COMMUN. MATH. SCI.
Vol. 5, No. 2, pp. 383–389


# AN EFFICIENT MODULO $P$ MULTIPLICATION ALGORITHM WITH MODERATE FACTORS OF $P+1$ AND $P-1$*

REN-JUNN HWANG†, FENG-FU SU‡, AND SHENG-HUA SHIAU§

**Abstract.** Modular multiplication plays an important role to several public-key cryptosystems such as the RSA cryptosystem. This paper proposes an efficient modulo $p$ multiplication algorithm with moderate factors of $p+1$ and $p-1$. In order to improve the RSA decryption performance, users can utilize our proposed algorithm and the strong prime criterion. It will prove that the decryption method based on our proposed algorithm can run at a speed almost 6.5 times faster than that of the traditional method, or almost 2 times faster than that of the method based on the Chinese Remainder Theorem. Furthermore, the proposed algorithm can greatly enhance the performance of RSA encryption.

**Key words.** modular multiplication, modular exponentiation, RSA cryptosystem, strong prime

**AMS subject classifications.** 65Y20, 68Q99

## 1. Introduction

Modular multiplication is fundamental to several public-key cryptosystems such as the RSA cryptosystem [1]. The RSA cryptosystem uses modular exponentiation performed by modular multiplication repeated enough times to ensure security. This is computationally expensive. So we need efficient modular multiplication for a large modulus.

Hayashi [2] proposed a new modular multiplication method to reduce the computational time of RSA. In his method, the modular exponentiation with modulus $n$ transforms into two substitute operations with moduli $n+1$ and $n+2$. If moduli $n+1$ and $n+2$ are factored, the user can then apply the Chinese Remainder Theorem(CRT) modulo $n+1$ and $n+2$, respectively. Hayashi's formula can generate the final result, but his method is not practical; as $n+1$ and $n+2$ grow larger and larger, it becomes expensive to factorize $n+1$ and $n+2$.

In this paper, we propose an efficient modulo $p$ multiplication algorithm with moderate factors of $p+1$ and $p-1$ inspired by Hayashi's method [2]. The proposed algorithm transforms a modular operation with the modulus $p$ into two substitute operations with decomposable moduli $p+1$ and $p-1$. The idea is that fast CRT evaluation may work for moduli $p+1$ and $p-1$, even if it does not work for modulus $p$. The algorithm can greatly enhance the performance of RSA decryption and reduce the computational time of RSA encryption. Based on the strong prime [3, 4, 5, 6, 7, 8] of RSA criterion, users can employ the proposed algorithm to enhance the performance of RSA decryption.

The rest of this article is organized as follows: section 2 introduces our efficient modular multiplication algorithm; section 3 introduces the application to the RSA cryptosystem; section 4 analyzes the computational complexity before we make a conclusion in section 5.

*Received: December 12, 2006; accepted: April 2, 2007. Communicated by Anna Gilbert.

†Department of Computer Science and Information Engineering, TamKang University, Tamsui, Taipei 251, Taiwan (victor@mail.tku.edu.tw).

‡Department of Computer Science and Information Engineering, TamKang University, Tamsui, Taipei 251, Taiwan (fengfusue@yahoo.com.tw).

§Department of Computer Science and Information Engineering, TamKang University, Tamsui, Taipei 251, Taiwan (891190067@s91.tku.edu.tw).

## 2. An efficient modular multiplication algorithm

In this section, we present an efficient modulo $p$ multiplication algorithm with moderate factors of $p+1$ and $p-1$. Let $a$, $b$, and $p$ be three $n$-bit positive binary integers where $a,b<p$ and $\gcd(2,p) = 1$. Assume $p-1$ and $p+1$ can be decomposed into products of mutually prime factors, that is to say, $p-1=u_1\times u_2\times\ldots\times u_f$, where $\gcd(u_i,u_j) = 1$ for $1\le i<j\le f$, and $p+1=v_1\times v_2\times\ldots\times v_g$, where $\gcd(v_i,v_j) = 1$ for $1\le i<j\le g$. The modular multiplication algorithm is the following algorithm.

