J. Math. Crypt. **3** (2009), 165–174
DOI 10.1515 / JMC.2009.008

Another look at some fast modular arithmetic methods

M. Jason Hinek and Charles C. Y. Lam

Communicated by Ronald C. Mullin

Abstract. In this work we re-examine a modular multiplication and a modular exponentiation method. The multiplication method, proposed by Hayashi in 1998, uses knowledge of the factorization of both $N+1$ and $N+2$ to compute a multiplication modulo N. If both $N+1$ and $N+2$ can be factored into k equally sized relatively prime factors then the computations are done modulo each of the factors and then combined using the Chinese Remainder Theorem. It was suggested that the (asymptotic) computational costs of the method is $1/k$ of simply multiplying and reducing modulo N. We show, however, that the computational costs of the method is (asymptotically) at least as costly as simply multiplying and reducing modulo N for both squarings and general multiplications when efficient arithmetic is used. The exponentiation method, proposed by Hwang, Su, Yeh and Chen in 2005, is based on Hayashi's method and uses knowledge of the factorization of $P+1$ and $P-1$ to compute an exponentiation modulo an odd prime P. We begin by showing that the method cannot be used as a general purpose exponentiation method and then modify the method so that it can work as a general purpose modular multiplication method. Like Hayashi's method, however, this method is at best (asymptotically) only as efficient as simply multiplying and reducing modulo P.

Keywords. Fast modular arithmetic, modular exponentiation.

AMS classification. 68W40, 94A60.

1 Introduction

Modular arithmetic is a frequently used but costly computation in many cryptosystems. For example, both encryption and decryption computations in the most widely used and well known public key cryptosystem today, RSA [6], use modular exponentiation with large moduli. Since this computation is so common and so costly, it is desirable to reduce the computational costs.

One example of reducing the computational costs of decryption for RSA was observed by Quisquater and Couvreur [5]. Since the factorization of the RSA modulus $N = pq$ is known to the party receiving the encrypted message, decryption (modular exponentiation) can be carried out modulo p and modulo q and then combined via the Chinese Remainder Theorem to obtain the desired result modulo N. If we assume that p and q have the same bitlength then the computational costs of this method is (asymptotically) $1/4$ the cost of simply computing the exponentiation modulo N. Of course, this requires that the factorization of N is known. Another way of reducing the computational costs of modular exponentiations is to reduce the cost of the modular multiplications that are needed for the exponentiation.

In [2], Hayashi extended the idea of using the Chinese Remainder Theorem for modular multiplication when the modulus has unknown factorization (or is prime). Here, instead of knowing the factorization of N, Hayashi assumes that the (partial)

factorizations of $N + 1$ and $N + 2$ are known. The method consists of three steps: computing the multiplication modulo $N + 1$ using the factorization of $N + 1$ and the Chinese Remainder Theorem; computing the multiplication modulo $N + 2$ using the factorization of $N + 2$ and the Chinese Remainder Theorem; and finally combining the results to obtain the multiplication modulo N.

In [3], Hwang et al. proposed a modification of Hayashi's method to be used for modular exponentiation. In this modification, the modulus is a prime P and it is assumed that the (partial) factorization of $P - 1$ and $P + 1$ are known. In addition, they propose a modular exponentiation algorithm based on this method.

The rest of the paper is as follows. In the remainder of this section we give some notation and assumptions that we will use when estimating the computational complexity of the algorithms. In Section 2, we review Hayashi's modular multiplication method and re-examine its computational complexity. In Section 3, we review Hwang et al.'s extension of Hayashi's method and modify it so that it is a general purpose modular multiplication method. We also show that their modular exponentiation method is incorrect.

1.1 Complexity assumptions

When considering the computational complexity of an algorithm we will focus on the number of word operations as a function of the bitlength of the input size. Following the style in von zur Gathen and Gerhard [7], we will reduce the cost of all operations to the cost of integer multiplications. For integers $n \geq m$, we let $M(n)$ denote the number of word operations needed to multiply two n-bit integers and $R(n, m)$ denote the number of word operations needed to reduce an n-bit integer modulo an m-bit integer (or, compute the remainder of an n-bit integer divided by an m-bit integer).

