Modular Arithmetic and Fast Algorithm Designed for Modern Computer Security Applications

Chia-Long Wu*

Professor and Director of Aviation Communication Electronics Department
Chinese Air Force Institute of Technology
chialongwu@gmail.com

Abstract. Modular arithmetic plays very crucial role for public key cryptosystems, such as the public key cryptosystem, the key distribution scheme, and the key exchange scheme. Modular exponentiation is a common operation used by several public-key cryptosystems, such as the RSA encryption scheme and the Diffie-Hellman key exchange scheme. In this paper, we have proposed a new method to fast evaluate modular exponentiation, which combines the complement recoding method and canonical recoding technique.

Keywords: Canonical recoding, modular arithmetic, complexity analyses, fast algorithm design, public-key cryptosystem.

1 Introduction

Modular exponentiation is the fundamental operation in implementing circuits for cryptosystem, as the process of encrypting and decrypting a message requires modular exponentiation which can be decomposed into multiplications. In this paper, a proposed multiplication method utilizes the complement recoding method and canonical recoding technique. By performing complement representation and canonical recoding technique, the number of partial products can be further reduced. Exponentiation is a basic yet important operation for public key cryptography. In this paper, an efficient modular exponentiation method is proposed by adopting the binary method, common-multiplicand multiplication, complement method and signed-digit recoding method. Hamming weight plays an important part for complexity efficiency. On average, by performing minimal Hamming recoding method and signed-digit recoding method, the number of multiplications for our proposed algorithm can be reduced effectively, where k is the bit-length of the exponent E. We can therefore efficiently speed up the overall performance of the modular exponentiation [1].

To compute modular exponentiation $C \equiv M^E \bmod N$, (where C, M, E, and N are ciphertext, plaintext, public key, and modulus respectively) is very time-consuming

* Corresponding author.

J.-S. Pan, S.-M. Chen, N.T. Nguyen (Eds.): ACIIDS 2012, Part III, LNAI 7198, pp. 276–285, 2012.

because the bit-length of E can be up to 2048 bits. Designing efficient algorithms that can speed up software and hardware implementation of modular exponentiation are often considered as practical significance for practical cryptographic applications such as the RSA public-key cryptosystem [1] and the ElGamal cryptosystem [2].

Speeding up modular exponentiation $C \equiv M^E \bmod N$, where $E = \sum_{i=1}^{k} e_i \times 2^i$ and $e_i \in \{0,1\}$, is very crucial for public key cryptosystems (PKC). There are several well-known algorithms for speeding up the exponentiations and the multiplications such as binary exponentiation method (sometimes called the square-and-multiply method) [3], signed-digit recoding method [4-11], exponent-folding-in-half method [12-13], Montgomery reduction method [14-15], common-multiplicand multiplication (CMM) method [16-19], and multi-exponentiation method [20-21], and so on.

The Hamming weight (the number of 1's in the binary representation) plays an important role for the computational efficiency. A novel method for speeding up modular exponentiation by using binary exponentiation method, complement recoding method, and signed-digit recoding method is proposed in this paper. We can efficiently speed up the overall performance of modular exponentiation.

The rest of this paper is organized as follows. Some related methods are introduced in Section 2. In Section 3, the proposed algorithm for fast modular exponentiation is described. Then, the computational complexity of the proposed algorithm is analyzed in Section 4. Finally, we conclude this work and future works in Section 5 [28-30].

2 Mathematical Preliminaries

2.1 The Binary Exponentiation Method

Fast computations of the exponentiation can be classified into two approaches: the faster multiplication designs, and the development of novel exponentiation algorithms. The multiplication involves two basic operations, the generation of partial products and their accumulation. The binary exponentiation method [3] also called square-and-multiply method is a generally acceptable method for exponentiation. It can convert the modular exponentiation of $C \equiv M^E \bmod N$ [23] into a sequence of modular multiplications. Let the exponent E have the binary representation $E = \sum_{i=1}^{k} e_i \times 2^i$, where $e_i \in \{0, 1\}$ and k is the bit- length of the exponent E.

