Efficient Modular Exponentiation Using Common-Multiplicand Multiplication and Signed-Digit Recoding Techniques

Chia-Long Wu
Department of Aviation & Communication Electronics, Chinese Air Force Institute of Technology, Kaohsiung 82042, Taiwan, R.O.C.
E-mail: chialongwu@seed.net.tw

Der-Chyuan Lou* and Te-Jen Chang*
***Department of Electrical Engineering, Chung Cheng Institute of Technology, National Defense University, Tahsi, Taoyuan 33509, Taiwan, R.O.C.**
E-mail: dclouprof@gmail.com

Abstract

In many computation problem, the modular exponentiation is a common operation for scrambling secret data and is used by several public-key cryptosystems, such as the RSA method. In this paper, an algorithm is proved, which combines binary exponentiation method, the CMM (common-multiplicand multiplication) method, and the SDR (signed-digit recoding) method for fast modular exponentiation. The proposed algorithm developed in this paper can be easily implemented in general signed-digit computing machine, and is therefore well suited for parallel implementation to fast evaluate modular exponentiation. The computational complexity of the proposed algorithm is $0.46m^3$-$0.92m^2$-$3.67m$ multiplications, which is less than $0.75m^3$+$0.75m$ multiplications in Dusse-Kaliski algorithm, $0.625m^3$+m^2 multiplications in Ha-Moon algorithm, where m is the bit-length of the exponent.

***Keywords*:** *CMM, SDR, public-key cryptosystem, Montgomery modular reduction.*

摘要

在許多計算的問題上，模指數是一個常見的問題，針對公開金鑰密碼系統的加減密而言，更是如此。在本文中所提出快速模指數的方法，結合了二元模指數法、共同被乘數乘法以及符號元編碼法。這個方法可以實現在以符號元為計算基礎的機器，亦適用在平行處理的運算上。我們提出的方法需要 $0.46m^3$-$0.92\ m^2$-$3.67m$個乘法量，而Dusse-Kaliski學者提出的方法，需要 $0.75m^3$+$0.75m$個乘法量，Ha-Moon學者提出的方法，仍需要 $0.625m^3$+m^2個乘法量，相較之下，我們提出的方法，複雜度低於其他兩派學者所提出的方法。

關鍵詞：共同被乘數乘法、符號元編碼法、公開金鑰密碼系統、蒙哥馬利模減法。

1. Introduction

Modular exponentiation is the cornerstone computations performed in public-key cryptosystem. The modular exponentiation problem can be described as follows. Given M (message), E (public key), and N (modulus), compute the ciphertext $C \equiv M^E \bmod N$. To evaluate the result for modular exponentiation of $M^E \bmod N$, the very intuitive way is to break the modular exponentiation into a series of modular multiplications.

An efficient evaluation of the modular exponentiation of $M^E \bmod N$ is very useful for public key cryptosystem and we need fast multiplication designs or novel exponentiation methods such as the Montgomery modular reduction method [1], binary exponentiation method [2], common-multiplicand multiplication (CMM) method [3-4]. In this paper, a method is proposed for speeding up modular exponentiation by using the binary exponentiation method, the common-multiplicand multiplication method, and the signed-digit recoding method.

This paper is organized as follows. Some related

works for efficiently solving the modular exponentiation are introduced in Section 2. In Section 3, a fast modular exponentiation method is proposed. Then, the computational complexity of the proposed method is analyzed in Section 4. Finally, we conclude our work in Section 5.

2. The Mathematical Preliminary

In 1985, ElGamal proposed the ElGamal scheme [5], which can be used for the cryptographic applications of both digital signatures and encryptions. Modular exponentiation is composed of repetition of modular multiplication and is often the dominant part of modern cryptographic method. Taking the RSA public key cryptosystem for example, both the encryption and decryption operations are accomplished by modular exponentiation.

In this section, we will give a brief review to the binary exponentiation method, CMM method, and SDR method.

2.1. The Binary Exponentiation Method

The most commonly used methods for computing exponentiation are the binary exponentiation methods. Its basic idea is to compute modular exponentiation by using the binary expression of the exponent E and the exponentiation operation is broken into a series of squaring and multiplication operations. Assume m denotes the bit-length of the exponent E, the exponent E can be expressed in binary representation as $(e_m e_{m-1}\ldots e_2 e_1)_2$ and $E = \sum_{i=1}^{m} e_i * 2^i$, where $e_i \in \{0,1\}$.

