# Justification of Montgomery Modular Reduction

By: Dr. Salem Sherif Elfard

Department of Computer Science, Faculty of Science

University of Az-zawia , Zawia-Libya

E-mail: s_elfard@yahoo.com

## *Abstract:*

*one of the most known and widely used methods in Cryptography is the method suggested by Peter Montgomery; this method is based on the changing of the original reduction modulo by some other convenient modulo, the original Montgomery paper gives the algorithm without any considerations leading to that algorithm.*

## *Key Words:*

*Cryptography, Montgomery Algorithm, modulo reduction, convenient modulo, multiple precision*

## Introduction:

The Montgomery method is the most famous and untraditional, by representing the residue classes **modulo *m*** in a non standard way, Montgomery's method replaces a division by *m* with a multiplication followed by division by a power of *b*. this operation will be called Montgomery reduction.

There are different forms of Montgomery that can be applied to the modular exponentiation problem. There are bit-serial architectures [1], where special purpose circuits perform multiplication and reduction simultaneously. Recently, Fischer and Seifert [2] made an interesting observation that there exists a duality between multiplication and modular reduction.

**Justification of Montgomery Scheme:** first of all let us describe some ideas which allow to obtain this algorithm in natural way as consequence of the number theory facts, in the original Montgomery paper [3], there is no such justification.

As well known in realization of number of data protection such that RSA or logarithmic exchange fast modular reduction method over large numbers are required. The effectiveness of this scheme to much extent depends on the effectiveness of algorithm of modular reduction.

Traditional method of representing large numbers is their representation as the representation in calculation base with some basis *b*. In computer applications *b* is usually defined as power of 2 which is equal to a computer word size [4]. Arithmetic over such numbers (addition, subtraction, multiplication, division, powers) is called multiple precision arithmetic.

Let *m* and *x* be multiple precision numbers with calculation base *b*,

DOI : 10.5121/acij.2012.3605

$$m = \sum_{i=1}^{k-1} m_i b^i, \quad 0 < m_{k-1} < b \quad \text{and} \quad 0 \le m_i < b, \; i = 0,1,2,\ldots\ldots k\text{-}2$$

$$m = \sum_{i=0}^{l-1} x_i b^i, \quad 0 < x_{l-1} < b \quad \text{and} \quad 0 \le x_i < b, \; i = 0,1,2,\ldots\ldots l\text{-}2$$

Computing *x mod m* is called modular reduction, *x* is an argument.

There are a few known modular reduction algorithms: classical method, Barrett's method and Montgomery's method. As was mentioned by many researches Montgomery reduction is the fastest in computing a reasonably long seria of modular reductions, for instance in computing exponential function $x^y$ ***mod*** *z*.[5]. It turns out that Montgomery method is based on the simple well known fact of representing ***gcd*** (greatest common divisor) of tow integers as linear combination of these integers .

Let ***x*** and ***y*** be tow positive integers and let ***d = gcd (x,y)***. then ***d*** could be expressed as a Linear combination of ***x*** and ***y*** , ***Ax – By = d.***
Moreover integers ***A*** and ***B*** could be found effectively on logarithmic on ***x*** and ***y*** time. This fact could be easily obtained for instance from Euclidean algorithm of finding ***gcd*** of tow integers.

If ***x*** and ***y*** are coprime numbers then "greatest common devisor" ***gcd (x,y) =1***, and one can find integers ***A*** and ***B*** such that the following relation holds true :

***Ax – By =1*** (1)

In this case for any integer ***k*** the following holds true :

***(A - ky)x - (B - kx)y =1***. (2)

This implies that if ***B – kx = B mod x***, then ***A - ky = A mod y*** .the inverse statement is also true . thus it is possible to consider ***A*** and ***B*** in representation (1) to be remainders in modulo respectfully ***y*** and ***x***.

Computations of type ***x mod m*** for some integers **m** are simple to be fulfilled and for some others are complex . for instance when $\boldsymbol{m = 2^k}$**,** **k** – size of computer word, this operation is very easy and inexpensive. So an idea arise to transform reduction computations of arbitrary modulo to reductions only of "Good" modulo [6].

Firstly this idea was developed by P. Montgomery he suggested a method for computing arithmetic operations on modulo ***m*** in which operations of addition and subtraction are practically unchanged but multiplication is slightly changed on a simple procedure not using reductions modulo ***m.*** Montgomery method demands nonstandard representation of residue classes modulo m and thus some necessary precomputations of input initial values should be performed in computer programs. Such precomputations are done only once before running a program and do not affect the speed of program executions.

Therefore; Montgomery method is very effective only in programs with active use of modular reduction inside bodies of cycle. Typical example is modular exponential function $x^y$ ***mod*** *z* .

Let ***m*** is a given modulo; one can choose ***R*** which is coprime with ***m*** and such that operations ***x mod R*** and ***x div R*** computationally "Good", ***R > m***. there exists the representation of 1 as a linear combination of ***R*** and ***m***, which could be found in logarithmic time:

$RR^{-1}$ ***– mm' =1*** (3)

Where $0 < R^{-1} < m$ , ***0 < m' < R*** , ***m' = -m mod R***

Let $x$ be some integer, it implies from relation 3 that $xR^{-1}R - xm'm = x$. the principal moment for justification the Montgomery scheme is the following fact analogous to the relation (3) which could be verified by simple algebraic computations.

For any integer k the identity relation holds:

$(xR^{-1} - km) R - (xm' - kR)m = x$. (4)

Implies that; $xR^{-1} - km = ((xm' - kR)m + x) / R$.

let $k = xm'$ *div* $R$. and $s = xm' - kR = xm'$ *mod* $R$. as $s \geq 0$ and $x \geq 0$, then such a choice of $k$ the value $xR^{-1} - km$ is also positive, so we obtain the following result.