**An efficient modular multiplication algorithm**

**Input:** $a,b,p,(u_1,u_2,\ldots,u_f),(v_1,v_2,\ldots,v_g)$
**Output:** $s=a\times b \bmod p$
**Step 1.** Compute $D=(p^2-1)/2$.
**Step 2.** Compute $M=a\times b$.
**Step 3.** If $M\ge D$, then $z=1$, else $z=0$.
**Step 4.** Compute $K_j=M \bmod u_j$, $j=1,\ldots,f$.
**Step 5.** Employ the CRT to compute $Y_1$.
**Step 6.** Compute $J_k=M \bmod v_k$, $k=1,\ldots,g$.
**Step 7.** Employ the CRT to compute $Y_2$.
**Step 8.** Compute $s=2^{-1}(Y_1+Y_2-z') \bmod p$ where $z'=z$ if $Y_1>Y_2$; otherwise $z'=z+1$.
**Step 9.** Return $(s)$

The following Theorem demonstrates that the result of "$a\times b \bmod p$" can be generated from "$(a\times b) \bmod (p-1)$" and "$(a\times b) \bmod (p+1)$". The proposed algorithm generates the correct result of "$a\times b \bmod p$" at Step 8.

THEOREM 2.1. *Given $Y_1=W \bmod (p-1)$, $Y_2=W \bmod (p+1)$, such that $0\le W\le p^2-1$ and $\gcd(2,p)=1$, then $s=W \bmod p=2^{-1}(Y_1+Y_2-[W\ge(p^2-1)/2]-[Y_1<Y_2]) \bmod p$. Here we use Knuth's bracket notation [9]: for a Boolean-valued expression B, [B] is 0 if B is false and 1 if B is true.*

*Proof.* Since $0\le W\le p^2-1$, we may write $W=r\cdot p+s$, where $0\le r\le p-1$ and $0\le s\le p-1$.
If $s+r\le p-1$, write $W=r(p-1)+(s+r)$, so $Y_1=W \bmod (p-1)=s+r$.
Otherwise, if $s+r\ge p$, write $W=(r+1)(p-1)+(s+r-p+1)$, so $Y_1=s+r-p+1$.
Similarly, if $s-r\ge 0$, write $W=r(p+1)+(s-r)$, so $Y_2=W \bmod (p+1)=s-r$;
otherwise, if $s-r\le -1$, write $W=(r-1)(p+1)+(s-r+p+1)$, so $Y_2=s-r+p+1$.
Thus, there are four cases to consider in computing $Y_1+Y_2$.

**Case 1:** $Y_1+Y_2=s+r+s-r=2s$.
Then $s=2^{-1}(Y_1+Y_2) \bmod p$.
Since $s+r\le p-1$ and $s-r\ge 0$, we obtain $W<(p^2-1)/2$ and $Y_1>Y_2$.

**Case 2:** $Y_1+Y_2=s+r+s-r+p+1=2s+p+1$.
Then $s=2^{-1}(Y_1+Y_2-1) \bmod p$.
Since $s+r\le p-1$ and $s-r\le -1$, we obtain $W<(p^2-1)/2$ and $Y_1<Y_2$.

**Case 3:** $Y_1+Y_2=s+r-p+1+s-r=2s-p+1$.
Then $s=2^{-1}(Y_1+Y_2-1) \bmod p$.
Since $s+r\ge p$ and $s-r\ge 0$, we obtain $W\ge(p^2-1)/2$ and $Y_1>Y_2$.

**Case 4:** $Y_1+Y_2=s+r-p+1+s-r+p+1=2s+2$.
Then $s=2^{-1}(Y_1+Y_2-2) \bmod p$.
Since $s+r\ge p$ and $s-r\le -1$, we obtain $W\ge(p^2-1)/2$ and $Y_1<Y_2$.

Thus, $s = 2^{-1}(Y_1 + Y_2 - [W \geq (p^2-1)/2] - [Y_1 < Y_2]) \bmod p$, where for a Boolean-valued expression B, [B] is 0 if B is false and 1 if B is true. □

In the modular multiplication computation, the remainder with modulus $p$ can be derived from both the remainder with $p-1$ and the remainder with $p+1$ by the previous theorem. If $p-1$ and $p+1$ can be decomposed into products of mutually prime factors then a computation with numbers of smaller scale must be faster. The computations of the remainder with modulus $p-1$ and the remainder with modulus $p+1$ consist of several independent parts, so that these computations can also be performed in parallel. Additionally, $2 \times (p+1)/2 = p+1 \equiv 1 \bmod p$, so $(p+1)/2$ is the multiplicative inverse of 2 modulo $p$. Therefore, the inverse value of Step 8 can be computed efficiently. It is clear that the proposed modular multiplication algorithm is more efficient than direct modular multiplication.

