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

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** 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 because preprocessing operations are time consuming

```
**function** ModExp($M, e, n$) { $n$ is odd }
1.  Compute $n'$ using Euclid's algorithm
2.  $\bar{M} := M \cdot r \bmod n$
3.  $\bar{C} := 1 \cdot r \bmod n$
4.  **for** $i = h-1$ **down to** $0$ **do**
5.      $\bar{C} :=$ MonPro($\bar{C}, \bar{C}$)
6.      **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

Separated Operand Scanning

First computes $t = a \cdot b$ and then interleaves the computations of $m = t \cdot n' \bmod r$ and $u = (t + m \cdot n)/r$. Squaring can be optimized.

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

Finely Integrated Product Scanning

Interleaves computation of $a \cdot b$ and $m \cdot n$ by scanning the words of m

It uses the same space to keep m and u, reducing the temporary space to $s + 3$ words

Coarsely Integrated Hybrid Scanning

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

This method also requires $s+3$ words of space

Montgomery Algorithms

Finely Integrated Operand Scanning

The computation of $a{\cdot}b$ and $m{\cdot}n$ is performed in a single loop

This method also requires $s{+}3$ words of space

Coarsely Integrated Operand Scanning

Improves the SOS method by integrating the multiplication and reduction steps. Instead of computing the entire product $a \cdot b$, then reducing, we alternate between iterations of the outer loops for multiplication and reduction.

This method also requires $s{+}3$ words of space

Comparing Montgomery Algorithms

Operation and space requirements:

	Add	Read/Write	Space
SOS	$4s^2 + 4s + 2$	$8s^2 + 13s + 5$	$2s + 2$
FIPS	$6s^2 + 2s + 2$	$14s^2 + 16s + 3$	$s + 3$
CIHS	$4s^2 + 4s + 2$	$9.5s^2 + 11.5s + 3$	$s + 3$
FIOS	$5s^2 + 3s + 2$	$10s^2 + 9s + 3$	$s + 3$
CIOS	$4s^2 + 4s + 2$	$8s^2 + 12s + 3$	$s + 3$

All of these five methods require $2s^2 + s$ multiplications.

Timings in milliseconds on a i486DX4-100:

	512 bits		1024 bits		2048 bits	
	C	ASM	C	ASM	C	ASM
SOS	1.01	0.20	3.66	0.74	14.45	2.84
FIPS	1.05	0.19	4.04	0.71	16.04	2.76
CIHS	1.17	0.20	4.60	0.80	18.30	3.13
FIOS	1.03	0.19	4.14	0.73	16.44	2.87
CIOS	0.94	0.16	3.71	0.60	14.78	2.29

Cryptographer's Wish List

(From Computer Architect)

- Single instruction cycle for Multiply-Add:

$$(C, S) := S + A * B + C$$

A, B, C, and S are 32-bit integers

- Single instruction cycle for Square-Add:

$$(D, C, S) := S + 2 * A * B + C + D * 2^{32}$$

A, B, C, and S are 32-bit, D is 1-bit integers.

- Single instruction cycle for Multiply-Add:

$$(C(x), S(x)) := S(x) + P(x) * Q(x) + C(x)$$

$A(x)$, $B(x)$, $C(x)$, and $S(x)$ are degree 31 (length 32) polynomials with coefficients from the field $GF(2)$.

Cryptographer's Wish List

Example:

$$\begin{aligned} P(x) &= x^3 + x^2 &&= 1100 \\ Q(x) &= x^3 + x^2 + 1 &&= 1101 \end{aligned}$$

$$\begin{aligned} (x^3) * (x^3 + x^2 + 1) &= x^6 + x^5 + x^3 \\ (x^2) * (x^3 + x^2 + 1) &= x^5 + x^4 + x^2 \\ P(x) * Q(x) &= x^6 + x^4 + x^3 + x^2 \end{aligned}$$

$$\begin{aligned} (1000) * (1101) &= 1101000 \\ (0100) * (1101) &= 0110100 \\ P(x) * Q(x) &= 1011100 \end{aligned}$$