Electronics and Communications in Japan, Part 3, Vol. 83, No. 12, 2000
Translated from Denshi Joho Tsushin Gakkai Ronbunshi, Vol. J81-A, No. 10, October 1998, pp. 1372–1376

A New Fast Modular Multiplication Method and Its Application to Modular Exponentiation-Based Cryptography

Akira Hayashi

Kanazawa Institute of Technology, Ishikawa, Japan 921-8501

SUMMARY

In order to apply exponentiation-based cryptography, such as RSA cryptography and El Gamal cryptography, to a wide range of practical problems, it is desired to devise faster ciphering and deciphering processes. This paper proposes a new algorithm for improving the speed of the exponentiation-based computation. The proposed method is based on the idea in the exponentiation computation that the remainder in square/multiplication with modulus n is constructed from the remainders with moduli different from n. More precisely, the method is based on the following two ideas. (1) The remainder in regard to n can be constructed from the remainder with modulus $n + 1$ and the remainder with modulus $n + 2$. (2) It often happens that $n + 1$ and $n + 2$ can easily be factorized, even if n is a prime number or difficult to be factorized into prime factors. Then, the Chinese remainder theorem can be applied to the remainder calculation with those numbers as the moduli. The bit computational complexity of the proposed method is estimated, and it is shown, assuming the parallel computation, that the computational complexity is less than in the conventional method. Especially when $n + 1$ and $n + 2$ are factorized almost uniformly into K factors, the computational complexity asymptotically follows $1/K$. The proposed method will be useful not only in the cryptography requiring the exponentiation computation, but also in improving the speed of the signal processing that requires similar computations. © 2000 Scripta Technica, Electron Comm Jpn Pt 3, 83(12): 88–93, 2000

Key words: Multiplication remainder computation; exponentiation computation; public key cryptography; RSA cryptography; El Gamar cryptography.

1. Introduction

There exist many cryptographic methods, including RSA cryptography [1], that are based on the exponentiation computation $x^e \bmod n$. Other examples are Rabin cryptography [2], El Gamar cryptography [3], elliptic curve cryptography [4–6], digital signature standard [7], Diffie–Hellman key exchange system [8], and Okamoto authentification system [9]. Of those, RSA cryptography is one of the powerful public key cryptographies, but faster ciphering and deciphering procedures are desired in order to apply the method widely to practical problems. The reason for the relatively low speed of the RSA and other cryptographies is the low speed of the exponentiation computation for a large number such as 200 to 700 digits.

In the exponent computation x^e, the number of multiplications can be reduced greatly by combining the square and multiplication operations, not by multiplying successively x. There is also an elaboration, such as the addition chain, to reduce the number of multiplications [10]. As another practically powerful approach, there is a method that replaces a part of the computation by the table search, by utilizing the preliminary computation. There is also a method, as in the deciphering process in RSA cryptography, that can be applied when the prime factorization of the modulus is known [11]. For details of those methods, see Refs. 10, 12 and 13.

The method proposed in this paper is based on an idea that is different from those methods, and can be combined with the latter. The proposed method is based on the idea that the remainder for modulus n is constructed from the remainders with moduli different from n. In the already known method based on the Chinese remainder theorem, more than one different modulus is used. The method proposed in this paper differs in that the remainder with modulus n is constructed from the remainder with modulus $n + 1$ and the remainder with modulus $n + 2$.

It may be sensible to consider what the advantage is in using different moduli. In RSA cryptography, the modulus n is a composite number, but its prime factorization is not known except for the decipherer. In El Gamar cryptography, the modulus is a prime number. If $n + 1$ and $n + 2$ can easily be prime factorized, the remainder operation can be sped up by applying the Chinese remainder theorem. In fact, n is set so that it is difficult to be prime factorized in RSA cryptography, but such a consideration is not applied to $n + 1$ and $n + 2$. More positively, it may be sensible to set n so that $n + 1$ and $n + 2$ can easily be prime factorized, although the prime factorization of n is difficult.

In short, the method proposed in this paper is to handle the case where the modulus n is difficult or impossible to be factorized, by determining the remainder efficiently for different moduli which can easily be factorized, and then deriving the remainder for modulus n from that result.

Section 2 presents some definitions for the notations and describes the Chinese remainder theorem. The exponentiation computation is briefly discussed. Section 3 presents the remainder theorem that gives the basis for the new remainder computation proposed by the authors, together with numerical examples. Section 4 describes the square computation based on the remainder computation derived in Section 3. This serves as the basis for the exponentiation computation. Section 5 gives the estimation of the computational complexity of the new algorithm, and it is shown that the procedure is much faster than the conventional algorithm. Lastly, the problems left for the future are briefly discussed.

