# Faster Interleaved Modular Multiplication Based on Barrett and Montgomery Reduction Methods

Miroslav Knežević, *Member*, *IEEE*, Frederik Vercauteren, and Ingrid Verbauwhede, *Senior Member*, *IEEE*

**Abstract**—This paper proposes two improved interleaved modular multiplication algorithms based on Barrett and Montgomery modular reduction. The algorithms are simple and especially suitable for hardware implementations. Four large sets of moduli for which the proposed methods apply are given and analyzed from a security point of view. By considering state-of-the-art attacks on public-key cryptosystems, we show that the proposed sets are safe to use, in practice, for both elliptic curve cryptography and RSA cryptosystems. We propose a hardware architecture for the modular multiplier that is based on our methods. The results show that concerning the speed, our proposed architecture outperforms the modular multiplier based on standard modular multiplication by more than 50 percent. Additionally, our design consumes less area compared to the standard solutions.

**Index Terms**—Modular multiplication, Barrett reduction, Montgomery reduction, public-key cryptography.

## 1 Introduction

PUBLIC-KEY cryptography (PKC), a concept introduced by Diffie and Hellman [10] in the mid 1970s, has gained its popularity together with the rapid evolution of today's digital communication systems. The best-known public-key cryptosystems are based on factoring, i.e., RSA [25], and on the discrete logarithm problem in a large prime field (Diffie-Hellman, ElGamal, Schnorr, DSA) [19] or on an elliptic curve (ECC/HECC) [15], [20], [14]. Based on the hardness of the underlying mathematical problem, PKC usually deals with large numbers ranging from a few hundreds to a few thousands of bits in size. Consequently, efficient implementation of PKC primitives has always been a challenge.

Modular multiplication forms the basis of modular exponentiation which is the core operation of the RSA cryptosystem. It is also present in many other cryptographic algorithms including those based on ECC and HECC. In particular, if one uses projective coordinates for ECC/HECC, modular multiplication remains the most time-consuming operation for ECC. For efficient implementation of modular multiplication, the crucial operation is modular reduction. Algorithms that are most commonly used for this purpose are Barrett reduction [4] and Montgomery reduction [21].

In this study, we propose two interleaved modular multiplication algorithms based on Barrett and Montgomery modular reduction. The methods are simple and especially suitable for hardware implementations. Four large sets of moduli for which the proposed methods apply are given and analyzed from a security point of view. We propose a hardware architecture for the modular multiplier that is based on our methods. The results show that concerning the speed, our proposed architecture outperforms the modular multiplier based on standard modular multiplication by more than 50 percent. Additionally, our design consumes less area compared to the standard solutions.

The remainder of this paper is structured as follows: Section 2 describes the algorithms of Barrett and Montgomery as the two most commonly used reduction methods and presents a short overview of related work. In Section 3, we show how precomputation can be omitted and the quotient evaluation simplified in Barrett and Montgomery algorithms. Section 4 analyzes the security implications, and in Section 5, we describe a hardware implementation. Section 6 concludes the paper.

• *The authors are with the Department of Electrical Engineering, Katholieke Universiteit Leuven, ESAT/SCD-COSIC, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium.*
*E-mail: {mknezevi, fvercaut, iverbauw}@esat.kuleuven.be.*

*Manuscript received 24 Apr. 2009; revised 18 Sept. 2009; accepted 23 Jan. 2010; published online 14 Apr. 2010.*
*Recommended for acceptance by P. Montuschi.*
*For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-2009-04-0173.*
*Digital Object Identifier no. 10.1109/TC.2010.93.*



## 2 Preliminaries

In this paper, we use the following notations. A multiple-precision $n$-bit integer $A$ is represented in radix $r$ representation as $A = (A_{n_w-1} \dots A_0)_r$, where $r = 2^w$; $n_w$ represents the number of digits and is equal to $\lceil n/w \rceil$, where $w$ is a *digit size*; and $A_i$ is called a *digit* and $A_i \in [0, r-1]$. A special case is when $r = 2$ $(w = 1)$ and the representation of $A = (A_{n-1} \dots A_0)_2$ is called a *bit* representation.

To make the following discussion easier, we define the floor function for integers in the following manner. Let $U$, $M \in \mathbb{Z}$ and $M > 0$, then there exist integers $q$ and $Z$ such that $U = qM + Z$ and $0 \le Z < M$. The integer $q$ is called the *quotient* and is denoted by the floor function as

$$q = \lfloor U/M \rfloor. \tag{1}$$

The integer $Z$ is called the *remainder* and can also be represented as $Z = U \bmod M$. Note here that the floor function always rounds toward negative infinity. This is very useful for hardware implementations, where the numbers are given in two's complement representation. If the divisor is of type $2^s$, the floor function is just a simple shift to the right for $s$ positions.

### 2.1 Classical and Montgomery Modular Multiplication Methods

Given a modulus $M$ and two elements $X$, $Y \in \mathbb{Z}_M$, where $\mathbb{Z}_M$ is the ring of integers modulo $M$, the ordinary modular multiplication is defined as

$$X \times Y \triangleq X \cdot Y \bmod M.$$

Let the modulus $M$ be an $n_w$-digit integer, where the radix of each digit is $r = 2^w$. The classical modular multiplication algorithm computes $XY \bmod M$ by interleaving the multiplication and modular reduction phases, as shown in Algorithm 1. The value $q$ is called an *intermediate quotient*, while $Z$ represents an *intermediate remainder*. The calculation of $q$ at step 4 of the algorithm is done by utilizing integer division which is considered as an expensive operation, especially in hardware. The idea of using the precomputed reciprocal of the modulus $M$ and simple shift and multiplication operations instead of division was first introduced by Barrett [3], [4] in 1984. The original algorithm considers only reduction, assuming that the multiplication is performed beforehand. To explain the basic idea, we rewrite the intermediate quotient $q$ as

$$q = \left\lfloor \frac{Z}{M} \right\rfloor = \left\lfloor \frac{\frac{Z}{2^{n+\beta}} \frac{2^{n+\alpha}}{M}}{2^{\alpha-\beta}} \right\rfloor \ge \left\lfloor \frac{\left\lfloor \frac{Z}{2^{n+\beta}} \right\rfloor \left\lfloor \frac{2^{n+\alpha}}{M} \right\rfloor}{2^{\alpha-\beta}} \right\rfloor = \hat{q}. \tag{2}$$

The value $\hat{q}$ represents an estimation of the intermediate quotient $q$. In most of the cryptographic applications, the modulus $M$ is fixed during the many modular multiplications, and hence, the value $\mu = \lfloor 2^{n+\alpha}/M \rfloor$ can be precomputed and reused multiple times. Since the value of $\hat{q}$ is an estimated value, some correction steps at the end of the modular multiplication algorithm have to be performed.

**Algorithm 1.** Classical modular multiplication algorithm
**Input:** $X = (X_{n_w-1} \dots X_0)_r$, $Y = (Y_{n_w-1} \dots Y_0)_r$, $M = (M_{n_w-1} \dots M_0)_r$ where $0 \le X, Y < M$, $2^{n-1} \le M < 2^n$, $r = 2^w$ and $n_w = \lceil n/w \rceil$.
**Output:** $Z = XY \bmod M$.
1: $Z \Leftarrow 0$
2: **for** $i = n_w - 1$ downto 0 **do**
3: $Z \Leftarrow Zr + XY_i$
4: $q \Leftarrow \lfloor Z/M \rfloor$
5: $Z \Leftarrow Z - qM$
6: **end for**
7: Return $Z$.

To reduce the number of correction steps, Dhem [9] determines the values of $\alpha = w + 3$ and $\beta = -2$ for which the classical modular multiplication based on Barrett reduction needs at most one subtraction at the end of the algorithm. To make the following explanations easier, we outline a similar analysis as given in [9]. We assume that step 4 of Algorithm 1 is performed according to (2) and $\hat{q}$ is used instead.

**Analysis of Algorithm 1.** Let us first consider the first iteration of Algorithm 1 ($i = 0$). We can find an integer $\gamma$ such that $Z_0 = XY_{n_w-1} < 2^{n+\gamma}$. This represents an upper bound of $Z$ ($Z_0$ for $i = 0$). The quotient $q = \lfloor \frac{Z_0}{M} \rfloor$ can now be written as

$$q = \left\lfloor \frac{Z_0}{M} \right\rfloor = \left\lfloor \frac{\frac{Z_0}{2^{n+\beta}} \frac{2^{n+\alpha}}{M}}{2^{\alpha-\beta}} \right\rfloor,$$

where $\alpha$ and $\beta$ are two variables. The estimation of the given quotient is now equal to