**Montgomery theorem** The value $(xm'$ *div* $R) m + x) /R$ is an integer and

$xR^{-1} \equiv (((xm'$ *div* $R) m+x )/ R)$ *mod* $m$.

Assume that $x < Rm$. Consider the expression $(xR^{-1} - (k+2) m) R - (xm' - (k+2) R) m = x$.

this expression is of equal value to $(xR^{-1} - (k+2) m) R + (R - s) m\ Rm = x$. as $R - s \geq 0$ and according to the made conjecture $0 \leq x < Rm$, then $xR^{-1} - (k = 2) < 0$.

Let's consider the expression $(xR^{-1} - (k+1) m) R - (xm' -(k+1) R) m = x$.

this expression is equivalent to $(xR^{-1} - (k+1) m) R + (R - s) m = x$.

where $R - s \geq 0$. If $xR^{-1} - (k+1) m < 0$ *then* $xR^{-1} - km = xR^{-1}$ *mod* m. If $xR^{-1} - (k+1) m \geq 0$ then $xR^{-1} - (k+1) m = xR^{-1}$ *mod* $m$. therefore $xR^{-1}$ *mod* $m$. where $x < Rm$ could fifer from

$((xm'$ *div* $R) m+x )/ R$ ***maximum on*** $m$.

It gives the Montgomery scheme:

```
int function REDC (int x)
    t ≡ ( x mod R) m' mod R;
    g ≡ ( x + tm)/R;
 if ( g ≥ m) return (g –m);
   else return (g).
```

The algorithm is based on the fact that the computation of $xR^{-1}$ *mod* $m$ can be done very efficiently by the algorithm REDC [6].

The Montgomery multiplication algorithm speeds up the modular multiplications and squaring required for exponentiation [7]. It computes the Montgomery product *MonPro*$(a,b)=abr^{-1}$ *mod* $n$

**given** $a ,b < n$ and $r$ such that the greatest common denominator $(n,r) =1$.

To describe the Montgomery reduction algorithm, we need an additional quantity, $n'$, the integer with property $rr^{--1}\ nn'=1$. We can compute both integers $r^{-1}$ and $n'$ with the extended Euclidean algorithm. We compute MonPro$(a,b)$ as follows: function *MonPro*$(a,b)$

```
t:=ab
u:=[t+(tn' mod r)n]/r
if u >= n then return u-n,   else return u
```

However, we did not take into account the space required to keep the input and output values $a,b,n,n_0'$, and $u$

The details of using Montgomery scheme and comparisons with other methods as; Barrett's method, classical method, could be found in [4, 8, 9, 10].

The arrangement of applying the modular operation after completing the multiplication is very expensive because the result of the multiplication by $\boldsymbol{2^{2n-k}}$ may be much greater than the modulus and a large amount of hardware will be required to handle it [11] However, the operation can be simplified by introducing the modular reduction after each multiplication by 2 as the following:

$$[((((((((a^{-1}2^{k-n}).2)\ mod\ p).2)\ .2)\ mod\ p).2)\ mod\ p)]=a^{-1}2^{n}mod\ p$$

The modular reduction operation is performed by a subtraction of $p$ whenever the number exceeds $p$ [7].

**Conclusion and Results:** in this paper several contributions have been achieved as summarized below:

- In arithmetic computation Montgomery reduction is an algorithm introduced by Peter Montgomery that allows modular arithmetic to be performed efficiently when the modulus is large.
- I proposed a theoretical justification of Montgomery modular reduction. And suggested a simple theoretical basis for Montgomery method which could be generalized for other applications and even for other schemes based on different basic relations between a given modulo and chosen one.

## References:

[1] A. F. Tenca and Ç. K. Koç, "A scalable architecture for Montgomery multiplication". In Proc. 1st Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES'99), LNCS 1717, p. 94 ff., Springer, Heidelberg, August, 1999.

[2] W. Fischer and J.-P. Seifert, Duality between Multiplication and Modular Reduction, IACR ePrint Archive, 2005.

[3] P.L.Montgomery, "Modular multiplication without trial division" Mathematics of computation. Vol. 44, 1985, pp.[519-521].

[4] A. F. Tenca and Ç. K. Koç, "A scalable architecture for Montgomery multiplication". In Proc. 1st Int. Workshop on Cryptographic Hardware and Embedded Systems (CHES'99), LNCS 1717, p. 94 ff., Springer, Heidelberg, August, 1999.

[5] A.V. Anisimov "Modulo Reduction Method" conference of Data Protection no 2. pp. 15-20, Ukraine, 1999.

[6] Jean-Claude Bajard, Laurent-Stephane Didier, and Peter Komerup." An RNS Montgomery modular multiplication algorithm". IEEE Transaction on Computers, 47(7):766–776, July 1998

[7] Fredrik Gundersen, Gjøvik University College, "Implementing modular arithmetic using OpenCL", PhD Thesis 07/2010.

[8] A.Bosselares, R. Govaerts and J. Vandewalle, "Comparison of three modular reduction

functions", In Advances in Cryptology – CRYPTO'93, LNCS 773, Spring – Verlag, 1994, pp.175-186

[9] S.R. Dusse, B.S. kaliski "A cryptographic library for the Motorola DSP56000", advanced in Cryptology, Eurocrypt, Lect. Notets Comput. Sci.- 1991, No 473, pp. 230 – 244.

[10] W. Hasenplaugh, G. Gaubatz, V. Gopal, "Fast Modular Reduction", 18th IEEE Symposium on Computer Arithmetic(ARITH'07), 2007.

[11] Blake, Seroussi, and Smart, "Elliptic Curves in Cryptography", Cambridge University Press: New York, 1999.