For integer multiplication, we have $M(n) \in O(n^{1+\epsilon})$ where $\epsilon = 1$ for classical quadratic arithmetic, $\epsilon = \log_2(3) - 1$ for Karatsuba multiplication and $\epsilon \to 0$ for (hypothetical) linear multiplication (see [7] for more discussion). In particular, we will let $M(n) = cn^{1+\epsilon} + o(n^{1+\epsilon})$ for some constants $c > 0$ and $\epsilon > 0$. For division with remainder (modular reduction), the best known result, from Burnikel and Ziegler [1], is given by

$$R(2n, n) = 2K(n) + O(n \log n), \tag{1.1}$$

where $K(n) = \Theta(n^{\log_2(3)})$ is the complexity of Karatsuba multiplication. Thus, when Karatsuba multiplication is used, we have $R(2n, n) = 2cn^{\log_2(3)} + o(n^{\log_2(3)})$, for some constant c. In general, we will make the assumption that

$$2M(n) \leq R(2n, n) \leq 5M(n) + O(n), \tag{1.2}$$

where the upper bound comes from a fast remainder algorithm (see [7, Theorem 9.6]).

To gauge the computational costs of the modular multiplication methods considered here, we compare with simply multiplying and then reducing modulo N (which we refer to as the *standard method*). For a (kn)-bit modulus N, let $T_0(kn)$ denote

the number of word operations needed for the standard method. Clearly $T_0(kn) = M(kn, kn) + R(2kn, kn)$.

In both of the modular multiplication methods that we consider here, the multiplicands need to be reduced modulo several different integers. We will assume that we wish to reduce a (kn)-bit integer modulo $k = 2^\ell$ different (relatively prime) n-bit integers and let $R(kn, n, k)$ denote the number of word operations needed to compute this. The best known method for fast simultaneous modular reduction, [7, Theorem 10.15], uses a binary tree approach. For example, suppose we wish to reduce x modulo each of m_1, m_2, m_3 and m_4. We first compute x mod m_1m_2 and x mod m_3m_4, and then reduce these by the appropriate m_i. We would then have

$$R(4n, n, 4) = 2R(4n, 2n) + 4R(2n, n).$$

In general, we have

$$R(kn, n, k) = 2R(kn, kn/2) + 4R(kn/2, kn/4) + \cdots + kR(2n, n). \tag{1.3}$$

2 Hayashi's method

First we review Hayashi's modular multiplication method. For some positive integer N with unknown factorization we assume that we know the (partial) factorization of $N+1$ and $N+2$. Let $x_1, x_2 \in \mathbb{Z}_N$ and $X = x_1x_2$ so that $0 \le X \le (N-1)^2$. Hayashi's method for computing X mod N consists of the following steps:

(1) Compute $y_1 = X \bmod (N+1)$,
(2) Compute $y_2 = X \bmod (N+2)$,
(3) Combine y_1 and y_2 to obtain $y = X \bmod N$.

The first two steps are computed via the Chinese Remainder Theorem, using the known factorizations of $N+1$ and $N+2$. For the last step, Hayashi again uses the Chinese Remainder Theorem. We restate the main result underlying this last step (see §3 of [2]) in the following theorem:

Theorem 2.1 (Hayashi [2]). *For any positive integer N, let X satisfy $0 \le X \le (N-1)^2$. Given $y_1 = X \bmod (N+1)$ and $y_2 = X \bmod (N+2)$, then X satisfies*

$$X \equiv 2y_1 - y_2 + 2z \pmod N, \tag{2.1}$$

where $z = 0$ if $y_1 \ge y_2$ and $z = 1$ otherwise.