2. Remainder Computation Algorithm

2.1. Preliminaries

When an integer a is divided by a positive integer n, the remainder $r, 0 \le r < n$ is given by $r = a \bmod n$. It is called the remainder with modulus n. When $a \bmod n = b \bmod n$, it is said that a and b are congruent with modulus n. It is written that $a \equiv b \pmod{n}$.

The Chinese remainder theorem [10], which is used later, is described in the following without proof.

[Theorem 1] Let h mutually prime moduli $m_1, \ldots, m_h$ and integers $a_1, \ldots, a_h$ be given. Then, there exists the unique x in the range $0 \le x < M = m_1 \ldots m_h$, satisfying the system of congruence equations

$$x \equiv a_i \pmod{m_i}, \quad i = 1, \ldots, h \tag{1}$$

It is given by

$$x = \sum_{i=1}^{h} a_i x_i \bmod M \tag{2}$$

where

$$x_i = M_i m_i', \quad M_i = M/m_i$$
$$m_i' M_i \equiv 1 \pmod{m_i}$$

The computation that derives the solution x for the above system of congruence equations (1), is written as $\mathbf{CR}(\boldsymbol{a}, \boldsymbol{m}, x)$, where $\boldsymbol{a} = (a_1, \ldots, a_h)$ and $\boldsymbol{m} = (m_1, \ldots, m_h)$.

2.2. Exponentiation remainder computation

The exponentiation computation is to calculate $x^e \bmod n$. It is possible simply to calculate x^e and determine the remainder by dividing the result by n. By this method, however, the computation steps are tremendous, and the digits reach e times the digits of x, at the maximum. In order to avoid this, the increase of the computation steps may be suppressed by dividing the result each time by n to determine the remainder. In this simple method, the number of multiplications is $e - 1$, whereas it can be reduced to approximately $2 \log_2 e$ by the square/multiply method [10].

3. New Remainder Algorithm

This section gives a new remainder algorithm. Let

$$y = X \bmod n \tag{3}$$

where

$$0 \leqq X \leqq (n-1)^2 \tag{4}$$

The reason for imposing condition (4) is that computations such as $X = x^2 \bmod n$ and $xu \bmod n$ must be considered for $0 \le x$ and $u < n$. In the following, it is set that

$$y_1 = X \bmod (n+1) \tag{5}$$
$$y_2 = X \bmod (n+2) \tag{6}$$

X is represented as follows in terms of y_1 and y_2.

[Lemma 1] Under condition (4), as well as Eqs. (5) and (6), the following relation applies:

$$X = (n+2)y_1 - (n+1)y_2 + k(n+1)(n+2) \tag{7}$$

where k is either 0 or 1.

(Proof) Since $n+2 \equiv 1 \pmod{n+1}$ and $n+1 \equiv -1 \pmod{n+2}$, there hold $X \equiv y_1 \pmod{n+1}$ and $X \equiv y_2 \pmod{n+2}$. Thus, Eqs. (5) and (6) are valid. It is then shown that k is either 0 or 1. Assume first that $k < 0$. Since $y_1 \le n$ and $y_2 \ge 0$, it follows that the right-hand side of Eq. (7) $\le (n+2)y_1 - (n+1)y_2 - (n+1)(n+2) \le (n+2)n - (n+1)(n+2) = -(n+2) < 0$. Since $X \ge 0$, on the other hand, k cannot be negative.

Then, assume that $k > 1$. Since $y_1 \ge 0$ and $y_2 \le n+1$, the right-hand side of Eq. (7) $\ge -(n+1)^2 + 2(n+1)(n+2) = (n+1)(n+3) > (n-1)^2$. Since, on the other hand, $X \le (n-1)^2$, k cannot be larger than 1. Thus, it is shown that $k = 0$ or 1, and the theorem is shown. □

Using Lemma 1, the following theorem is obtained, which relates y, y_1, and y_2.