## 3. Application

The key point of RSA cryptosystem performance lies in its modular exponentiation. Efficient evaluation of modular exponentiation to the power $e$ uses the square-and-multiply exponentiation method determined by the binary representation $e[k]\ldots e[1]e[0]$ of $e$ (each $e[i]$ is either 0 or 1) [10]:

RSA$(n,m,e)$
    $result = 1$;
    for $i = k$ down to 0 do
      $result = (result \times result) \bmod n$;
      if $e[k]$ then $result = (result \times m) \bmod n$;
    return $result$.

Our proposed algorithm can be used to improve the performance of computing $(result \times result) \bmod n$ and $(result \times m) \bmod n$, both of which can be derived from both the remainder with modulus $n-1$ and the remainder with modulus $n+1$. In general, a computation with numbers of smaller scale must be faster. The computations of the remainder with modulus $n-1$ and the remainder with modulus $n+1$ consist of several independent parts, so that these computations can also be performed in parallel. Clearly, we can perform the encryption of RSA more efficiently using the proposed algorithm.

In addition to reducing the computational time of RSA encryption, the proposed algorithm can greatly enhance the performance of RSA decryption. The security of RSA depends critically on the problem of factorizing $n$ into its prime factors $p$ and $q$. A recommended way of maximizing the difficulty of factorizing $n$ is to choose $p$ and $q$ as strong primes [3, 7, 8, 11]. The ANSI X9.31 standard [3] defines a prime $p$ to be strong if $p$ satisfies the following conditions:

1. $p-1$ should contain a large prime factor $u_f$ such that $p-1 = u_1 \times u_2 \times \ldots \times u_f$, where $\gcd(u_i, u_j) = 1$ for $1 \leq i < j \leq f$.
2. $p+1$ should contain a large prime factor $v_g$ such that $p+1 = v_1 \times v_2 \times \ldots \times v_g$, where $\gcd(v_i, v_j) = 1$ for $1 \leq i < j \leq g$.

The "level-2 prime" numbers $u_f$ and $v_g$ can easily be found by a probabilistic primality test. These methods use the "level-2 primes" $u_f$ and $v_g$ to find the "level-1 prime" $p$. Another prime $q$ can also be generated by using the same method and the "level-2 primes" $w_h$ and $x_i$.

Based on the strong prime criterion, our proposed algorithm can apply to RSA decryption. The structures of strong primes $p$ and $q$ are clear to the secret key holder. We first take advantage of the square-and-multiply exponentiation method and use

the level-2 prime factors as modulus, then construct level-1 modular exponentiation, and finally employ the CRT to compute the result of $c^d \bmod n$.

## 4. Computational complexity

This section demonstrates that the decryption method of utilizing our proposed modular multiplication algorithm is more efficient than both the traditional decryption method and the method based on the CRT. First, we have to define some notations as follows:

- $MOD_E(y,z)$ denotes the computational complexity of modular exponentiation ($x^y \bmod z$).
- $M(w), A(w)$ and $Mod(w)$ denote the computational complexities of multiplication, addition, and modulus, which are associated with the bit length of operand $w$.
- $l(w)$ denotes the bit length of $w$.
- $S$ denotes the computational complexity of shift operator.