The result follows from the Chinese Remainder Theorem. In particular, applying Gauss' algorithm to solve the system of congruences (equations for y_1 and y_2) yields

$$X = (N+2)y_1 - (N+1)y_2 + z(N+1)(N+2),$$

for some integer z. Applying the constraints $0 \le y_1 < N+1$, $0 \le y_2 < N+2$ and $0 \le X \le (N-1)^2$ lead to the desired restrictions on z. For more details see [2], or the proof of Theorem 3.3 below.

2.1 Efficiency

Let $H_i(nk)$ denote the complexity of step i in Hayashi's method with modulus size kn where each of $N+1$ and $N+2$ have k equally sized factors.

From Theorem 2.1, it is clear that the last step in Hayashi's method can be computed efficiently. Given y_1 and y_2, simply compute $y' = 2y_1 - y_2 + 2z$ and reduce modulo N. Computing y' is very efficient as it requires only one bit-shift (multiplication by 2), one subtraction and possibly one addition. Further, since

$$-N + 3 \leq y' \leq 2N - 2,$$

reducing y' modulo N can be done by simply adding N to y' if $y' < 0$ or subtracting N from y' if $y' > N$. Thus, given y_1 and y_2, the number of word operations needed to compute y is equivalent to a small (constant) number of additions of integers. Therefore, the last step of Hayashi's method is very efficient. We will assume that $H_3(kn)$ is negligible compared to $H_1(kn)$ and $H_2(kn)$.

We now consider the first step in the method (the second step is the same). Let the modulus N be a (kn)-bit number and let $N+1 = p_1 \cdots p_k$ be the partial factorization of $N+1$. We assume that $\gcd(p_i, p_j) = 1$ for all $i \neq j$ and also that the bitlength of each p_i is n. Let x_1 and x_2 be the multiplicands ($x_1 = x_2$ for a squaring). This step consists of first reducing x_1 and x_2 modulo each of the p_i, computing $x_1 x_2$ modulo each of the p_i and then recombining via the Chinese Remainder Theorem to obtain $x_1 x_2$ modulo $N+1$. Thus, we have

$$H_1(n,k) = R(kn, n, k) + kM(n) + CR(n,k), \tag{2.2}$$

where $CR(n,k)$ is the complexity of the Chinese Remainder Theorem (with k inputs each n bits long). In this scenario, it is claimed in [2], that the complexity of the multiplication method approaches $1/k$ for large k and n. The analysis, however, failed to consider the cost of reducing the multiplicands modulo each of the factors. In fact, if we only consider the modular reductions notice that (using equations (1.2), (1.3) and (2.2))

$$\begin{aligned}
\frac{H_1(nk)}{T_0(nk)} &= \frac{R(kn,n,k) + kM(n) + CR(n,k)}{M(kn) + R(2kn, kn)} \\
&\geq \frac{R(kn,n,k)}{M(kn) + R(2kn,kn)} \\
&= \frac{2^s \left(2R(kn, kn/2) + 4R(kn/2, kn/4) + \cdots + kR(2n,n)\right)}{M(kn) + R(2kn,kn)} \\
&\geq \frac{2^s \left(4M(kn/2) + 8M(kn/4) + \cdots + 2kM(n) + O(kn)\right)}{M(kn) + 5M(kn) + O(kn)},
\end{aligned}$$

where $s = 0$ for a squaring and $s = 1$ for a multiplication. If we consider the limit of large n, the $O(kn)$ terms can be ignored, and we have

$$\frac{H_1(nk)}{T_0(nk)} \geq \frac{2^{1+s}}{6} \frac{2M(kn/2) + 4M(kn/4) + \cdots kM(n)}{M(kn)}.$$

In the limit of large n, we can let $M(n) = cn^{1+\epsilon}$ for some constant c and ϵ, and so we have

$$\begin{aligned}\frac{H_1(nk)}{T_0(nk)} &\geq \frac{2^{1+s}}{6}\frac{2(kn/2)^{1+\epsilon} + 4(kn/4)^{1+\epsilon} + \cdots k(kn/k)^{1+\epsilon}}{(kn)^{1+\epsilon}} \\ &= \frac{2^{1+s}}{6}\left(\frac{1}{2^\epsilon} + \frac{1}{4^\epsilon} + \cdots + \frac{1}{k^\epsilon}\right) \\ &= \frac{2^{1+s}}{6}\sum_{i=1}^{\ell} 2^{i(1-\epsilon)} \\ &= \frac{2^{1+s}}{6}\frac{1-2^{-\epsilon\ell}}{2^\epsilon - 1}.\end{aligned}$$

