Fast Binary Multiplication Method for Modular Exponentiation

Chia-Long Wu, Der-Chyuan Lou* and Te-Jen Chang*

Department of Aviation & Communication Electronics, Chinese Air Force Institute of Technology, Kaohsiung 820, Taiwan, R.O.C.
E-mail: chialongwu@seed.net.tw

***Department of Electrical Engineering, Chung Cheng Institute of Technology, National Defense University, Tahsi, Taoyuan 33509, Taiwan, R.O.C.**
E-mail: dclou@ccit.edu.tw

Abstract

This paper proposes a new fast technique to fast evaluate modular exponentiation which combines a binary exponentiation method, a complement representation method, and a signed-digit representation method. Because modular exponentiation is one of the most time-consuming operations for many cryptosystems, we adopt the proposed method to reduce the computational complexity from $\frac{11k}{8}$ to $\frac{13k}{12}$, where k is the bit-length of the exponent E. We can therefore efficiently speed up the overall performance of the modular exponentiation.

Keywords: **Complement, signed-digit representation, modular exponentiation, binary method, public key cryptography.**

摘要

本論文結合二元乘法、符號元表示法、以及補數表示法三種運算技術提出新的快速模指數演算法。對於大部份的公開金鑰密碼系統而言，模指數運算是相當耗時且佔記憶體的運算。而當我們利用本論文所提出的演算法時，我們可以將整體模指數運算複雜度由 $\frac{11k}{8}$ 有效降至 $\frac{13k}{12}$，其中 k 為指數 E 的長度。

關鍵詞：補數法、符號元表示法、二元法、模指數運算、公開密碼學。

1. Introduction

Modular exponentiation plays an important role in public key cryptosystems. Binary exponentiation is one of the most frequently used arithmetic operations in the microprocessor. There are several modern binary methods which can be used to fast evaluate modular exponentiation. In 2000, Joye and Yen presented a signed-digit representation method [14], which was efficiently signed-digit representations with the digit set {1, 0, $\bar{1}$} to decrease computational complexity. In 2003, Chang, Kuo, and Lin presented a fast algorithm, which takes $\frac{5}{4}k+2$ multiplications by performing complements, where k is the bit-length of the exponent E [4].

In this paper, we describe a new method for speeding up the performance of modular exponentiation. The proposed method combines a binary method, a complement recoding method, and a signed-digit recoding method. The computational complexity of the proposed algorithm is $\frac{13k}{12}$.

The rest of this paper is organized as follows. In Section 2, we first review the binary method, complement recoding method, and signed-digit recoding method. The proposed method for fast evaluate modular exponentiation is detailed described in Section 3. Finally, we put our conclusions for the proposed method and state some future works.

2. Preliminaries

2.1 Binary Method

The exponentiation operation is broken into a series of squarings and multiplications [12] by using the binary method. The basic idea of the binary method is to compute M^E using the binary exponentiation of exponent E.

Assume k denotes the bit-length of the exponent E, which can be expressed in binary representation as $E = (e_{k-1}e_{k-2}...e_1e_0)_2$ and $E = \sum_{i=0}^{k-1} e_i * 2^i$, where $e_i \in \{0,1\}$.

The binary method (also called the square-and-multiply method) scans the bits of exponent E either from right to left (Algorithm 1) or from left to right (Algorithm 2) to accomplishes the exponentiation arithmetic [5, 9, 16].

Algorithm 1: Right-to-left square-and-multiply method.

Input: Exponent: $E = (e_{k-1}e_{k-2}...e_1e_0)_2$;
Message: M
Output: Ciphertext: $C = M^E$

```
C = 1; S = M;
  begin
    for i = 0 to k – 1 do
                    /* scan from right to left */
    begin
      if (e_i = 1) C = C * S;      /* multiply */
      S = S * S;                   /* square */
    end;
  end.
```

Algorithm 2: Left-to-right square-and-multiply method.