There are two useful modular exponentiation methods [2] in binary exponentiation methods, i.e., the LSB (least-significant-bit) method and the MSB (most-significant-bit) method. The LSB method is depicted in Algorithm 1. The MSB method is depicted in Algorithm 2. Both the MSB and LSB binary methods have the same computations for multiplication and squaring operations, therefore they have the same computational complexity. Take m-bit length exponent, for the average case, we assume the occurrence probabilities for both bit "1" and bit "0" are the same. Then, the expectation numbers for bit "1" and bit "0" are both $\frac{m}{2}$. Therefore, on average the complexities for both methods are $2*\frac{m}{2}+1*\frac{m}{2}=1.5m$ modular multiplications.

Algorithm 1: (LSB Binary Modular Exponentiation Method)
Input: Message: M; Exponent: $E = (e_m e_{m-1}\ldots e_2 e_1)_2$; $0 \le i \le m$ and $e_i \in \{0,1\}$
Output: Ciphertext: $C = M^E \bmod N$

```
begin
C = 1
S = M
for i = 1 to m do                    /*scan from right to left*/
  begin
    if (ei = 1) then C = C·S mod N
                                     /*modular multiplication*/
    S = S*S mod N                    /*modular squaring*/
  end
end
```

Algorithm 2: (MSB Binary Modular Exponentiation Method)
Input: Message: M; Exponent: $E = (e_m e_{m-1}\ldots e_2 e_1)_2$; $0 \le i \le m$ and $e_i \in \{0,1\}$
Output: Ciphertext: $C = M^E \bmod N$

```
begin
C = 1
for i = m to 1 do                    /*scan from left to right*/
  begin
    C=C×C mod N                      /*modular squaring*/
    if (ei=1) C = C×M mod N
                                     /*modular multiplication*/
  end
end
```

2.2. The CMM (Common-Multiplicand Multiplication) Method

In 1993, Yen and Laih [3] developed the method of common-multiplicand multiplication to improve the performance of the binary exponentiation method. Here we focus on the computations of $\{X*Yi \mid i = 1, 2, \cdots, t; t \geq 2\}$, where t is the number of division group. An arithmetic method [6-7] is presented and applies to the binary exponentiation method for the efficient evaluation of exponentiation.

The following variables are required in the CMM method.

$$Y_{common} = \mathbf{AND}_{i=1}^{t} Y_i, \quad (1)$$

$$Y_{i,c} = Y_i \ \mathbf{XOR}\ Y_{common}. \quad \text{for } i = 1, 2, \cdots, t$$

/* t is the number of division group/(2)

Hence, each can be represented as:

$$Y_i = Y_{i,c} + Y_{common}. \quad (3)$$

Therefore, the common-multiplicand multiplications $X*Y_i$ ($i = 1, 2, \cdots, t$) can be computed with the assistance of $X*Y_{common}$ as

$$X*Y_i = X*Y_{i,c} + X*Y_{common} \text{ for } i = 1, 2, \ldots, t. \quad (4)$$

The basic idea of CMM method is to extract the common parts of multiplicands and then save the number of binary additions for the computation of common parts. By using the above method, the computations $\{X*Y_1, X*Y_2\}$ can be represented as $\{X*Y_{1,c} + X*Y_{common}, X*Y_{2,c} + X*Y_{common}\}$.

Let both X and Y_i be m-bit integers. On average, the Hamming weights of Y_i, Y_{common} and $Y_{i,c}$ are $\frac{m}{2}$, $\frac{m}{2^t}$, and $\frac{m}{2} - \frac{m}{2^t}$, respectively. The total number of binary additions for the common-multiplicand multiplication evaluation of $\{X*Yi \mid i = 1, 2, \cdots, t; t \geq 2\}$ is $\frac{m}{2^t} + t*(\frac{m}{2} - \frac{m}{2^t})$. Without the CMM algorithm, the evaluation of $\{X*Yi \mid i = 1, 2, \cdots, t; t \geq 2\}$ are computed one after another independently using total $t*\frac{m}{2}$ binary addition. The performance improvement [3] of the common-multiplicand multiplication method can be denoted as：

$$\frac{\frac{m}{2}*t}{\frac{m}{2^t} + t*(\frac{m}{2} - \frac{m}{2^t})} = \frac{t}{t + (1-t)*2^{1-t}}. \quad (5)$$

Based on the Eq. (5) shown above, the optimal performance can be obtained as $\frac{4}{3}$ when $t = 2$, which implies we need $1.5m$ ($=2m*\frac{3}{4}$) multiplications by using the CMM method for evaluating $X*Y_1$ and $X*Y_2$.