Considering both $H_1(nk)$ and $H_2(nk)$, we have for squarings

$$\begin{aligned}\frac{H_1(kn) + H_2(kn)}{T_0(kn)} &\geq \frac{4}{6}\left(\frac{1-2^{-\epsilon\ell}}{2^\epsilon - 1}\right) \\ &= \begin{cases} \frac{2}{3}(1-2^{-\ell}) & \text{for } \epsilon = 1 \text{ (quadratic multiplication)}, \\ \frac{4}{3}(1-\left(\frac{2}{3}\right)^{\ell}) & \text{for } \epsilon = \log_2(3) - 1 \text{ (Karatsuba multiplication)}, \\ \frac{2}{3}\ell & \text{for } \epsilon \to 0 \text{ (linear multiplication)}. \end{cases}\end{aligned}$$

The cost for multiplication ($s = 1$) will be double these costs. Also, for the Karatsuba multiplication, the bound is actually twice what is shown. This follows since we can use the fast division with remainder result that uses Karatsuba multiplication instead of the more general upper bound used. Thus, the denominator can be replaced with $3K(kn)$ instead of using $6M(kn)$. Taking this into consideration, we thus have the following lower bounds for $(H_1(nk) + H_2(nk))/T_0(nk)$:

Squaring

ℓ	1	2	∞
Quadratic	1/3	1/2	2/3
Karatsuba	8/9	40/27	8/3
Linear	2/3	4/3	∞

Multiplication

ℓ	1	2	∞
Quadratic	2/3	1	4/3
Karatsuba	16/9	80/27	16/3
Linear	4/3	8/3	∞

Thus, when efficient multiplication is used (Karatsuba or better), we see that the costs of reducing the multiplicands is already more costly than the standard method except for squarings when $N + 1$ and $N + 2$ have only $k = 2$ ($\ell = 1$) factors. Of course, in the above computations, we ignored the costs of using the Chinese Remainder Theorem to construct the multiplications modulo $N + 1$ and $N + 2$. Let's now consider this additional cost for the case of squarings ($\ell = 1$).

Using Garner's algorithm (see [4]) to implement the Chinese Remainder Theorem with two inputs, we only need two additions, two multiplications and one modular

reduction, assuming we precompute the needed inverse. Thus, we have

$$CR(n,k) = 2M(n) + R(2n,n) + O(n),$$

and so in the limit of large n we have

$$\frac{CR(n,2)}{T_0(2n)} = \frac{2M(n)+R(2n,n)}{M(2n)+R(4n,2n)} > \frac{4M(n)}{2^t 3 M(2n)} = \frac{4}{2^t 3}\frac{1}{2^{1+\epsilon}},$$

where $t = 0$ for Karatsuba multiplication and $t = 1$ otherwise. Thus, we have that the costs for each step is given by

$$\frac{CR(n,2)}{T_0(2n)} > \begin{cases} \frac{2}{9} & \epsilon = \log_2(3) - 1 \text{ (Karatsuba)}, \\ \frac{1}{3} & \epsilon \to 0 \text{ (linear)}. \end{cases}$$

Adding these costs to each of H_1 and H_2 results in $(H_1(2n)+H_2(2n))/T_0(2n) \geq 1$ for all methods with multiplication at least as efficient as Karatsuba. Adding the costs for $CR(n,2)$ does not make Hayashi's method less efficient for squaring when classical multiplication is used though.