It can be divided into two kinds of methods. One is the right to left binary exponentiation method; the other is the left to right binary exponentiation method. The right to left binary exponentiation method scans the exponent E from the least significant bit (LSB) toward the most significant bit (MSB). It performs one multiplication operation and one square operation when the exponent bit e^i is 1 and performs one square operation when the exponent bit e^i is 0. It will be shown as Algorithm 1[28-30].

Algorithm 1. Right to Left binary exponentiation algorithm

Input: Message: M;
Exponent: $E = (e^k e^{k-1} \ldots e^2 e^1)2$;
Output: Ciphertext: $C = M^E$;
begin
$C = 1$;
$S = M$;
for $i = 1$ to k do /*scan from right to left */
{ if ($e^i = 1$) then C = C×S; /*multiply*/
S = S×S;} /*square*/
endfor
end.

The left to right binary exponentiation method scans the exponent E from the most significant bit (MSB) toward the least significant bit (LSB). It performs one multiplication operation when the exponent bit e^i is 1 and performs one square operation when the exponent bit e^i is 0. It will be shown as Algorithm 2 [25-27].

Algorithm 2. Left to Right binary exponentiation algorithm

Input: Message: M;
Exponent: $E = (e^k e^{k-1} \ldots e^2 e^1)_2$;
Output: Ciphertext: $C = M^E$;
begin
C = 1;
S = M;
for $i = k$ to 1 do /*scan from left to right */
{C = C×C; /*square*/
if ($e^i = 1$) then C = C×S;} /*multiply*/
endfor
end.

The computational complexity of both algorithms expresses as follows. On an average, we assume the occurrence probabilities for both bit "1" and bit "0" are the same i.e. {S×S} and {S×S, C×S} with the same probability. Then, the expectation value for bits "1" and "0" is the same "$\frac{k}{2}$", where k is the bit-length of the exponent E.

2.2 The Bit-Complement Recoding Method

To compute the modular exponentiation of $C \equiv M^E \bmod N$, we express the exponent E as a binary representation $e_k e_{k-1} \ldots e_2 e_1$. Performing complements is advantageous in the speed up of exponential computations [24-26]. The equation and example will be shown as Equation 1.

$$E = \sum_{i=1}^{k} e_i \times 2^i = (e^k e^{k-1} \ldots e^2 e^1)_2 = (10\ldots0)_{(k+1)\text{bits}} - \overline{E} - 1, \quad (1)$$

where $\overline{E} = \overline{e_k}\,\overline{e_{k-1}} \ldots \overline{e_1}$ and $\overline{e_i} = 0$ if $e_i = 1$; $\overline{e_i} = 1$ if $e_i = 0$, for $i = 1, 2, \ldots, k$.

2.3 The 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 [3]. It shows that the average Hamming weight of a k-bit canonically recorded binary number approaches $\frac{k}{3}$ as $k \to \infty$ [4-5, 22]. We should note that a number using the digit $\{\bar{1}, 0, 1\}$ is not uniquely represented in binary signed-digit notation [6]. The equation and example will be shown as Algorithm 3 [25-29].

Algorithm 3. Signed-Digit Recoding Method

Input: E = $(e^k e^{k-1} \ldots e^2 e^1)_2$;
Output: E_{SD}
begin
c1 = 0; rn+2=0; rn+1=0;
for i = 1 to k do
$c_{i+1} = \left\lfloor \frac{c_i + e_i + e_{i+1}}{2} \right\rfloor$;
$e_i = c_i + e_i - 2c_{i+1}$
endfor
return E_{SD}
end.