Input: Exponent: $E = (e_{k-1}e_{k-2}...e_1e_0)_2$;
Message: M
Output: Ciphertext: $C = M^E$

```
C = 1;
  begin
    for i = k – 1 downto 0 do
                    /* scan from left to right */
    begin
      C = C * C;                   /* square */
      if (e_i = 1) C = C * M;      /* multiply */
    end;
  end.
```

In each step, a squaring is performed and depending on whether the scanned bit-value is equal to 1 or not, a multiplication is also performed. Therefore, the computational complexity of both right-to-left and left-to-right binary methods are $2*\frac{k}{2} + 1*\frac{k}{2} = \frac{3k}{2}$

multiplications, where k is the bit-length of the exponent E.

2.2 Complement Recoding Method

To compute the modular exponentiation of $M^E \bmod N$, we express the exponent E as a binary representation $e_{k-1}e_{k-2}...e_1e_0$. Performing complements is advantageous in the speedup of multiplication computations. We consider the equation shown as follows.

$$E = (10...0)_{(k+1)\text{ bits}} - \overline{E} - 1, \qquad (1)$$

where $\overline{E} = \overline{e_{k-1}}\,\overline{e_{k-2}}\,...\,\overline{e_0}$,

and $\overline{e_i} = 0$ if $e_i = 1$,

$\overline{e_i} = 1$ if $e_i = 0$,

for i = 0, 1, …, k–1.

Based on Eq. (1), we can obtain

$$\begin{aligned} & M^E \bmod N \\ &= [M^{(100...0)_{(k+1)bits}} * M^{-\overline{E}} * M^{-1}] \bmod N \\ &= [M^{(100...0)_{(k+1)bits}} * (M^{\overline{E}})^{-1} * M^{-1}] \bmod N \\ &= [M^{(100...0)_{(k+1)bits}} * (M^{-1})^{\overline{E}} * M^{-1}] \bmod N \\ &= [M_k * (M^{-1})^{\overline{E}} * M^{-1}] \bmod N, \qquad (2) \end{aligned}$$

where M^{-1} is the inverse of M under modulus N. M^{-1} can be pre-computed using the Euclidean algorithm or Euler theory [10]

and $M_k = M^{(100...0)_{(k+1)bits}}$.

We assume k is the bit-length of the exponent E and we can compute M_k in advance.

If Ham(E) > $\frac{k}{2}$, we apply Eq. (1) to compute $M^E \bmod N$. If Ham(E) ≤ $\frac{k}{2}$, we use binary method to compute $M^E \bmod N$ straightforwardly [11].

2.3 Signed-Digit Recoding Method

In a signed-digit number with radix 2, three symbols $\{\bar{1}, 0, 1\}$ are allowed for the digit set, in which 1 and $\bar{1}$ in bit position i represented $+2^i$ and -2^i respectively [6]. The recoding representation is called canonical if it contains no adjacent nonzero digits. The auxiliary carry C_0 is set to 0 and subsequently the binary number A is scanned two bits at a time. The canonically recorded digit B_i and the next value of the auxiliary binary variable C_{i+1} for i = 0, 1, 2, …, n are generated using Table 1.

Table 1 Canonical recoding method

A_{i+1}	A_i	C_i	B_i	C_{i+1}
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, when A = 3038, we compute the canonical signed-digit vectors B as:

$A = (0101111011110) = (10\bar{1}0000\bar{1}000\bar{1}0) = B$

Note that in this example the number A contains nine zero bits and its canonically recorded version contains only four nonzero digits. It shows that the average Hamming weight of a k-bit canonically recorded binary number approaches $\frac{k}{3}$ as $k \to \infty$ [15, 19]. We should note that a number using the digit $\{\bar{1}, 0, 1\}$ is not uniquely represented in binary signed-digit notation [13].

3 The Proposed Method

In Section 2, we describe a binary method, a complement recoding method, and a signed-digit recoding method for exponentiation respectively. Now the proposed method is described as follows.

We assume there are *k* bits in exponent *E*. For M^E, the exponent *E* should be calculated by combining a binary method, a complement recoding method, and a signed-digit recoding method.