3 Hwang et al.'s modification

Based on Hayashi's modular multiplication method, Hwang et al. [3], proposed a modular exponentiation method when the modulus is a prime P and the partial factorizations of $P-1$ and $P+1$ are known. In order to compute $x^c \bmod P$ for some positive integer c, Hwang et al.'s method consists of the following steps:

(1) Compute $y_1 = x^c \bmod P-1$,

(2) Compute $y_2 = x^c \bmod P+1$,

(3) Combine y_1 and y_2 to obtain $y = x^c \bmod P$.

As with Hayashi's multiplication method, the first two steps are computed using the Chinese Remainder Theorem, using the known factorizations of $P-1$ and $P+1$. It is also suggested in [3] that the exponent in each exponentiation (modulo the factors of $P-1$ and $P+1$) can be reduced (via Euler's theorem) to further speed up the computations. The last step is based on Hayashi's efficient combining result (Theorem 2.1). We restate the main result suggested for their last step (see §3 of [3]) in the following theorem:

Theorem 3.1 (Hwang, Su, Yeh and Chen [3]). *For any odd prime P, let $0 \leq X < (P^2-1)/2$. Given $y_1 = X \bmod (P-1)$ and $y_2 = X \bmod (P+1)$, then X satisfies*

$$X \equiv 2^{-1}(y_1 + y_2 - z) \bmod P,$$

where $z = 1$ when $y_1 \geq y_2$ and $z = 0$ otherwise. Further, the inverse of 2 modulo P is simply $\frac{1}{2}(P+1)$.

Compared to Hayashi's modular multiplication method this result has a serious shortcoming. In particular, the method is not a general purpose modular exponentiation method since it only applies to computations of $x^c \bmod P$ where x and c satisfy $x^c < \frac{1}{2}(P^2-1)$. Even when the exponentiation is a squaring, using the smallest possible exponent $c = 2$, the method excludes any computation in which $x \geq \frac{1}{\sqrt{2}}P$. Simplifying Hwang et al.'s method to only perform modular multiplication instead of exponentiation, we show below that this shortcoming can be overcome.

Let $x_1, x_2 \in \mathbb{Z}_P$ and $X = x_1 x_2$. As with Hayashi's method, if we want to be able to multiply any $x_1, x_2 \in \mathbb{Z}_P$, we need to handle any product $0 \leq X < (P-1)^2$. The result in Theorem 3.1 already handles the lower part of this range. We will show that the upper range can also be easily handled. To show this, we use the following result.

Lemma 3.2. *For any odd prime P, let $(P^2-1)/2 \leq X < (P-1)^2$. Given $y_1 = X \bmod (P-1)$ and $y_2 = X \bmod (P+1)$, then*

$$X = \tfrac{1}{2}(P+1)\,y_1 - \tfrac{1}{2}(P-1)\,y_2 + \tfrac{1}{2}(P-1)(P+1)z, \tag{3.1}$$

where $z = 1$ or 2.

Proof. Since $y_1 = X \bmod (P-1)$ and $y_2 = X \bmod (P+1)$, we know that there exist integers z_1 and z_2 such that

$$X = z_1(P-1) + y_1 \text{ and} \tag{3.2}$$
$$X = z_2(P+1) + y_2. \tag{3.3}$$

Multiplying the first equation for X by $(P+1)$ and the second by $(P-1)$, we obtain

$$(P+1)X = (P+1)(P-1)z_1 + (P+1)y_1,$$
$$(P-1)X = (P+1)(P-1)z_2 + (P-1)y_2.$$

Subtracting these equations and solving for X yields

$$X = \tfrac{1}{2}(P+1)\,y_1 - \tfrac{1}{2}(P-1)\,y_2 + \tfrac{1}{2}(P-1)(P+1)z, \tag{3.4}$$

where $z = z_1 - z_2$. To show the bounds on z, we first assume that $z \leq 0$. Using the inequalities $y_1 < P-1$, $y_2 \geq 0$ and $z \leq 0$ in equation (3.4), we obtain

$$X < \tfrac{1}{2}(P+1)(P-1) = \tfrac{1}{2}(P^2-1),$$

which contradicts our assumption on X. Thus, $z > 0$. Next we assume that $z \geq 3$. Using the inequalities $y_1 \geq 0$, $y_2 < (P+1)$ and $z \geq 3$ in equation (3.4), we obtain

$$X \geq -\tfrac{1}{2}(P-1)(P+1) + \tfrac{3}{2}(P-1)(P+1) = P^2 - 1 > (P-1)^2,$$

which, again, contradicts our assumption on X. Thus, $z < 3$. Since z is an integer, we have the desired result that $z = 1$ or 2. □

With Lemma 3.2 in hand, we can easily extend Theorem 3.1 to handle the upper part of the range with the following result.