Since the addition of k bits can generate an integer with magnitude of $\log(k)$ bit addition, the cost needs only $\frac{\log(k)}{k}$ k-bit additions. Here "$A<<<8$" stands for the integer which is obtained by the left-shift eight bits from the multiplicand A. Since the Hamming weight of multiplier B is larger than $\frac{k}{2}$, the Hamming weight of $\overline{B}$ is $(\frac{k}{2} * \frac{1}{2}) = \frac{1}{4}k$ in average, i.e., $A * \overline{B}$ needs $\frac{k}{4}$ k-bit additions. Therefore, we need two $2k$-bit subtractions. Assume that both addition and subtraction have the same computational complexity [25-28].

2.4 The Common-Multiplicand Multiplication Method

In 1993, Yen and Laih proposed the common-multiplicand multiplication (CMM) method to improve the performance of the right-to-left binary exponentiation algorithm for evaluating modular exponentiation "$M^E \bmod N$". Here we concentrate on the computations of $\{A \times B_i | i = 1, 2, \ldots, t;\ t \geq 2\}$. The following variables are required in the CMM method (for $i = 1, 2, \ldots, t$) [25-27],

$$B_{com} = B_1 \textbf{ AND } B_2 \ldots \textbf{AND } B_t, \tag{2}$$

$$B_{i,c} = B_i \textbf{ XOR } B_{com}, \tag{3}$$

where "**AND**" and "**XOR**" are bitwise logical operators.

Hence B_i can be depicted as:

$$B_i = B_{i,c} + B_{com} \quad \text{for } i = 1, 2, \ldots, t. \tag{4}$$

Therefore, the common-multiplicand multiplications $A \times B_i$ (i = 1, 2,…, t) can be computed with the assistance of $A \times B_{com}$ as:

$$A \times B_i = A \times B_{i,c} + A \times B_{com} \quad \text{for } i = 1, 2, \ldots, t. \tag{5}$$

In 1961, Avizienis proposed a signed-digit (SD) representation, also called redundant number representations for parallel and high-speed arithmetic. A signed-digit vector representation of an integer a in radix r is a sequence of digits $a = (a_{k+1}, a_k \ldots, a_2, a_1)_{\text{SD}}$ with $a_i \in \{0, \pm 1, \cdots, \pm r-1\}$ for $k \geq i \geq 0$, i.e., $a = \sum_{i=1}^{k+1} a_i \times r^i$. In a binary signed-digit number (BSD) system, three symbols {$\bar{1}$, 0, 1} are allowed for the digit set, the symbol $\bar{1}$ is used to denote the value -1[28-30].

The basic idea of CMM method is to extract the common parts of multiplicands, and save the number of binary additions for the computation of common parts. Let A and B_is (i = 1, 2) be m-bit integers, the Hamming weights of B_i, B_{com} and $B_{i,c}$ are $m/2$, $m/2^t$ and ($m/2 - m/2^t$), respectively. By using the CMM method, the computations of $\{A \times B_1, A \times B_2\}$ can be represented as $\{A \times B_{1,c} + A \times B_{com}, A \times B_{2,c} + A \times B_{com}\}$.

The total number of binary additions for the common-multiplicand multiplications evaluation is $m/2^t + t \times (m/2 - m/2^t)$. Without the CMM method, the multiplications $\{A \times B_1, A \times B_2\}$ are computed one after another independently using total $t \times (m/2)$ binary addition. Thus, the performance improvement of the common-multiplicand multiplication method shown above can be denoted as [28-30]:

$$\frac{\frac{mt}{2}}{\frac{m}{2^t} + t \times (\frac{m}{2} - \frac{m}{2^t})} = \frac{t}{t + (1-t) \times 2^{1-t}}. \tag{6}$$

The auxiliary carry C_0 is set to 0 and subsequently the binary number A is scanned two bits at a time. The canonically recoded digit Bi and the next value of the auxiliary binary variable C_{i+1} for i = 0, 1, 2, … , n are generated as shown in Table 1.

Table 1. Canonical recoding table

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

3 The Proposed Signed-Digit Recoding Algorithm

In Section 2, we describe the binary exponentiation method, complement recoding method, and signed-digit recoding method respectively. We combine these methods as Algorithm 5 [28-30] to accelerate the exponentiation. Algorithm 4 is the signed-digit recoding method and is depicted as follows).

Algorithm 4. Signed-Digit Recoding Algorithm

Input: Message: M;
Exponent: E = $(e^k e^{k-1} \ldots e^2 e^1)_2$;
Output: Ciphertext: $C = M^E$;
begin
C = 1;
S = M;
for i = 1 to k do /*scan from right to left */
{if $(e_i = 1)$ then $C \equiv (S \times C) \bmod N$; /*multiply*/
if $(e_i = \bar{1})$ then $C \equiv (S^{-1} \times C) \bmod N$;
$S \equiv (S \times S) \bmod N$. /*square*/
end.

Algorithm 5. The Proposed Signed-Digit Recoding Algorithm

Input: Message: M;
Exponent: $E = (e^k e^{k-1} \ldots e^2 e^1)_2$;
Modulus: N;
Output: Cipher-text: $C \equiv M^E \bmod N$
begin
Count the Hamming weight of E, denote as Ham(E).
if Ham(E)> $\frac{k}{2}$
Perform the complement recoding and the signed-digit recoding procedures.
C = 1;
S = M^{-1};
for i=1 to k do
{if $(e_i = 1)$ then $C \equiv (S \times C) \bmod N$;
if $(e_i = \bar{1})$ then $C \equiv (S^{-1} \times C) \bmod N$;
$S \equiv (S \times S) \bmod N$; }
else
Perform the signed-digit of E is E_{SD}.
Call Signed-Digit Binary algorithm (M, E, N): C;
Output C;
end.

4 The Complexity Analyses of the Proposed Algorithm

In this section, we will describe the computational complexity of the proposed algorithm. The computational complexity of the proposed method is $\frac{1}{3}k+2*\frac{\log(k)}{k}+5\approx 0.333k$ k-bit additions that are faster than $\frac{3}{4}k\approx 0.75k$ in Yen-Laih method, $\frac{23}{32}k\approx 0.719k$ in Wu-Chang method, $\frac{7}{12}k\approx 0.583k$ in Yen's method and $\frac{1}{2}k+2*\frac{\log(k)}{k}+5\approx 0.5k$ in Chang- Kuo-Lin method. Here are the complete complexity analyses.

We assume there are k bits in exponent E. There are two cases [28-30]:

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

The computational complexity of $C\equiv M^E \bmod N$ = (the computational complexity of Step 1)+ ($\frac{1}{2}\times$ the computational complexity of Step 2) + ($\frac{1}{2}\times$ the computational complexity of Step 3).

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

Assume that the multiplicand A and multiplier B are k-bit unsigned binary numbers. The computational complexity of $P=A*B$ is defined as " (the computational complexity of Step 1) + ($\frac{1}{2}\times$the computational complexity of Step 2) + ($\frac{1}{2}\times$the computational complexity of Step 3)". The second and the third items "$\frac{1}{2}$" in the above equation mean the probabilities of Ham(B) > $\frac{k}{2}$ and Ham(B) $\leq\frac{k}{2}$.

Now we describe the computational complexity of Step 1, Step 2(Case 1) and Step 3 (Case 2) respectively. First, we define E_{SD} a binary signed-digit representation for E and $\overline{E_{SD}}$ a binary signed-digit representation for $\overline{E}$ respectively.

Then, the computational complexity is counted on the number of k-bit multiplication [28-30].

Step 1: scan E from LSB to MSB

We scan E from the least significant bit (LSB) toward the most significant bit (MSB) to sum them up and check if Ham(E)>$\frac{k}{2}$. The computational complexity of this step is much less than that of multiplication [28-30].

Step 2: Ham(E)>$\frac{k}{2}$

We consider 1's complement of E as $\overline{E}$, i.e. Ham $(\overline{E})<\frac{k}{2}$. We can replace $C\equiv M^E \bmod N$ by $C\equiv M^{(10...0)_{(k+1)\text{bits}}-\overline{E}-1} \bmod N\equiv M^{(10...0)_{(k+1)\text{bits}}-\overline{E_{SD}}-1} \bmod N$. On an

average, the Hamming weight of $\overline{E_{\mathrm{SD}}}$ is $\frac{k}{2}\times\frac{1}{3}=\frac{k}{6}$, where "$\frac{1}{3}$" is non-zero digit probability for $\overline{E_{\mathrm{SD}}}$ by using signed-digit recoding method [28-30].

$$\begin{aligned} C &\equiv M^{E} \bmod N \equiv M^{(1111100001)_2} \bmod N \\ &\equiv M^{(10..0)_{(k+1)\text{bits}}-\bar{E}-1} \bmod N \equiv M^{(10000000000)_{11\text{bits}}-0000011110-1} \bmod N \\ &\equiv M^{(10..0)_{(k+1)\text{bits}}-\overline{E_{\mathrm{SD}}}-1} \bmod N \equiv M^{(10000000000)_{11\text{bits}}-000010000\bar{1}0-1} \bmod N \\ &\equiv (M^{(10000000000)_{11\text{bits}}} \times (M^{-1})^{000010000\bar{1}0} \times M^{-1}) \bmod N \end{aligned} \tag{7}$$

where k is the bit-length of the exponent and $E=(1111100001)_2$.

5 Conclusions

In this paper, we have proposed a fast method to efficiently evaluate modular multiplication, which combines the complement recoding method and canonical recoding technique [29-30]. The computational complexity of the proposed method is faster than Yen-Laih method, Wu-Chang method [12], Yen's method [17] and in Chang- Kuo-Lin method [24-28]. We can efficiently speed up the overall performance of multiplication operation by using the proposed algorithm.

As the modular squaring operation in finite field can be done by a simple shift operation when a normal basis is used, and the modular multiplications and modular squaring operations in our proposed signed-digit recoding scheme can be executed in parallel, by using our proposed generalized r-radix signed-digit folding algorithm, hardware design and parallel technique, we can effectively decrease the computational complexity [27-30].

References

1. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
2. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
3. Knuth, D.E.: The Art of Computer Programming, 3rd edn. Seminumerical Algorithms, vol. 2. Addison-Wesley, MA (1997)
4. Yang, W.C., Guan, D.J., Laih, C.S.: Algorithm of asynchronous binary signed-digit recording on fast multi-exponentiation. Applied Mathematics and Computation 167(1), 108–117 (2005)
5. Koc, C.K., Johnson, S.: Multiplication of signed-digit numbers. Electronics Letters 30(11), 840–841 (1994)
6. Avizienis: Signed-digit number representations for fast parallel arithmetic. IRE Transactions on Electronic Computers 10, 389–400 (1961)
7. Arno, S., Wheeler, F.S.: Signed digit representations of minimal Hamming weight. IEEE Transactions on Computers 42(8), 1007–1010 (1993)

8. Syuto, M., Satake, E., Tanno, K., Ishizuka, O.: A high-speed binary to residue converter using a signed-digit number representation. IEICE Transaction on Information and Systems E85-D(5), 903–905 (2002)
9. Heuberger, C., Prodinger, H.: Carry propagation in signed digit representations. European Journal of Combinatorics 24(3), 293–320 (2003)
10. Joye, M., Yen, S.M.: Optimal left-to-right binary signed-digit recoding. IEEE Transactions on Computers 49(7), 740–748 (2000)
11. Koren: Computer Arithmetic Algorithms, 2nd edn. A. K. Peters, MA (2002)
12. Lou, D.C., Chang, C.C.: Fast exponentiation method obtained by folding the exponent in half. Electronics Letters 32(11), 984–985 (1996)
13. Lou, D.C., Wu, C.L., Chen, C.Y.: Fast exponentiation by folding the signed-digit exponent in half. International Journal of Computer Mathematics 80(10), 1251–1259 (2003)
14. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation 44(170), 519–521 (1985)
15. Tenca, F., Koc, C.K.: A scalable architecture for modular multiplication based on Montgomery's algorithm. IEEE Transactions on Computers 52(9), 1215–1221 (2003)
16. Yen, S.M., Laih, C.S.: Common-multiplicand-multiplication and its applications to public key cryptography. Electronics Letters 29(17), 1583–1584 (1993)
17. Yen, S.M.: Improved common-multiplicand-multiplication and fast exponentiation by exponent decomposition. IEICE Transaction on Fundamentals E80-A(6), 1160–1163 (1997)
18. Wu, T.C., Chang, Y.S.: Improved generalization common-multiplicand-multiplications algorithm of Yen and Laih. Electronics Letters 31(20), 1738–1739 (1995)
19. Ha, C., Moon, S.J.: A common-multiplicand method to the Montgomery algorithm for speeding up exponentiation. Information Processing Letters 66(2), 105–107 (1998)
20. Dimitrov, V.S., Jullien, G.A., Miller, W.C.: Complexity and fast algorithms for multi-exponentiations. IEEE Transactions on Computers 49(2), 141–147 (2000)
21. Chang, C.C., Lou, D.C.: Parallel computation of multi-exponentiation for cryptosystems. International Journal of Computer Mathematics 63(1-2), 9–26 (1997)
22. Wu, C.-L., Lou, D.-C., Lai, J.-C., Chang, T.-J.: Fast modular multi-exponentiation using modified complex arithmetic. Applied Mathematics and Computation 186(2), 1065–1074 (2007)
23. Stallings, W.: Cryptography and Network Security Principles and Practice, 3rd edn. Prentice-Hall, NY (2002)
24. Chang, C.C., Kuo, Y.T., Lin, C.H.: Fast algorithms for common multiplicand multiplication and exponentiation by performing complements. In: Proceeding of 17th International Conference on Advanced Information Networking and Applications, pp. 807–811 (March 2003)
25. Wu, C.L.: Fast modular multiplication based on complement representation and canonical recoding. In: The 7th Conference of Crisis Management (CMST 2009), Tainan, Taiwan, pp. 1–8 (November 27, 2009)
26. Wu, C.L.: Modular exponentiation arithmetic and number theory for modern cryptographic security applications. In: 8th Conference of Crisis Management (CMST 2010), CCM1010002IFS, Kaohsiung, Taiwan, pp. 169–176 (2010)
27. Wu, C.L.: High performance of modular arithmetic and theoretical complexity analyses. In: Proceedings of the 7th Pacific Symposium on Flow Visualization and Image Processing (PSFVIP-7), pp. 18–35 (November 2009)

28. Wu, C.L., Lou, D.C., Chang, T.-J., Chen, C.-Y.: Fast modular exponentiation algorithm theoretical design and numerical analysis for modern cryptographic applications. In: 17th National Defense Science Technology Symposium (ND17), Taoyuan, Taiwan, November 27-28, vol. 5-1–5-7 (2008)
29. Wu, C.-L.: Complexity analyses and design for cryptographic modular algorithm. In: 2011 Symposium on Communication Information Technology on Management and Application, Paper No. 0505, Kaohsiung, A2: Communication Theory, pp. 1–6 (2011)
30. Wu, C.-L.: Fast Montgomery binary algorithm for information security. In: 2011 International Symposium on NCWIA, Paper No. 111, Kaohsiung, D6: Information Systems and Innovative Computing, pp. 1–5 (2011)