[Theorem 2] Under the same condition as in Theorem 1, the following relations apply:

$$y_1 \ge y_2 \qquad y \equiv 2y_1 - y_2 \pmod{n} \tag{8}$$

$$y_1 < y_2 \qquad y \equiv 2y_1 - y_2 + 2 \pmod{n} \tag{9}$$

(Proof) It follows from Eq. (7) that $X \equiv 2y_1 - y_2 + 2k \pmod{n}$. Equation (7) is rewritten as

$$X = (n+1)(y_1 - y_2) + y_1 + k(n+1)(n+2)$$

Then, the following properties are derived from the proof of Lemma 1. If $y_1 \ge y_2$, Eq. (8) is obtained since $k = 0$. If $y_1 < y_2$, Eq. (9) is obtained since $k = 1$. Thus, the theorem is shown. □

The following situation should be noted. In order to determine y using Eqs. (8) and (9), mod n value of the right-hand side must be calculated. For this, however, the division is not required. The reason is as follows, as is easily seen. The right-hand side is larger than $-n$ and less than $2n$. Consequently, it suffices to add or subtract n at most once.

[Example 1] Let $n = 19$ and $X = x^2$.

(a) When $x = 11$, $y_1 = 121 \bmod 20 = 1$ and $y_2 = 121 \bmod 21 = 16$. In this case, $y_1 < y_2$. Applying Eq. (9), $y \equiv 2 - 16 + 2 = -12 \equiv 7 \pmod{19}$ is obtained. This agrees with the result of direct calculation 121 mod 19.

(b) When $x = 13$, $y_1 = 169 \bmod 20 = 9$ and $y_2 = 169 \bmod 21 = 1$. In this case, $y_1 > y_2$. Applying Eq. (8), $y \equiv 18 - 1 \equiv 17 \pmod{19}$ is obtained. This agrees with the result of direct calculations169 mod 19. □

[Example 2] Consider the RSA cryptography example in the original paper [1]. Let $n = 2773$ and $X = x^5$. Then, $n + 1 = 2774 = 2 \times 19 \times 73$ and $n + 2 = 2775 = 3 \times 5^5 \times 37$. When $x = 920$, $y_1 = 920^2 \bmod 2774 = 84640 \bmod 2774 = 330$ and $y_2 = 920^2 \bmod 2775 = 25$. In this case, $y_1 > y_2$. Applying Eq. (8), $y \equiv 2 \times 330 - 25 = 660 - 25 \equiv 635 \pmod{2773}$ is obtained. This agrees with the result of direct calculation $920^2 \bmod 2773 = 635$.

As is seen from the above examples 1 and 2, two computations of mod $(n + 1)$ and mod $(n + 2)$ are used, instead of the single computation of mod n. It may seem that the computation is made more complex, but the method has an advantage as is shown in the next section.

4. New Remainder Multiplication Algorithm

The exponentiation computation can be composed of the remainder square computation and the remainder multiplication computation:

$$x^2 \bmod n, \qquad xu \bmod n \tag{10}$$

It was shown in the previous section that $y = X \bmod n$ can be derived from $y_1 = X \bmod (n+1)$ and $y_2 = X \bmod (n+2)$. In the following, the calculation method **sqmod** for y_1 and y_2 is shown, which is needed in applying the algorithm based on Theorem 2 to the calculation of Eq. (10) in the exponentiation computation. In this algorithm, the Chinese remainder algorithm is used to derive y_1 and y_2. This helps to improve the speed, but the computational complexity is discussed in the next section.

[Preliminary computation for algorithm **sqmod**]

As the preliminary computation, $n + 1$ and $n + 2$ are decomposed into products of mutually prime factors. This need not be the prime factorization.

$$n+1 = \Pi_{i=1}^{h} p_i$$

$$n+2 = \Pi_{i=1}^{m} q_i$$

Assuming that moduli $n + 1$ and $n + 2$ are decomposed as above, the next algorithm receives x such that $0 \le x \le n - 1$, and outputs $y = x^2 \bmod (p_1, \ldots, p_h)$.

[Algorithm **sqmod**$(x, \boldsymbol{p}, y)$
Input: x, $0 \le x \le n-1$, $\boldsymbol{p} = (p_1, \ldots, p_h)$
Output: $y = x^2 \bmod (p_1, \ldots, p_h)$
1. Calculate $x_i = x \bmod p_i$, $i = 1, \ldots, h$.
2. Calculate $a_i = x_i^2$, $i = 1, \ldots, h$.
3. Calculate $a_i = a_i \bmod p_i$, $i = 1, \ldots, h$.
4. Calculate y by **CR**$(\boldsymbol{a}, \boldsymbol{p}, y)$. □

The algorithm **sqmod** is used. $y_1 = x^2 \bmod (n+1)$ and $y_2 = x^2 \bmod (n+2)$ are obtained by **sqmod**$(x, \boldsymbol{p}, y_1)$ and **sqmod**$(x, \boldsymbol{q}, y_2)$, respectively. In the above, the remainder of x^2 is calculated. The remainder of xu is similarly calculated. It suffices to add $u_i = u \bmod p_i$ in stage 1 and use $a_i = x_i u_i$ instead of $a_i = x_i^2$ in stage 2.

[Example 3] Consider the same RSA cryptography as in example 2. Let $n = 2773$. As preliminary computations, $n + 1$ and $n + 2$ are decomposed:

$$n + 1 = 2774 = 2 \times 19 \times 73$$
$$n + 2 = 2775 = 3 \times 5^2 \times 37$$

Let $p_1 = 38$, $p_2 = 73$, $q_1 = 75$, and $q_2 = 37$. As the exponentiation computation, assume that $x = 920$ and $y = x^2 \bmod n$ is to be calculated. The computation procedure for **sqmod**(920, (38, 73), y_1) is shown in the following:

1. $x_1 = 920 \bmod 38 = 8$
   $x_2 = 920 \bmod 73 = 44$
2. $a_1 = 8^2 = 64$
   $a_2 = 44^2 = 1936$
3. $a_1 = 64 \bmod 38 = 26$
   $a_2 = 1936 \bmod 73 = 38$

4. Solving the following system of congruence equations, $y_1 = 330$ is obtained:

$$y_1 \equiv 26 \pmod{38}$$
$$y_1 \equiv 38 \pmod{73}$$

Similarly, $y_2 = 25$ is obtained from **sqmod**(920, (75, 37), y_2). From $y_1 = 330$ and $y_2 = 25$, $y = 2 \times 330 - 25 = 635$ is obtained by Eq. (8) of Theorem 2, as was already seen in Example 2. □

5. Computational Complexity

In this section, the computational complexity is compared between the ordinary direct method and the proposed method for the remainder square/multiplication computation. The additions and subtractions are not counted, and the computational complexity is considered only for the multiplications and divisions. It is assumed that parallel computation is not used in each multiplication or division, and ordinary straightforward computation is applied. As in Ref. 14, the standard bit computational complexity is considered. Then, the computational complexities for the multiplication and the division are given, respectively, as follows:

$M(k, l)$ = computational complexity for $k \times l$ bit number
$= l(k + l)$

$D(k, l)$ = computational complexity for $k \div l$ bit number
$= l(k - l)$

In the method proposed in this paper, the individual multiplication and division are separated. In this case, however, those partial computations can be executed in parallel. This is the same method of comparison as in Quisquater's method [11], known as the fast deciphering algorithm of RSA cryptography, which is evaluated as 8 times faster than the conventional method. The computational complexity in the preliminary computation is not included. In the following, n is assumed to consist of k bits, and $p_1, \ldots, p_h$ and $q_1, \ldots, q_m$ of l bits at the maximum.

[Ordinary direct method]

In the ordinary calculation of $x^2 \bmod n$, the multiplication of two k-bit numbers and the division of a $2k$-bit number by a k-bit number are required. The computational complexity for this is $T_0 = M(k, k) + D(2k, k) = 3k^2$.

[Proposed method]

The computational complexity of the **sqmod** algorithm is examined as follows. In order to calculate the right-hand side of the system of congruence equations, the division of k-bit number by l-bit number is required in stage 1, the multiplication of two l-bit numbers is required in stage 2, and the division of $2l$-bit number by l-bit number is required in stage 3. In total, the computational complexity is $D(k, l) + M(l, l) + D(2l, l) = l(k - l) + 2l^2 + l^2 = l(k + 2l)$, which is needed for each i.

In solving the system of congruence equations, the multiplication of l-bit number and k-bit number, as well as the division of $(k + l)$-bit number by k-bit number, are required. Then, the total is $M(k, l) + D(k + l, k) = l(2k + l)$. Thus, the computational complexity of the proposed method is $T = 3l(k + l)$.

Especially when $l = k/2$ as in the previous numerical example, $T = 9/4k^2$, which is 3/4 of the direct method. When $l = k/K$ in general, $T = \frac{K+1}{K^2} T_0$. When $K = 4$, for example, the computational complexity can be reduced to approximately 30%. When K is further increased, the computational complexity asymptotically approaches $1/K$. In the exponentiation computation $x^e \bmod n$, the numbers of the square operations and multiplications are, respectively, the same in the direct method and the proposed method, and the ratio of the computational complexities of the two methods is also T/T_0.

6. Conclusions

When modulus n is a prime number or a composite number for which prime factorization is difficult, the speed of the exponentiation computation cannot be improved by utilizing the Chinese remainder theorem. In this paper, the property is noted that the prime factorization of $n + 1$ and $n + 2$ is not always difficult, and it is shown that the

remainder for mod n can be determined from the remainders y_1 and y_2 with those as the modulus. It is shown that the Chinese remainder theorem can be applied to the calculation of y_1 and y_2, and the remainder multiplication can be realized with less computational complexity. It is shown that the proposed computation method is useful in improving the speed of the exponentiation-based cryptography, such as RSA cryptography and El Gamar cryptography.

The effectiveness of the proposed method is examined as follows. The effectiveness of the method depends strongly on whether or not $n + 1$ and $n + 2$ can be decomposed into smaller prime factors. When an integer x not greater than a is selected at random, the maximum prime factor is less than $a^{0.6}$ with a probability of approximately 50% [10]. In other words, when a is a k-bit number, it is expected with a probability of 1/2 that the maximum prime factor of an integer not greater than a is of at most $0.6k$ bit. If n is set so that it is decomposed into three or more prime factors, the computational complexity will further be decreased. When n is the public key of RSA cryptography, if $n + 1$ and $n + 2$ are publicized in the factorized forms, the preliminary computation in ciphering is made unnecessary.

In this paper, the remainders for $n + 1$ and $n + 2$ are used, but it will be possible to use the remainder of $n + a$ as a more general approach. In the evaluation of the computational complexity, the bit complexity of the algorithm is simply used as the measure, but a more practical measure for the computational complexity may be considered. The author hopes that the method proposed herein will effectively be used to improve the speed of the signal processing, where the cryptography accompanying the exponentiation computation or similar computations is being used alone or combined with other methods.

Acknowledgments. Useful advice was provided by Professors E. Okamoto, T. Matsumoto, K. Kuroiwa, T. Matsumoto, and N. Yanagawa. Discussions with Mr. H. Shimizu and A. Shinho were very useful.

REFERENCES

1. Rivest R, Shamir A, Adleman L. A method for obtaining digital signatures and public key cryptosystems. Commun ACM 1978;21:120–126.
2. Rabin MO. Digital signatures and public-key functions as intractable as factorization. MIT/LCS/TR-212 Tech. memo, Massachusetts Institute of Technology, 1979.
3. El Gamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans Inf Theory 1985;31:469–472.
4. Koyama K, Maurer U, Okamoto T, Vanstone SA. New public-key schemes based on elliptic curves over the ring Z_n. Proc CRYPTO'91, Santa Barbara, p 252–266.
5. Menezes AJ, Vanstone SA. Elliptic curve cryptosystems and their implementation. J Cryptol 1993;6:209–224.
6. Kuwakado K, Koyama K. Efficient cryptosystems over elliptic curves based on a form-free primes. IEICE Trans Fundam 1994;E77-A:1309–1318.
7. NIST. Specifications for a digital signature standard. Federal Information Processing Standard Publication 186, 1991.
8. Diffie W, Hellman M. New directions in cryptography. IEEE Trans Inf Theory 1976;22:644–654.
9. Okamoto T. Provably secure and practical identification schemes and corresponding signature schemes. Proc CRYPTO'92, Santa Barbara, p 31–53.
10. Knuth DE. The art of computer programming. Vol. 2, 2nd ed. Addison-Wesley; 1980. Chapter 4.
11. Quisquater JJ, Couvreur C. Fast decipherment algorithm for RSA public-key cryptosystem. Electron Lett 1982;18:905–907.
12. Ito T, Sako K. Algorithm on finite field and high-speed algorithm for multiple length remainder operation. Inf Process 1993;34:170–179.
13. Morita H. Cryptography and high-speed algorithm. Inf Process 1993;34:336–342.
14. Koblitz N. A course in number theory and cryptography. Springer; 1987. Chapter 1.

AUTHOR

Akira Hayashi graduated from the Department of Electrical Engineering of Kanazawa University in 1964. He completed his master's program at the University of Minnesota in 1973 and his doctoral program at the University of Hawaii in 1976. He joined Toshiba Corp. Central Res. Lab. in 1964. He became a lecturer in 1970 and has been a professor in the Department of Information Engineering, Kanazawa Institute of Technology, since 1977. He holds a Ph.D. degree. His research interests are acoustic engineering, communication theory, information theory, theory of cryptography, and neurocomputing. He is a member of the Information Processing Society, IEEE, and the Information Theory and Application Society.