Theorem 3.3. *For any odd prime P, let $(P^2-1)/2 \leq X < (P-1)^2$. Given $y_1 = X \bmod (P-1)$ and $y_2 = X \bmod (P+1)$, then X satisfies*

$$X \equiv 2^{-1}(y_1 + y_2 - z) \pmod{P},$$

where $z = 1$ when $y_1 \geq y_2$ and $z = 2$ otherwise. Further, the inverse of 2 modulo P is simply $(P+1)/2$.

Proof. As in the proof of Lemma 3.2, let z_1 and z_2 denote the non-negative integers that satisfy

$$X = y_1 + (P-1)z_1, \tag{3.5}$$
$$X = y_2 + (P+1)z_2, \tag{3.6}$$

and again let $z = z_1 - z_2$. From Lemma 3.2, we know that

$$X = \tfrac{1}{2}(P+1)\,y_1 - \tfrac{1}{2}(P-1)\,y_2 + \tfrac{1}{2}(P-1)(P+1)z,$$

where $z = 1$ or 2. Reducing this equation modulo P, and noting that 2^{-1} exists since P is odd, we obtain the desired relation

$$X \equiv 2^{-1}(y_1 + y_2 - z) \pmod{P},$$

where $z = 1$ or 2. To show the dependence of z on y_1 and y_2, we begin by taking the difference between equations (3.5) and (3.6), to obtain

$$0 = y_1 - y_2 + (P-1)z_1 - (P+1)z_2.$$

After some rearrangement, and recalling $z = z_1 - z_2$, this gives

$$\begin{aligned} y_2 - y_1 &= (P-1)(z_1 - z_2) - 2z_2 \\ &= (P-1)z - 2z_2. \end{aligned} \tag{3.7}$$

Next, we consider cases for the values of y_1 and y_2. First, let $y_1 \geq y_2$ so that $y_2 - y_1 \leq 0$. From equation (3.7), we have $(P-1)z - 2z_2 \leq 0$, or simply

$$(P-1)z \leq 2z_2. \tag{3.8}$$

Now, rearranging equation (3.6), we know that $(P+1)z_2 = X - y_2$. Since $X < (P-1)^2$ and $y_2 \geq 0$, this leads to $(P+1)z_2 < (P-1)^2$, which implies that $z_2 < P-1$. Using this condition on z_2 with inequality (3.8), we then have

$$(P-1)z < 2(P-1),$$

and so $z < 2$. Since we know that $z = 1$ or 2, from Lemma 3.2, we conclude that $z = 1$.

Next we consider the case $y_1 < y_2$, so that $y_2 - y_1 > 0$. From equation (3.7), we have $(P-1)z - 2z_2 > 0$, or simply

$$(P-1)z > 2z_2. \tag{3.9}$$

Again, from equation (3.6), we have $(P+1)z_2 = X - y_2$. Since $X \geq (P^2-1)/2 = (P+1)(P-1)/2$ and $y_2 < p+1$, this leads to $(P+1)z_2 > (P^2-1)/2 - (P+1)$, which implies that $z_2 > (P-1)/2 - 1$ or simply $z_2 \geq (P-1)/2$. Using this lower bound for z_2 with inequality (3.9), we then have

$$(P-1)z > 2(\tfrac{1}{2}(P-1)) = (P-1),$$

and so $z > 1$. Again, since we know that $z = 1$ or 2, from Lemma 3.2, we conclude that $z = 2$. □

Combining the results of Theorems 3.1 and 3.3, we now have a general purpose modular multiplication method for $\mathbb{Z}_P$.