So there are two conditions should be discussed as following:

Case 1: Ham(E) > $\frac{k}{2}$ and

Case 2: Ham(E) ≤ $\frac{k}{2}$.

Then the overall computational complexity of the M^E should be represented as follows.

The computational complexity of M^E

= $\frac{1}{2}$ * the computational complexity of case 1

+ $\frac{1}{2}$ * the computational complexity of case 2.

The first and the second items "$\frac{1}{2}$" in the above equation mean the probabilities of Ham(E) > $\frac{k}{2}$ and Ham(E) ≤ $\frac{k}{2}$.

Now we introduce the computational complexities of case 1 and case 2 respectively. At first, we define $\overline{E_{\text{sgn}}}$ a binary signed-digit representation for $\overline{E}$.

- Case 1: Ham(E) > $\frac{k}{2}$

When Ham(E) > $\frac{k}{2}$ in case 1, we take 1's complement for *E*. In other words, it can be represented by Ham($\overline{E}$) ≤ $\frac{k}{2}$. We can replace the pattern for $M^E \bmod N$ with $M^{(10\ldots0)_{(k+1)bits}-\overline{E}-1} \bmod N$. If we combine a complement recoding method and a signed-digit recoding method, we can obtain Ham($\overline{E_{\text{sgn}}}$) ≤ $\frac{k}{2} * \frac{1}{3} = \frac{k}{6}$. So we can obtain

$M^E \bmod N$

= $M^{(10\ldots0)_{(k+1)bits}-\overline{E}-1} \bmod N$

= $M^{(10\ldots0)_{(k+1)bits}-\overline{E_{\text{sgn}}}-1} \bmod N$

The computational complexity of case 1 is:

$$k + \frac{k}{6} * \frac{1}{2} = \frac{13k}{12}, \qquad (3)$$

where *k* represents exponent-bit of *E*.

The first item "*k*" in Eq. (3) denotes the computational complexity of squarings and the second item "$\frac{k}{6} * \frac{1}{2}$" in Eq. (3) denotes the computational complexity of multiplications.

- Case 2: Ham(E) ≤ $\frac{k}{2}$

When Ham(E) ≤ $\frac{k}{2}$ in case 2, we use a left-to-right binary method to compute $M^E \bmod N$ straightforwardly. If we combine a complement recoding method and a signed-digit recoding method, we can also obtain

Ham($\overline{E_{\text{sgn}}}$) ≤ $\frac{k}{2} * \frac{1}{3} = \frac{k}{6}$.

The computational complexity is

$$(k)+(\frac{k}{6}*\frac{1}{2})=\frac{13k}{12} \quad (4)$$

The first item "k" in Eq. (4) denotes the computational complexity of squarings and the second item "$\frac{k}{6}*\frac{1}{2}$" in Eq. (4) denotes the computational complexity of multiplications.

From the above case 1 and case 2 depicted in the proposed method, we can therefore get the overall computational complexity of modular exponentiation "$M^E \bmod N$" as following:

$$(\frac{1}{2}*\frac{13k}{12})+(\frac{1}{2}*\frac{13k}{12})=\frac{13k}{12}. \quad (5)$$

4 Conclusions and future works

In this paper we propose a new method to fast evaluate modular exponentiation, which combines a binary method, a complement recoding method, and a signed-digit recoding method.

The computational complexity of the proposed method is $\frac{13k}{12}$ that are faster than $\frac{11k}{8}$ in [17] and $\frac{13k}{10}$ in [18]. We can efficiently speed up the overall performance of the modular exponentiation.

Recently, some new techniques are studied and proposed to fast evaluate modular exponentiation such as the unified-multipliers method [2, 8], elliptic curve encrypt method [20], and improved Montgomery's method [1, 3, 7]. In the future works, we will try to research and properly adopt some techniques mentioned above to further decrease the modular arithmetic computational complexity for cryptographic usages.

References

[1] A. F. Tenca and C. K. Koc, "A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm," *IEEE Transactions on Computers*, Vol. 52, No. 9, pp.1215-1221, Sep. 2003.

[2] A. F. Tenca, E. Savas, and C. K. Koc, "A Design Framework for Scalable and Unified Multipliers in GF(p) and GF(2^m)," International Journal of Computer Research, Vol. 13, No. 1, pp. 68-83, 2004.

[3] A. Z. Alkar and R. Sonmez, "A Hardware Version of the RSA Using the Montgomery's Algorithm with Systolic Arrays," Integration, the VLSI Journal, Vol. 38, No. 2, pp. 299-307, Dec. 2004.

[4] C.-C. Chang, Y.-T. Kuo, and C.-H. Lin, "Fast Algorithms for Common-Multiplicand Multiplication and exponentiation by performing complements," Proceedings of the 17th International Conference on Advanced Information Networking and Applications, pp. 807-811, IEEE, March 2003.

[5] C. Heuberger and H. Prodinger, "Carry Propagation in Signed Digit Representations," European Journal of Combinatorics, Vol. 24, No. 3, pp. 293-320, April 2003.

[6] C. K. Koc and S. Johnson, "Multiplication of Signed-Digit Numbers," IEE Electronics Letters, Vol. 30, No. 11, pp. 840-841, May 26, 1994.

[7] C.-L. WU, D.-C. LOU, and T.-J. CHANG, "An Efficient Montgomery Exponentiation Algorithm for Cryptographic Applications", Informatica-Institute of Mathematics and Informatics, Vol. 16, No. 3, pp. 449-468, 2005.

[8] D.-C. LOU and C.-L. WU, "Parallel Modular Exponentiation Using Signed-Digit-Folding Technique," Informatica-An International Journal of Computing and Informations, Vol. 28, No. 2, pp. 197-205, 2004.

[9] 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.

[10] D. E. Knuth, The Art of Computer Programming: Vol. II, Seminumerical Algorithms, 3rd edition, Addison Wesley, 1997.

[11] F. Huang and Z. H. Guan, "A Modified Method of a Class of Recently Presented Cryptosystems," Chaos, Solitons and Fractals, Vol. 23, No. 5, pp. 1893-1899, March 2005.

[12] I. Koren, Computer Arithmetic, 2nd edition, A. K. Peters, Natick, MA, 2002.

[13] M. E. Kaihara and N. Takagi, "A Hardware Algorithm for Modular Multiplication/ Division", IEEE Transactions on Computers, Vol. 54, No. 1, Jan. 2005, pp. 12-21.

[14] M. Joyce and S.-M. Yen, "Optimal Left-to-Right Binary Signed-Digit Recoding," IEEE Transactions on Computers, Vol. 49, No. 7, pp. 740-748, July 2000.

[15] 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.

[16] S. Masui, K. Mukaida, M. Takenaka, and N. Torii, "Design Optimization of a High-Speed, Area-Efficient and Low-Power Montgomery Modular Multiplier for RSA Algorithm," IEICE Transactions on Electronics Vol. 88, N.4, pp. 576-581, April 2005.

[17] S.-M. Yen and C.-S. Laih, "Common-Multiplicand Multiplication and its Applications to Public Key Cryptography," IEE Electronics Letters, Vol. 29, No. 17, pp. 1583-1584, Aug. 1993.

[18] S.-M. Yen, "Improved Common-Multiplicand Multiplication and Fast Exponentiation by Exponent Decomposition," IEICE Transactions on Fundamentals, Vol. 80-A, No. 6, pp. 1160-1163, June 1997.

[19] T. Elgamal, "A Public Key Cryptosystem and a Signature Scheme Based on the Discrete Logarithms," IEEE Transactions on Information Theory, Vol. 31, No. 4, pp. 469-472, July 1985.

[20] T.-S. Chen, "A Threshold Signature Scheme Based on the Elliptic Curve Cryptosystem," Applied Mathematics and Computation, Vol. 162, No. 3, pp. 1119-1134, March 25, 2005.