By the addition chain method [9], the modulo operation $c^d \bmod n$ can be expressed as:

$$MOD_E(d,n) = 1.5 \times l(d)[M(l(n)) + 2Mod(l(n)) + 1]. \tag{4.1}$$

The multiplication and addition operations can be expressed as follows [12]:

$$M(w) = 3M(w/2) + 5A(w) + 2S, \tag{4.2}$$

$$A(w) = w/32. \tag{4.3}$$

Also, the modular operation can be conveyed by the following equations based on the divide and conquer concept [13]:

$$Mod(w) = Mod(w/2) + 4M(w/2) + 1.5A(w) + 3S. \tag{4.4}$$

Without losing its generality, we assume that all of $Mod(32), M(32), A(32)$ and $S$ take one clock cycle. By Equations 4.2 and 4.3, we obtain

$$\begin{aligned}
M(1024) &= 3M(512) + 5A(1024) + 2S \\
&= 3M(512) + 162 \\
&= 3[3M(256) + 5A(512) + 2S] + 162 \\
&= 9M(256) + 408 \\
&= 9[3M(128) + 5A(256) + 2S] + 408 \\
&= 27M(128) + 786 \\
&= 27[3M(64) + 5A(128) + 2S] + 786 \\
&= 81M(64) + 1380 \\
&= 81[3M(32) + 5A(64) + 2S] + 1380 \\
&= 243M(32) + 2352 \\
&= 2595.
\end{aligned}$$

TABLE 4.1. *Numbers of CPU clock cycles for realizing the RSA decryptions based on three different methods*

| | Traditional method | CRT method | Our method |
|---|---|---|---|
| Estimation | 18293760 | 5434328 | 2821635 |
| Simulation | 107578368 | 35201744 | 19781840 |

By Equations 4.2, 4.3 and 4.4, we obtain

$$\begin{aligned} Mod(1024) &= Mod(512) + 4M(512) + 1.5A(1024) + 3S \\ &= [Mod(256) + 4M(256) + 1.5A(512) + 3S] + 3295 \\ &= [Mod(128) + 4M(128) + 1.5A(256) + 3S] + 4294 \\ &= [Mod(64) + 4M(64) + 1.5A(128) + 3S] + 4577 \\ &= [Mod(32) + 4M(32) + 1.5A(64) + 3S] + 4646 \\ &= 4657. \end{aligned}$$

In order to ensure data security, the bit length of modulus should be 1024 at least. By Equation 4.1, the traditional decryption method can be represented as $MOD_E(d,n) = 1.5 \times 1024[M(1024) + 2Mod(1024) + 1]$. In other words, the traditional decryption method should take 18293760 clock cycles.

If the decryption method based on the CRT and the bit lengths of $p$ and $q$ are the same, the operation of the decryption method can be conveyed as $2MOD_E(d_p,p) + 3A(512) + 2M(512) + Mod(512)$. Thus, the decryption method takes 5434328 clock cycles.

Without losing its generality, we assume that the bit length of the level-2 large prime factor is about $l(n)/4$, such as: $u_f, v_g, \ldots$ [3, 4, 5, 6, 7]. The decryption method of utilizing our proposed algorithm consists of several independent parts, so that these computations can be performed in parallel. The total time needed to operate the proposed method is
$MOD_E(d_p,u_1) + 3l(n)/4[3A(256) + 2M(256) + Mod(256)] + [A(512) + Mod(512)] + 3A(512) + 2M(512) + Mod(512)$. It takes 2821635 clock cycles.

The traditional decryption method takes 18293760 clock cycles, while the decryption method based on the CRT takes 5434328 clock cycles. Significantly, the method utilizing our proposed algorithm takes only 2821635 clock cycles. The results are listed in Table 4.1.

In order to evaluate the performance of our method, we examine the methods mentioned above with the Texas Instruments TMS320C54x family of signal processors [14]. Table 4.2 shows the CPU clock cycles taken to realize the modular multiplication on ′C54x signal processor, while Table 4.1 lists the CPU clock cycles taken to process the traditional RSA decryption method, the method based on CRT, and our method, respectively. In addition, we assume the calculated CPU clock cycles for the realization of the RSA decryption with the following parameters: RSA modulus $n = 1024$ bits, RSA exponent length = 1024 bits, message length = 1024 bits, level-1 prime length = 512 bits, and level-2 large prime length = 256 bits. We compute the modular exponentiation with the square and multiply algorithm [9], which needs $3n/2$ modular multiplications for an $n$-bit exponent. It may be presumed that the computational time is about one modular multiplication for the traditional method, and about two modular multiplications for the CRT-based method.

TABLE 4.2. *Numbers of CPU clock cycles for realizing the modular multiplication*

| $a$ (bit) | $b$ (bit) | $n$ (bit) | $r=(a\times b) \bmod (n)$ (clock cycles) |
|---|---|---|---|
| 128 | 128 | 128 | 3692 |
| 256 | 256 | 256 | 8566 |
| 512 | 512 | 512 | 22888 |
| 1024 | 1024 | 1024 | 70038 |

The results are listed in Table 4.1, which show that the method utilizing our proposed algorithm takes only 15% of the computational time taken by the 1024-bit RSA traditional method, and that even the method based on the CRT must take 30% of the same computational time. In other words, the method utilizing our proposed algorithm needs only 52% of the computational cost needed by the decryption method based on the CRT. It is noteworthy that the proposed modular multiplication algorithm can significantly improve the decryption performance, giving it an increase of 82% from the speed of the traditional method and an increase of 44% from the speed of the CRT method by Texas Instruments TMS320C54x simulation. That is to say, the method utilizing our proposed algorithm is almost 2 times faster than the decryption method based on the CRT.

## 5. Conclusions

This paper proposes an efficient modular multiplication algorithm. The proposed algorithm is based on the idea that the remainder for modulus $p$ can be generated from the remainder with modulus $p-1$ and the remainder with modulus $p+1$. The proposed algorithm greatly enhances the performance of RSA decryption, in addition to reducing the computational time of RSA encryption. The computational performance analysis shows that the 1024-bit RSA decryption utilizing our proposed method takes only 15% of the computational time that the traditional method needs, while the decryption method based on the CRT must take 30% of the same computational time. In evaluating the performance of our method, we examine all three methods with Texas Instruments TMS320C54x family of signal processors. The result shows an increase of 82% from the speed of the traditional method and an increase of 44% from the speed of the CRT-based method. In a word, RSA decryption utilizing our proposed method runs almost 2 times faster than the CRT-based one.

**Acknowledgement.** This work was partially supported by the iCAST project sponsored by the National Science Council, Taiwan, under the Grants No. NSC95-3114-P-001-001-Y02.

REFERENCES

[1] R. Rivest, A. Shamir and L. Adleman, *A method for obtaining digital signature and public-key cryptosystems*, Commun. of ACM, 21(2), 120-126, 1978.

[2] A. Hayashi, *A new fast modular multiplication method and its application to modular exponentiation-based cryptography*, Electronics and Communications in Japan, 83(12), 88-93, 2000.

[3] ANSI Standard X9.31, *Digital signatures using reversible public key cryptography for the financial services industry (rDSA)*, 1998.

[4] C.-S. Lai, W.-C. Yang and C.-H. Chen, *Efficient method for generating strong primes with constraint of bit length*, Electronics Letters, 27(20), 1807-1808, 1991.

[5] J. Gordon, *Strong RSA keys*, Electronics Letters, 20(12), 514-516, 1984.

[6] L. Batina, S. B. Örs, B. Preneel and J. Vandewalle, *Hardware architectures for public key cryptography*, Integration VLSI Journal, 34, 1-64, 2003.
[7] M. J. Ganley, *Note on the generation of $P_0$ for RSA keysets*, Electronics Letters, 26(6), 369, 1990.
[8] R. D. Diaz and J. M. Masqu, *Optimal strong primes*, Information Processing Letters, 93, 47-52, 2005.
[9] D. E. Knuth, *Seminumerical Algorithms*, Volume 2 of The Art of Computer Programming. Addison-Wesley, Reading, MA, USA, 1997.
[10] C. K. Koc, *High-speed RSA implementation, Version 2.0*, RSA Laboratories, Nonmember, 1994.
[11] M. Ogiwara, *A method for generating cryptographically strong primes*, IEICE Trans. Fundamentals, E73, 6, 985-994, 1990.
[12] Davida GI, Wells DL and Kam JB, *A database encryption system with subkeys*, ACM Trans. Database Systems, 6, 312-328, 1981.
[13] M. S. Hwang, *Dynamic participation in a secure conference scheme for mobile communications*, IEEE Trans. Vehicular Technology, 48(5), 1469-1474, 1999.
[14] G. Dorôević, T. Unkašević and M. Markocić, *Optimization of modular reduction procedure in RSA algorithm implementation on assembler of TMS320C54x signal processors*, in Proc. of the IEEE 14th International Conference on Digital Signal Processing, 811-814, 2002.