3.1 Efficiency

Compared to Hayashi's method, the last (recombining) step of this method is slightly less efficient. Notice that in addition to at most three addition/subtractions, there is also the multiplication involving $2^{-1} \bmod P = (P+1)/2$. If $(y_1 + y_2 - z)$ is even, we can compute this with a single bit shift. If $(y_1 + y_2 - z)$ is odd, we can compute this by adding P to make it even and then applying a single bit shift. In this case, we need to carry out another addition. Also, this method might require two comparisons to decide if which value to assign z as opposed to one comparison in Hayashi's method. However, this last step is still very efficient.

The first two steps, however, have the exact same problem of reducing the multiplicands modulo each of the factors of $P-1$ and $P+1$. The same analysis above for Hayashi's method (Section 2.1) applies here. Thus, the method is less efficient than the standard method whenever efficient multiplication is used for all cases.

4 Modular exponentiation and RSA

As has been shown above, the costs of reducing all the multiplicands modulo each of the factors of $N+1$ and $N+2$ (or $P-1$ and $P+1$) is just as expensive as computing the modular multiplication using the standard method. We would like to point out that this shortcoming does not apply to modular exponentiation methods that carry out the exponentiations in smaller groups and then recombine the results, as used in RSA for example. Here, the modular reductions are done only once. If Hayashi's method (or Hwang's modified method) were to be used as a basic step in a modular exponentiation method, we would have to compute modular reductions at each step. Otherwise, as seen in Hwang's proposed exponentiation method, the inputs will not satisfy the constraints of the combining step.

5 Conclusion

We have shown that the modular multiplication method proposed by Hayashi, is at best only as efficient as the standard method of multiplying and reducing (asymptotically) when efficient arithmetic is used. The bottleneck of Hayashi's method is reducing the multiplicands modulo each of the factors of $N+1$ and $N+2$. We have also shown that the modular exponentiation method that Hwang et al., based on Hayashi's method, is not a general purpose exponentiation method. We have modified their method so that it is a general purpose modular multiplication method, but this method has (essentially) the same complexity as Hayashi's method.

Acknowledgments. The authors would like to thank the anonymous reviewers for pointing out some errors in the initial submission.

References

[1] C. Burnikel and J. Ziegler, *Fast Recursive Division*, Max-Planck-Institut für Informatik, Saarbrücken, Germany, Report no. MPI-I-98-1-022, 1998.

[2] A. Hayashi, *A New Fast Modular Multiplication Method and Its Application to Modular Exponentiation–Based Cryptography*, Electron. Comm. Jpn. Pt. III 83 (2000), pp. 88–93. Translated from Denshi Joho Tsushin Gakkai Ronbunshi, J81-A(10): 1372–1376 (1998).

[3] R.-J. Hwang, F.-F. Su, Y.-S. Yeh, and C.-Y. Chen, *An Efficient Decryption Method for RSA Cryptosystem*. 19th International Conference on Advanced Information Networking and Applications (AINA 2005), 28–30 March 2005, Taipei, Taiwan, 1, pp. 585–590. IEEE Computer Society, 2005.

[4] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996.

[5] J.-J. Quisquater and C. Couvreur, *Fast Decipherment algorithm for RSA public-key cryptosystems*, Electronics Letters 18 (1982), pp. 905–907.

[6] R. L. Rivest, A. Shamir, and L. M. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Commun. ACM 21 (1978), pp. 120–126.

[7] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra*. Cambridge University Press, 1999.

Received 30 September, 2007; revised 10 January, 2009

Author information

M. Jason Hinek, iCORE Information Security Lab, Department of Computer Science, University of Calgary, Calgary, Alberta, T2N 1N4, Canada.*
Email: mjhinek@alumni.uwaterloo.ca

Charles C. Y. Lam, Department of Mathematics, SCI 14, California State University, Bakersfield, Bakersfield, California, 93311-1022, USA.
Email: clam@csub.edu

*Note: work initiated at David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada.