^4)^2 = M^8$	M^8
$\bar{1}$	$(M^8)^2 = M^{16}$	$M^{16} \cdot M^{-1} = M^{15}$
0	$(M^{15})^2 = M^{30}$	M^{30}
0	$(M^{30})^2 = M^{60}$	M^{60}
$\bar{1}$	$(M^{60})^2 = M^{120}$	$M^{120} \cdot M^{-1} = M^{119}$

The number of multiplications: $7 + 2 = 9$

Also requires M^{-1} (which is costly)

Applications to Elliptic Curves

Addition-subtraction chains are suitable for elliptic curves since computing $-P$ is trivial

For elliptic curves over $GF(p)$:
if $P = (x, y)$, then $-P = (x, -y)$

Non-supersingular elliptic curves over $GF(2^k)$:
if $P = (x, y)$, then $-P = (x, x + y)$

Input: $P, -P, e$
Output: $Q := eP$

0. Obtain a signed-digit recoding f of e
1. **if** $f_k = 1$ **then** $Q := P$ **else** $Q := O$
2. **for** $i = k - 1$ **downto** 0

 2a. $Q := Q + Q$

 2b. **if** $f_i = 1$ **then** $Q := Q + P$

 if $f_i = \bar{1}$ **then** $Q := Q + (-P)$

3. **return** Q

Reitwiesner's Algorithm

The canonical recoding algorithm optimally encodes the exponent using the digits $\{0, 1, \bar{1}\}$ (*Reitwiesner 60*)

e_{i+1}	e_i	a_i	f_i	a_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	0
1	0	1	$\bar{1}$	1
1	1	0	$\bar{1}$	1
1	1	1	0	1

For example, $e = 3038$ is encoded as

$$\begin{aligned} e &= (0101111011110) \\ f &= (10\bar{1}0000\bar{1}000\bar{1}0) \end{aligned}$$

requiring 3 multiplications instead of 9 (in addition to the squarings)

Canonical Recoding m-ary Method

If the digits of the canonically recoded exponent are scanned d at a time, we obtain the canonical recoding m-ary method
(*Eğecioğlu & Koç 94*)

- Pre-processing multiplications:

$$\frac{1}{3}\,[2^{d+2} + (-1)^{d+1}\,] - 3$$

$\mathcal{L}$: the formal language of all words w over the alphabet $\{\overline{1}, 0, 1\}$ in which none of these patterns appears:

$$1\,1\ ,\ 1\,\overline{1}\ ,\ \overline{1}\,1\ ,\ \overline{1}\,\overline{1}$$

τ_d: the number of words of length d in $\mathcal{L}$
λ: the empty word, and $+$: disjoint union

$$\mathcal{L} = \lambda + 1 + \overline{1} + 10\,\mathcal{L} + \overline{1}0\,\mathcal{L} + 0\,\mathcal{L}$$

Analysis of Recoding m-ary Method

The generating function

$$f_{\mathcal{L}}(t) = \sum_{w \in \mathcal{L}} t^{|w|} = \sum_{d \geq 0} \tau_d \, t^d$$

satisfies

$$f_{\mathcal{L}}(t) = 1 + 2t + 2t^2 f_{\mathcal{L}}(t) + t f_{\mathcal{L}}(t) \quad ,$$

and therefore

$$f_{\mathcal{L}}(t) = \frac{1+2t}{1-t-2t^2} \;=\; \frac{4}{3} \cdot \frac{1}{1-2t} - \frac{1}{3} \cdot \frac{1}{1+t}$$

from which it follows that

$$\tau_d = \frac{1}{3} \, [2^{d+2} + (-1)^{d+1} \,]$$

The pre-processing multiplications: $\tau_d - 3$

(since M^0, M, and M^{-1} are available)

Analysis of Recoding m-ary Method

- Squarings $(\frac{k}{d} - 1) \cdot d = k - d$
- Multiplications: $(1 - P(d)) \cdot (\frac{k}{d} - 1)$

$P(d)$: the probability that a length d bit-section in a canonically recoded signed-digit vector has all bits equal to zero

We show that $P(d) = \frac{2}{3} \cdot \left(\frac{1}{2}\right)^{d-1} = \frac{1}{3 \cdot 2^{d-2}}$

State		Output	Next State	
s_i	(e_{i+1}, e_i, a_i)	(f_i, a_{i+1})	$e_{i+2} = 0$	$e_{i+2} = 1$
s_0	$(0,0,0)$	$(0,0)$	s_0	s_4
s_1	$(0,0,1)$	$(1,0)$	s_0	s_4
s_2	$(0,1,0)$	$(1,0)$	s_0	s_4
s_3	$(0,1,1)$	$(0,1)$	s_1	s_5
s_4	$(1,0,0)$	$(0,0)$	s_2	s_6
s_5	$(1,0,1)$	$(\overline{1},1)$	s_3	s_7
s_6	$(1,1,0)$	$(\overline{1},1)$	s_3	s_7
s_7	$(1,1,1)$	$(0,1)$	s_3	s_7

Generation of Recoded Digits

$\mathcal{P}_{ij}$: Probability that the successor of s_i is s_j

$$\mathcal{P} = \begin{bmatrix} 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \end{bmatrix}$$

Limiting probabilities of the states:

$$\pi = \left[\frac{1}{6}\,,\ \frac{1}{12}\,,\ \frac{1}{12}\,,\ \frac{1}{6}\,,\ \frac{1}{6}\,,\ \frac{1}{12}\,,\ \frac{1}{12}\,,\ \frac{1}{6}\right]$$

The probability that $f_i = 0$

$$\pi_0 + \pi_3 + \pi_4 + \pi_7 = \frac{2}{3}$$

The probability that $f_{i+1} = 0$ when $f_i = 0$

$$\frac{\displaystyle\sum_{j=0,3,4,7} \pi_0\mathcal{P}_{0j} + \pi_3\mathcal{P}_{3j} + \pi_4\mathcal{P}_{4j} + \pi_7\mathcal{P}_{7j}}{\pi_0 + \pi_3 + \pi_4 + \pi_7} = \frac{1}{2}$$

Comparing the m-ary Methods

$$T_r(k,d) = k - d + (1 - P(d))(\frac{k}{d} - 1) + \tau_d - 3$$

$$T_s(k,d) = k - d + (1 - 2^{-d})(\frac{k}{d} - 1) + 2^d - 2$$

	standard	recoding
binary	$\frac{3}{2}\ k - \frac{3}{2}$	$\frac{4}{3}\ k - \frac{4}{3}$
quaternary	$\frac{11}{8}\ k - \frac{3}{4}$	$\frac{4}{3}\ k - \frac{2}{3}$
octal	$\frac{31}{24}\ k - \frac{17}{8}$	$\frac{23}{18}\ k - \frac{75}{18}$

d	$T_s(k,d)/k$	$T_k(k,d)/k$
1	1.50000	1.33333
2	1.37500	1.33333
3	1.29167	1.27778
4	1.23437	1.22917
5	1.19375	1.19167
6	1.16406	1.16319
7	1.14174	1.14137
8	1.12451	1.12435

Comparing the m-ary Methods

For constant d as k gets larger, we have

$$\lim_{k \to \infty} \frac{T_r(k,d)}{T_s(k,d)} \; = \; \frac{(d+1)2^d - \frac{4}{3}}{(d+1)2^d - 1} \; < \; 1$$

However, when we consider the optimal values of d for every k, we obtain

$$\frac{T_r(k,d_r)}{T_s(k,d_s)} \; > \; 1$$

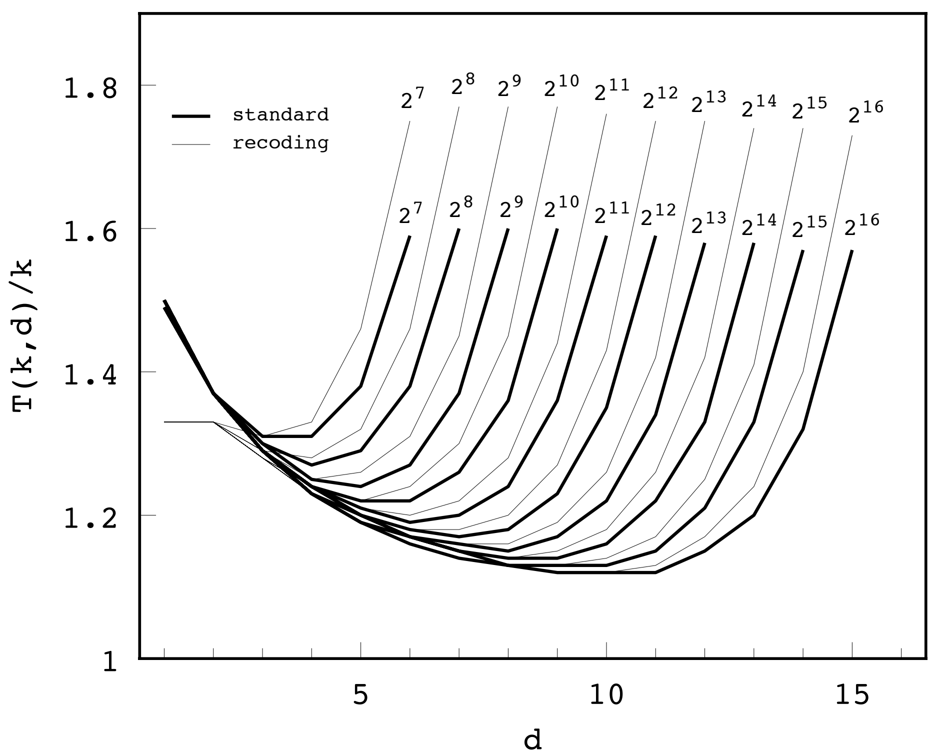
as k gets larger

We have shown that $d_r < d_s$, and

$$\frac{T_r(k,d_r)}{T_s(k,d_s)} \approx \frac{1 + \frac{1}{d_r}}{1 + \frac{1}{d_s}} \; > \; 1$$

Comparing the m-ary Methods

k	d_s	$T_s(k,d_s)$	d_r	$T_r(k,d_r)$
128	4	168	3	168
256	4	326	4	328
512	5	636	4	643
1024	5	1247	5	1255
2048	6	2440	6	2458
4096	7	4795	7	4836
8192	8	9457	7	9511
16384	8	18669	8	18751
32768	9	36902	9	37070
65536	10	73095	10	73433



Modular Multiplication

Given $A, B < n$, compute $P = A \cdot B \bmod n$

Methods:

- Multiply and reduce:

  Multiply: $P' = A \cdot B$ ($2k$-bit number)

  Reduce: $P = P' \bmod n$ (k-bit number)

- Interleave multiply and reduce steps

- Montgomery's method

Montgomery's Method

This method replaces division by n operation with division by 2^k (*Montgomery 85*)

Assuming n is a k-bit odd integer, we assign $r = 2^k$, and map the integers $a \in [0, n-1]$ to the integers $\bar{a} \in [0, n-1]$ using the one-to-one mapping

$$\bar{a} = a \cdot r \pmod{n}$$

We call $\bar{a}$ the n-residue of a

The **Montgomery product** of two n-residues is defined as

$$\text{MonPro}(\bar{a}, \bar{b}) = \bar{a} \cdot \bar{b} \cdot r^{-1} \pmod{n}$$

where r^{-1} is the inverse of r modulo n

Montgomery Product

Property of the Montgomery product:

If $c = a \cdot b \bmod n$, then $\bar{c} = \text{MonPro}(\bar{a}, \bar{b})$

$$\begin{aligned}\bar{c} &= a \cdot b \cdot r^{-1} \pmod{n} \\ &= (a \cdot r) \cdot (b \cdot r) \cdot r^{-1} \pmod{n} \\ &= \text{MonPro}(\bar{a}, \bar{b})\end{aligned}$$

In order to compute $\text{MonPro}(\bar{a}, \bar{b})$, we need n'

$$r \cdot r^{-1} - n \cdot n' = 1$$

(Use the extended Euclidean algorithm)

function $\text{MonPro}(\bar{a}, \bar{b})$
1. $t := \bar{a} \cdot \bar{b}$
2. $u := (t + (t \cdot n' \bmod r) \cdot n)/r$
3. **if** $u \geq n$ **then return** $u - n$ **else return** u

Only modulo r arithmetic is required

Montgomery Exponentiation

Montgomery's method is not suitable for a single modular multiplication since pre-processing operations are time consuming

function ModExp(M, e, n) { n is odd }

1. Compute n' using Euclid's algorithm
2. $\bar{M} := M \cdot r$ mod n
3. $\bar{C} := 1 \cdot r$ mod n
4. **for** $i = h - 1$ **down to** 0 **do**
5. $\quad \bar{C} :=$ MonPro($\bar{C}, \bar{C}$)
6. $\quad$ **if** $e_i = 1$ **then** $\bar{C} :=$ MonPro($\bar{C}, \bar{M}$)
7. $C :=$ MonPro($\bar{C}, 1$)
8. **return** C

Note for Step 7:

$$\begin{aligned} C &= (C \cdot r) \cdot 1 \cdot r^{-1} \pmod{n} \\ &= \text{MonPro}(\bar{C}, 1) \end{aligned}$$

Algorithms for Montgomery Product

- The Dussé-Kaliski Method
- The Product Scanning Method
- The Modified Dussé-Kaliski Method
- The Product Interleaving Method
- The m-ary Add-Shift Method

Computer wordsize: w bits, radix $W = 2^w$

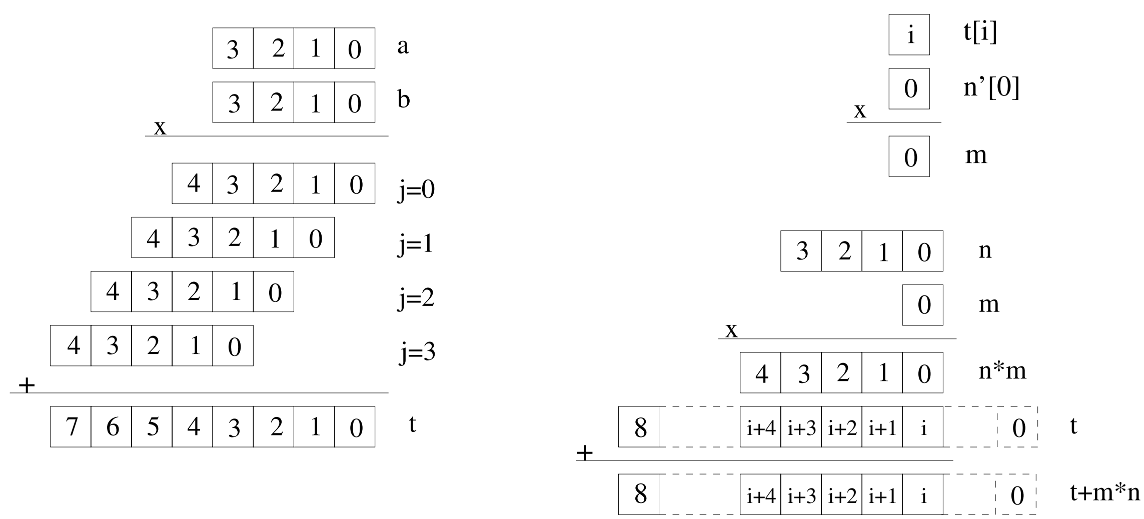
Numbers are s words:

$$n_0 \ , \ n_1 \ , \ \cdots \ , \ n_{s-1}$$

The Montgomery radix: $r = 2^{sw}$

Dussé-Kaliski Method

- First compute $t = a \cdot b$
- Interleave the computations of
  - $m = t \cdot n' \bmod r$
  - $u = (t + m \cdot n)/r$



- Squaring optimization when $a = b$
- Requires n'_0 instead of n'

$$
\begin{aligned}
2^{sw} \cdot 2^{-sw} - n \cdot n' &= 1 \\
-n_0 \cdot n'_0 &= 1 \pmod{2^w}
\end{aligned}
$$

- Requires $2s + 2$ words of temporary space

Montgomery Algorithms

Product Scanning: Interleaves computation of $a \cdot b$ and $m \cdot n$ by scanning the words of m (*Kaliski 93*)

We also use the same space to keep m and u, reducing the temporary space to $s + 3$ words

Modified Dussé-Kaliski: The computation of $a \cdot b$ is split into 2 loops, and the second loop is interleaved with the computation of $m \cdot n$

We show that $s + 2$ words of space suffice

Product Interleaving: The computation of $a \cdot b$ and $m \cdot n$ is performed in a single loop

This method also requires $s + 2$ words of space

The Binary Add-Shift

The computation of $u = a \cdot b \cdot r^{-1} \pmod{n}$ for an odd n and $r = 2^k$

$$2^{-k} \cdot (a_{k-1}2^{k-1} + a_{k-2}2^{k-2} + \cdots + a_0) \cdot b \bmod n$$

The multiplicative factor 2^{-k} reverses the direction of summation, i.e., we start multiplying a and b from the least significant bit:

$$u = (a_{k-1}2^{-1} + a_{k-2}2^{-2} + \cdots + a_0 2^{-k}) \cdot b \bmod n$$

> $u := 0$
> **for** $i = 0$ **to** $k - 1$
> $u := u + a_i \cdot b$
>   **if** u is odd **then** $u := u + n$
> $u := u/2$

The m-ary method proceeds word by word, and multiplies the current word of a by b, and then adds it to u

The m-ary Add-Shift

Then, an integer multiple of n is added to u to make its least significant word equal to zero:

$$U := u + X \cdot n$$

For $U = 0 \bmod 2^w$, we get $0 = u_0 + X \cdot n_0$, and

$$X = -u_0 \cdot n_0^{-1} \pmod{2^w}$$

Note that $-n_0^{-1} \pmod{n}$ is equal to n'_0 since

$$\begin{aligned} 2^{sw} \cdot 2^{-sw} - n \cdot n' &= 1 \pmod{2^w} \\ -n_0 \cdot n'_0 &= 1 \pmod{2^w} \end{aligned}$$

$u := 0$
for $i = 0$ **to** $s - 1$
  $u := u + a_i \cdot b$
  if $u_0 \neq 0$ **then** $u := u + (u_0 \cdot n'_0 \bmod 2^w) \cdot n$
  $u := u/2^w$

The m-ary method requires $s+1$ words of space

Comparing Montgomery Algorithms

Operation and space requirements:

	Mul	Add	Read/Write	Space
DK	$2s^2+s$	$4s^2+4s$	$8s^2+13s+2$	$2s+2$
PS	$2s^2+s$	$6s^2$	$14s^2+15s$	$s+3$
MDK	$2s^2+s$	$4s^2+4s$	$9.5s^2+11.5s$	$s+2$
PI	$2s^2+s$	$4s^2+4s$	$12s^2+15s$	$s+2$
MAS	$2s^2+s$	$4s^2+2s$	$8s^2+9s$	$s+1$

Timings in milliseconds on a i486DX2-66:

	512 bits		1024 bits		2048 bits	
	C	ASM	C	ASM	C	ASM
DK	1.01	0.20	3.66	0.74	14.45	2.84
PS	1.05	0.19	4.04	0.71	16.04	2.76
MDK	1.17	0.20	4.60	0.80	18.30	3.13
PI	1.03	0.19	4.14	0.73	16.44	2.87
MAS	0.94	0.16	3.71	0.60	14.78	2.29

References

P. Downey, B. Leony, and R. Sethi. Computing sequences with addition chains. *SIAM Journal on Computing*, 3:638–696, 1981.

S. R. Dussé and B. S. Kaliski Jr. A cryptographic library for the Motorola DSP56000. *Eurocrypt '90*, pages 230–244. Springer, 1990.

Ö. Eğecioğlu and Ç.K. Koç. Exponentiation using canonical recoding. *Theoretical Computer Science*, 129(2):407–417, 1994.

B.S. Kaliski Jr. The Z80180 and big-number arithmetic. *Dr. Dobb's Journal*, 50–58, September 1993.

D.E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, Volume 2, Addison-Wesley, 1981.

N. Koblitz. *A Course in Number Theory and Cryptography*, 2nd Edition, Springer, 1987.

Ç.K. Koç. High-Speed RSA Implementation. Technical Report TR–201, RSA Laboratories, November 1994.

A.J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer, 1993.

V. Miller. Uses of elliptic curves in cryptography. *Crypto '85*, pages 417–426. Springer, 1985.

P.L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, April 1985.

J.-J. Quisquater and C. Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics Letters*, 18:905–907, October 1982.

G. W. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1960.

A. Schönhage. A lower bound for the length of addition chains. *Theoretical Computer Science*, 1:1–12, 1975.