On average, by applying the CMM method and the binary exponentiation method, exponentiation can be evaluated by using $\frac{(1.5+1)}{2}*m = 1.25m$ multiplications for exponent E being a m-bit integer. Consider the application to public key cryptography for evaluating M^E mod N, where both M and E are 512-bit, the binary exponentiation method and the CMM method combined with the binary exponentiation method require 768 multiplications and 640 multiplications, respectively.

2.3. The SDR (Signed-Digit Recoding) Method

In 1993 [8-9], Arno and Wheeler proposed the signed-digit recoding method. This signed-digit recoding method is defined in Algorithm 3.

Algorithm 3: (Signed-digit Recoding Method)

Input: $E = (e_m e_{m-1} \ldots e_2 e_1)_2$

Output: $E_{SD} = (r_m r_{m-1} \ldots r_2 r_1)_{SD2}$, where $0 \leq i \leq m$ and $e_i \in \{0, 1, \bar{1}\}$

begin

$c_1 = 0$; $r_{m+2}=0$; $r_{m+1}=0$

for i = 1 to m+1 **do**

begin

$$c_{i+1} = \left\lfloor \frac{c_i + e_i + e_{i+1}}{2} \right\rfloor$$

$$e_i = c_i + e_i - 2c_{i+1}$$

end

end.

3. The Proposed Method

In this section, we give detailed descriptions for the proposed method for fast modular exponentiation used in public-key cryptosystems and information security applications. We propose an efficient method by using common-multiplicand multiplication method, Montgomery reduction method, and signed-digit method for fast evaluating M^E mod N operation as shown in Algorithm 4 and Algorithm 5.

Algorithm 4: (Montgomery Modular Reduction (MMR) Method)

Input: A, B, N

/*A, B and N are m-digit integers in base 2*/

Output: X /*$X = AB*2^{-m} \bmod N$*/

begin

$X = 0$

/*$X = (X_m \ldots X_1 X_0)_2$ where m is the bit-length of X*/

for i = 1 **to** m **do**

/* scan from least significant bit */

begin

$X = X + A_i B$

$Y = X_0 N_0' \bmod 2$

/* $N_0' = -N_0^{-1} \bmod 2$, $N' = -N^{-1} \bmod 2$*/

$$X = \frac{X + Y * N}{2}$$

end

if ($X \geq N$) **then** $X = X - N$

/* $X = AB*2^{-m} \bmod N$ =MMR(AB) */

end

Algorithm 5: (Proposed Method)

Input: M, E_{SD}, N, R /*$E_{SD} = (e_m e_{m-1} \ldots e_2 e_1)_{SD2}$, $R = 2^m$ */

Output: $C_1 = M^{E_{common[1]}}$, $C_2 = M^{E_{1,c[1]}}$, $C_3 = M^{E_{2,c[1]}}$,

$D_1 = M^{E_{common[\bar{1}]}}$, $D_2 = M^{E_{1,c[\bar{1}]}}$, $D_3 = M^{E_{2,c[\bar{1}]}}$

begin

$C_1 = C_2 = C_3 = D_1 = D_2 = D_3 = 2^m$

/*M and N are m-digit integers in base 2*/

$S = M*R \bmod N$ /* $R = 2^m$ */

for i = 1 **to** m **do**

/*scan exponent E_{SD} from right to left */

begin

if (e_{1i}'=1) **then** C_1=MMR(SC_1)

/*evaluate $M^{E_{common}}$ for positive signed-digit*/

if (e_{1i}'=$\bar{1}$) **then** D_1=MMR(SD_1)

/*evaluate $M^{E_{common}}$ for negative signed-digit*/

if (e_{2i}'=1) **then** C_2 = MMR(SC_2)

/*evaluate $M^{E_{1,c}}$ for positive signed-digit*/

if (e_{2i}'=$\bar{1}$) **then** D_2 = MMR(SD_2)

/*evaluate $M^{E_{1,c}}$ for negative signed-digit*/

if (e_{3i}'=1) **then** C_3 = MMR(SC_3)

/*evaluate $M^{E_{2,c}}$ for positive signed-digit*/

if (e_{3i}'=$\bar{1}$) **then** D_3 = MMR(SD_3)

/*evaluate $M^{E_{2,c}}$ for negative signed-digit*/

S = MMR(SS)

end

end

In order to execute the exponentiation operation, we divide the exponent E_{SD} (m-bit) into two equal-length parts as E_1 and E_2, each with of $\frac{m}{2}$ bits. In the proposed method, we put the operation results of positive digit in the register C_1 and C_2, and we put the operation results of negative digit in the registers D_1 and D_2. The C_1 and D_1 are used to store the operation results in decomposition segment E_1 of the signed-digit exponent E_{SD}. Meanwhile, the C_2 and D_2 are used to store the

operation results in the decomposition segment E_2 of exponent E_{SD}.

Our main goal for applying common-multiplicand multiplication technique is that the common part among MMR(SC_1), MMR(SD_1), MMR(SC_2), MMR(SD_2), and MMR(SS) can be therefore computed just once. If we denote T as $S*2^{-i+1}$ mod N, the operations of $S*2^{-i}$ mod N for $1 \le i \le n-2$ are repeatedly computed using the previous computation result of T, where $S*2^{-i}$ mod $N=T*2^{-1}$ mod N. Assume we have two n-bits exponents E_1 and E_2 (where $n=\frac{m}{2}$, and m is the bit-length of the exponent E), the exponentiation operation $M^{E_{SD}}$ can be depicted as: $M^E = M^{E_1 \| E_2} = M^{E_2 * 2^n} * M^{E_1}$ ("||" is the concatenation operator) (6)

Let the decomposition segments E_1 and E_2 be expressed as $E_{1i} = \sum_{i=1}^{m} e_{1i} * 2^i$ and $E_{2i} = \sum_{i=1}^{m} e_{2i} * 2^i$, where e_{1i} and e_{2i} are signed-digits and e_{1i}, $e_{2i} \in \{0, 1, \bar{1}\}$. Moreover, the exponentiations $M^{E_1[1]}$ and $M^{E_2[1]}$ are evaluated for handling positive signed-digit in E_1 and E_2, respectively.

Similarly, the exponentiations $M^{E_1[\bar{1}]}$ and $M^{E_2[\bar{1}]}$ are evaluated for handling negative signed-digit in E_1 and E_2, respectively. Finally, we can output the exponentiation results of "$M^{E_1[1]}$ mod N, $M^{E_2[1]}$ mod N, $M^{E_1[\bar{1}]}$ mod N, and $M^{E_2[\bar{1}]}$ mod N" into four different registers C_1, C_2, D_1, D_2, respectively. By applying the common part extraction technique of the common-multiplicand multiplication method upon the two exponent segments E_1 and E_2, we can obtain the three temporary exponents E_{common}, $E_{1,c}$, $E_{2,c}$ [10-12].

4. Computational Complexity Analyses

In this section, we first analyze the performance of the proposed method and then calculate the number of binary additions and modular multiplications needed. Let the m-bit exponent E be recoded as signed-digit representation E_{SD}. On average, the occurrence probabilities of digits, "1" and "$\bar{1}$", are the same [8]. We can thus have the occurrence probabilities of "0", "1" and "$\bar{1}$" as $\{P_{probability}(0)=\frac{2}{3},\ P_{probability}(1)=P_{probability}(\bar{1})=\frac{1}{6}\}$.

In the proposed method, based on the computational analyses of Montgomery reduction method [6], the probability of executing modular multiplication MMR(SC_1), MMR(SC_2), or MMR(SC_3) in the proposed method is all equivalent to the occurrence probability of signed-digit "1" in E_{SD}. Therefore, the operations MMR(SC_1), MMR(SC_2), and MMR(SC_3) total require

$(3/6)*[1.5m*(n-2)*(n+1)]=(3/4)*m*(n^2-n-2)$ (7)

single-precision multiplications, where m is the bit-length of the exponent E and $n=(m/2)$.

Similarly, operations MMR(SD_1), MMR(SD_2), and MMR(SD_3) total require

$(3/6)*[1.5m*(n-2)*(n+1)]=(3/4)*m*(n^2-n-2)$ (8)

single-precision multiplications.

The operation MMR(SS) requires

$(2/3)*[0.5m*(n-2)*(n+1)]=(1/3)*m*(n^2-n-2)$ (9)

single-precision multiplications.

The Ha-Moon algorithm takes $0.625m^3+m^2$ multiplications [6] and the Dusse-Kalisk algorithm takes $0.75m^3+0.75m$ multiplications [10] for calculating an m-bits exponent exponentiation. However, the proposed method only takes

$(3/4)*m*(n^2-n-2)+ (3/4)*m*(n^2-n-2)+ (1/3)*m*(n^2-n-2)$

$\approx 0.458m^3-0.916m^2-3.666m$ (10)

single-precision multiplications.

Take a 512-bit exponent E for example to evaluate M^E, the number of the multiplications by using the proposed method is 61,263,141 multiplications, the number of the multiplications by using the Dusse-Kaliski algorithm [13-14] is 100,663,680 multiplications, and the number of the multiplications by using the Ha-Moon [11, 15-16] algorithm is 84,148,224 multiplications.

5. Conclusions

As we know, one of the most important computer arithmetic operations for public-key cryptosystem is modular exponentiation. In this paper, a method for

speeding up modular exponentiation is investigated by using the binary exponentiation method, CMM method, and SDR method. Based on the computational complexity analyses, we observe that the proposed algorithm can further speed up, i.e., the total number of multiplications can be reduced. On average, the total number of multiplications can be reduced by about 39.14%, 37.35% as compared with Dusse-Kaliski algorithm, Ha-Moon algorithm.

References

[1] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of Computation*, vol. 44, no. 170, pp. 519-521, April 1985.

[2] D. E. Knuth, *The Art of Computer Programming, Vol. II: Seminumerical Algorithms, 3rd Edition*, MA: Addison-Wesley, 1997.

[3] S.-M. Yen and C.-S. Laih, "Common-multiplicand multiplication and its applications to public key cryptography," *Electronics Letters*, vol. 29, no. 17, pp. 1583-1584, Aug. 1993.

[4] T.-C. Wu and Y.-S. Chang, "Improved generalization common-multiplicand multiplications algorithm of Yen and Laih," *Electronics Letters*, vol. 31, no. 20, pp. 1738-1739, Sept. 1995.

[5] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, pp. 469-472, July 1985.

[6] W. Shugang, C. Shuangching, and K. Shimizu, "Fast modular multiplication using Booth recoding based on signed-digit number arithmetic," *Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS)* , 2002, vol. 2, pp. 31-36.

[7] D.-C. Lou, C.-L. Wu, and C.-Y. Chen, "Fast exponentiation by folding the signed-digit exponent in half," *International Journal of Computer Mathematics*, vol. 80, no. 10, pp. 1251-1259, Oct. 2003.

[8] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Prentice-Hall, 1999.

[9] I. Koren, *Computer Arithmetic Algorithms, 2nd Edition, A. K. Peters*, Natick, MA, 2002.

[10] S. R. Dusse and B. S. Kaliski, "A cryptographic library for the Motorola DSP 56000,"*Proceedings of EUROCRYPT'90 on Advance in Cryptology*, LNCS, Springer-Verlag, 1990, vol. 473, pp. 230-244.

[11] J.-C. Ha and S.-J. Moon, "A common-multiplicand method to the Montgomery algorithm for speeding up exponentiation," *Information Processing Letters*, vol. 66, no. 2, pp. 105-107, April 1998.

[12] A. Avizienis, "Signed-digit number representation for fast parallel arithmetic," *IRE Transaction on Electronic Computers*, vol. EC-10, no. 3, pp. 389-400, Sep. 1961.

[13] S. Arno and F. S. Wheeler, "Signed-digit representations of minimal Hamming weight," *IEEE Transactions on Computers*, vol. 42, no. 8, pp. 1007-1010, Aug. 1993.

[14] S.-M. Yen, "Improved common-multiplicand multiplication and fast exponentiation by exponent decomposition," *IEICE Transactions on Fundamentals*, vol. E80-A, no. 6, pp. 1160-1163, June 1997.

[15] J. Chung and A. M. Hasan, "Low-weight polynomial form integers for efficient modular multiplication," *IEEE Transactions on Computers*, vol. 56, no. 1, pp. 44-57, Jan. 2007.

[16] C.-L. Wu, D.-C. Lou, J.-C. Lai and T.-J. Chang, "Fast modular multi-exponentiation using modified complex arithmetic," *Applied Mathematics and Computation*, vol. 186, no. 2, pp. 1065-1074, March 2007.