$$\hat{q} = \left\lfloor \frac{\lfloor \frac{Z_0}{2^{n+\beta}} \rfloor \lfloor \frac{2^{n+\alpha}}{M} \rfloor}{2^{\alpha-\beta}} \right\rfloor = \left\lfloor \frac{\lfloor \frac{Z_0}{2^{n+\beta}} \rfloor \mu}{2^{\alpha-\beta}} \right\rfloor,$$

where $\mu = \lfloor \frac{2^{n+\alpha}}{M} \rfloor$ is a constant and may be precomputed. Let us now define the quotient error as a function of the variables $\alpha$, $\beta$, and $\gamma$

$$e = e(\alpha, \beta, \gamma) = q - \hat{q}.$$

Since $\frac{A}{B} \ge \lfloor \frac{A}{B} \rfloor > \frac{A}{B} - 1$ for any $A, B \in \mathbb{Z}$, we can write the following inequality:

$$\begin{aligned} q = \left\lfloor \frac{Z_0}{M} \right\rfloor \ge \hat{q} &> \frac{\lfloor \frac{Z_0}{2^{n+\beta}} \rfloor \lfloor \frac{2^{n+\alpha}}{M} \rfloor}{2^{\alpha-\beta}} - 1 \\ &> \frac{\left(\frac{Z_0}{2^{n+\beta}} - 1\right)\left(\frac{2^{n+\alpha}}{M} - 1\right)}{2^{\alpha-\beta}} - 1 \\ &= \frac{Z_0}{M} - \frac{Z_0}{2^{n+\alpha}} - \frac{2^{n+\beta}}{M} + \frac{1}{2^{\alpha-\beta}} - 1 \\ &\ge \left\lfloor \frac{Z_0}{M} \right\rfloor - \frac{Z_0}{2^{n+\alpha}} - \frac{2^{n+\beta}}{M} + \frac{1}{2^{\alpha-\beta}} - 1 \\ &= q - \frac{Z_0}{2^{n+\alpha}} - \frac{2^{n+\beta}}{M} + \frac{1}{2^{\alpha-\beta}} - 1. \end{aligned}$$

Now, since $e \in \mathbb{Z}$, the quotient error can be estimated as

$$e = e(\alpha, \beta, \gamma) \le \left\lfloor 1 + \frac{Z_0}{2^{n+\alpha}} + \frac{2^{n+\beta}}{M} - \frac{1}{2^{\alpha-\beta}} \right\rfloor.$$

According to Algorithm 1, we have $Z_0 < 2^{n+\gamma}$ and $M \ge 2^{n-1}$. Hence, we can evaluate the quotient error as

$$e = e(\alpha, \beta, \gamma) \le \left\lfloor 1 + 2^{\gamma-\alpha} + 2^{\beta+1} - \frac{1}{2^{\alpha-\beta}} \right\rfloor.$$

Following the previous inequality, it is obvious that for $\alpha \ge \gamma + 1$ and $\beta \le -2$, it holds $e = 1$.

Next, we need to ensure that the intermediate remainder $Z_i$ does not grow uncontrollably as $i$ increases. Since $X < M$, $Y_i < 2^w$, $Z_i < M + eM$ and $M < 2^n$, after $i$ iterations, we have

$$\begin{aligned} Z_i &= Z_{i-1}2^w + XY_i \\ &< (M + eM)2^w + M2^w \\ &< (2 + e)2^{n+w}. \end{aligned}$$

Since we want to use the same value for $e$ during the algorithm, the next condition must hold

$$Z_i < (2 + e)2^{n+w} < 2^{n+\gamma}.$$

To minimize the quotient error ($e = 1$), we must choose $\gamma$ such that

$$3 \cdot 2^w < 2^\gamma.$$

In other words, we choose $\gamma \ge w + 2$. Now, according to the previous analysis, we can conclude that for $\alpha \ge \gamma + 1$, $\beta \le -2$ and $\gamma \ge w + 2$, we may realize a modular multiplication with only one correction step at the end of the whole process.

The only drawback of the proposed method is the size of the intermediate quotient $\hat{q}$ and the precomputed value $\mu$. Due to the parameters $\alpha$ and $\beta$ chosen in a given way, the size of $\hat{q}$ is $w + 2$ and $\mu$ is at most $w + 4$ bits. This introduces an additional overhead for the software implementations, while it can be easily overcome in the hardware implementations.

Montgomery's algorithm [21] is the most commonly utilized modular multiplication algorithm today. In contrast to the classical modular multiplication, it utilizes right to left divisions. Given an $n$-digit odd modulus $M$ and an integer $U \in \mathbb{Z}_M$, the image or the Montgomery residue of $U$ is defined as $X = UR \bmod M$, where $R$, the Montgomery radix, is a constant relatively prime to $M$. If $X$ and $Y$ are the images of $U$ and $V$, respectively, the Montgomery multiplication of these two images is defined as

$$X * Y \triangleq XYR^{-1} \bmod M.$$

The result is the image of $UV \bmod M$ and needs to be converted back at the end of the process. For the sake of efficient implementation, one usually uses $R = r^{n_w}$, where $r = 2^w$ is the radix of each digit. Similar to a classical modular multiplication based on Barrett reduction, this algorithm uses a precomputed value $M' = -M^{-1} \bmod r = -M_0^{-1} \bmod r$. The algorithm is shown as follows:

**Algorithm 2.** Montgomery modular multiplication algorithm
**Input:** $X = (X_{n_w-1} \dots X_0)_r$, $Y = (Y_{n_w-1} \dots Y_0)_r$, $M = (M_{n_w-1} \dots M_0)_r$, $M' = -M_0^{-1} \bmod r$ where $0 \le X, Y < M$, $2^{n-1} \le M < 2^n$, $r = 2^w$, $\gcd(M, r) = 1$ and $n_w = \lceil n/w \rceil$.
**Output:** $Z = XYr^{-n_w} \bmod M$.
1: $Z \Leftarrow 0$
2: **for** $i = 0$ to $n_w - 1$ **do**
3: $Z \Leftarrow Z + XY_i$
4: $q_M \Leftarrow (Z \bmod r)M' \bmod r$
5: $Z \Leftarrow (Z + q_M M)/r$
6: **end for**
7: **if** $Z \ge M$ **then**
8: $Z \Leftarrow Z - M$
9: **end if**
10: Return $Z$.

### 2.2 Related Work

Before introducing related work, we note here that for the moduli used in all common ECC cryptosystems, the modular reduction can be done much faster than the one proposed by Barrett or Montgomery. Even without any multiplication. This is the reason behind standardizing generalized Mersenne prime moduli (sums/differences of a few powers of 2) [22], [1], [26].

The idea of simplifying an intermediate quotient evaluation was first presented by Quisquater [23] at the rump session of Eurocrypt '90. The method is similar to the one of Barrett except that the modulus $M$ is preprocessed before the modular multiplication in such a way that the evaluation of the intermediate quotient $q$ basically comes for free. Preprocessing requires some extra memory and computational time, but the latter is negligible when many modular multiplications are performed using the same modulus.

Lenstra [16] points out that choosing moduli with a predetermined portion is beneficial both for storage and computational requirements. He proposes a way to generate RSA moduli with any number of predetermined leading (trailing) bits, with the fraction of specified bits only limited by security considerations. Furthermore, Lenstra discusses security issues and concludes that the resulting moduli do not seem to offer less security than regular RSA moduli. In [13], Joye enhances the method for generating RSA moduli with a predetermined portion proposed in [16].

In [12], Hars proposes a long modular multiplication method that also simplifies an intermediate quotient evaluation. The method is based on Quisquater's algorithm and requires a preprocessing of the modulus by increasing its length. The algorithm contains conditional branches that depend on the sign of the intermediate remainder. That increases the complexity of the algorithm, especially concerning the hardware implementations where additional control logic needs to be added.

In this paper, we propose four sets of moduli that specifically target efficient modular multiplication by means of classical modular multiplication based on general Barrett reduction [9] and Montgomery modular multiplication [21]. In addition to simplified quotient evaluation, our algorithms do not require any additional preprocessing. The algorithms are simple and especially suitable for hardware implementations. They contain no conditional branches inside the loop, and hence, require a very simple control logic. Note that the same algorithms are applicable to general moduli if the preprocessing described in [12] is performed beforehand.

The methods describing how to generate such moduli in case of RSA are discussed in [16], [13]. Furthermore, from the sets proposed in this paper, one can also choose the primes that generate the RSA modulus to speed up a decryption of RSA by means of the Chinese Remainder Theorem (CRT). In Section 4, we discuss security issues concerning this matter.

## 3 The Proposed Modular Multiplication Methods for Integers

In both Barrett and Montgomery modular multiplications, the precomputed values of either modulus reciprocal ($\mu$) or modulus inverse ($M'$) are used in order to avoid multiple-precision divisions. However, single-precision multiplications still need to be performed (step 4 of the Algorithms 1 and 2). This especially concerns the hardware implementations, as the multiplication with the precomputed values often occurs within the critical path of the whole design. Section 5 discusses this issue in more detail.

Let us, for now, assume that the precomputed values $\mu$ and $M'$ are both of type $\pm 2^{\delta} - \varepsilon$, where $\delta \in \mathbb{Z}$ and $\varepsilon \in \{0, 1\}$. By tuning $\mu$ and $M'$ to be of this special type, we transform a single-precision multiplication with these values into a simple shift operation in hardware. Therefore, we find sets of moduli for which the precomputed values are both of type $\pm 2^{\delta} - \varepsilon$.

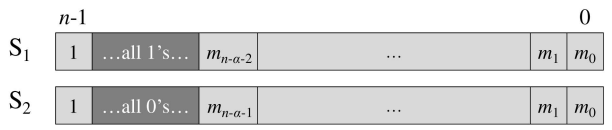


Fig. 1. Binary representation of the proposed sets $S_1$ and $S_2$.

### 3.1 Speeding Up Classical Modular Multiplication

Before describing the actual algorithm, we provide two lemmas to make the following explanation easier:

**Lemma 1.** *Let $M = 2^n - \Delta$ be an n-digit positive integer in radix 2 representation and let $\mu = \lfloor 2^{n+\alpha}/M \rfloor$, where $\alpha \in \mathbb{N}$. If $0 < \Delta \leq \lfloor \frac{2^n}{1+2^{\alpha}} \rfloor$, then*

$$\mu = 2^{\alpha}. \tag{3}$$

**Proof of Lemma 1.** Rewrite $2^{n+\alpha}$ as

$$2^{n+\alpha} = M2^{\alpha} + 2^{\alpha}\Delta.$$

Since it is given that $0 < \Delta \leq \lfloor \frac{2^n}{1+2^{\alpha}} \rfloor$, we conclude that $0 < 2^{\alpha}\Delta < M$. By the definition of euclidean division, this shows that $\mu = 2^{\alpha}$. □

**Lemma 2.** *Let $M = 2^{n-1} + \Delta$ be an n-digit positive integer in radix 2 representation and let $\mu = \lfloor 2^{n+\alpha}/M \rfloor$, where $\alpha \in \mathbb{N}$. If $0 < \Delta \leq \lfloor \frac{2^{n-1}}{2^{\alpha+1}-1} \rfloor$, then*

$$\mu = 2^{\alpha+1} - 1. \tag{4}$$

**Proof of Lemma 2.** Rewrite $2^{n+\alpha}$ as

$$2^{n+\alpha} = M(2^{\alpha+1} - 1) + 2^{n-1} - \Delta(2^{\alpha+1} - 1).$$

Since $0 < \Delta \leq \lfloor \frac{2^{n-1}}{2^{\alpha+1}-1} \rfloor$, we conclude that $0 \leq 2^{n-1} - \Delta(2^{\alpha+1} - 1) < M$. By the definition of euclidean division, this shows that $\mu = 2^{\alpha+1} - 1$. □

The interleaved modular multiplication algorithm based on general Barrett reduction is given in Section 2. Now, according to Lemmas 1 and 2, we can define two sets of moduli for which the modular multiplication based on Barrett modular reduction can be improved. These sets are of type:

$$\begin{aligned} S_1 &: M = 2^n - \Delta \quad \text{where } 0 < \Delta \leq \left\lfloor \frac{2^n}{1+2^{\alpha}} \right\rfloor, \\ S_2 &: M = 2^{n-1} + \Delta \quad \text{where } 0 < \Delta \leq \left\lfloor \frac{2^{n-1}}{2^{\alpha+1}-1} \right\rfloor. \end{aligned} \tag{5}$$

Fig. 1 further illustrates the properties of the two proposed sets $S_1$ and $S_2$. As can be seen in the figure, approximately $\alpha$ bits of the modulus are fixed to be all 0s or all 1s, while the other $n - \alpha$ bits are arbitrarily chosen.[1]

The proposed modular multiplication algorithm is shown in Algorithm 3. The parameters $\alpha$ and $\beta$ are important for the quotient evaluation. As we show later, to minimize the error in quotient evaluation, $\alpha$ and $\beta$ are chosen such that $\alpha = w + 3$ and $\beta = -2$. The same values of the parameters are obtained in [9] for the classical modular multiplication based on Barrett reduction.

1. If $M_{n-\alpha-2} = 1$ for $M \in S_1$ ($M_{n-\alpha-1} = 0$ for $M \in S_2$), then the remaining $n - \alpha - 2$ ($n - \alpha - 1$) least significant bits can be arbitrarily chosen. Otherwise, if $M_{n-\alpha-2} = 0$ ($M_{n-\alpha-1} = 1$), then the remaining $n - \alpha - 2$ ($n - \alpha - 1$) least significant bits are chosen such that (5) is satisfied.

**Algorithm 3.** Proposed interleaved modular multiplication based on generalized Barrett reduction ($\alpha = w+3$ and $\beta = -2$)

**Input:** $X = (X_{n_w-1} \dots X_0)_r$, $Y = (Y_{n_w-1} \dots Y_0)_r$, $M \in \mathrm{S}_1 \cup \mathrm{S}_2$ where $0 \le X, Y < M$, $r = 2^w$ and $n_w = \lceil n/w \rceil$.

**Output:** $Z = XY \bmod M$.

$Z \Leftarrow 0$

**for** $i = n_w - 1$ downto 0 **do**

$Z \Leftarrow Z2^w + XY_i$

$$\hat{q} = \begin{cases} \left\lfloor \dfrac{Z}{2^n} \right\rfloor & \text{if } M \in \mathrm{S}_1; \\ \left\lfloor \dfrac{Z}{2^{n-1}} \right\rfloor & \text{if } M \in \mathrm{S}_2. \end{cases}$$

$Z \Leftarrow Z - \hat{q}M$

**end for**

**if** $Z \ge M$ **then**

$Z \Leftarrow Z - M$ // At most 1 subtraction is needed.

**end if**

**while** $Z < 0$ **do**

$Z \Leftarrow Z + M$ // At most 2 additions are needed.

**end while**

return $Z$.

In contrast to the classical modular multiplication based on Barrett reduction where the quotient is evaluated as

$$\hat{q} = \left\lfloor \frac{\lfloor \frac{Z}{2^{n+\beta}} \rfloor \lfloor \frac{2^{n+\alpha}}{M} \rfloor}{2^{\alpha-\beta}} \right\rfloor,$$

in our proposed algorithm, the evaluation basically comes for free:

$$\hat{q} = \begin{cases} \left\lfloor \dfrac{Z}{2^n} \right\rfloor, & \text{if } M \in \mathrm{S}_1, \\ \left\lfloor \dfrac{Z}{2^{n-1}} \right\rfloor, & \text{if } M \in \mathrm{S}_2. \end{cases}$$

This saves one single-precision multiplication and additionally increases the speed of the proposed modular multiplication algorithm.

**Proof of Algorithm 3.** To prove the correctness of the algorithm, we need to show that there exist $\alpha, \beta \in \mathbb{Z}$, such that $\hat{q}$ can indeed be represented as

$$\hat{q} = \begin{cases} \left\lfloor \dfrac{Z}{2^n} \right\rfloor, & \text{if } M \in \mathrm{S}_1, \\ \left\lfloor \dfrac{Z}{2^{n-1}} \right\rfloor, & \text{if } M \in \mathrm{S}_2. \end{cases}$$

As shown in the analysis of Algorithm 1, to have the minimized quotient error, the parameters $\alpha$ and $\beta$ need to be chosen such that $\alpha \ge w+3$ and $\beta \le -2$. Let us first assume that $M \in \mathrm{S}_1$. According to Lemma 1, it follows that $\mu = 2^\alpha$. Now, $\hat{q}$ becomes equal to

$$\hat{q} = \left\lfloor \frac{\lfloor \frac{Z}{2^{n+\beta}} \rfloor \mu}{2^{\alpha-\beta}} \right\rfloor = \left\lfloor \frac{\lfloor \frac{Z}{2^{n+\beta}} \rfloor 2^\alpha}{2^{\alpha-\beta}} \right\rfloor = \left\lfloor \left\lfloor \frac{Z}{2^{n+\beta}} \right\rfloor 2^\beta \right\rfloor.$$

For $\beta \le 0$, the previous equation becomes equivalent to

$$\hat{q} = \left\lfloor \frac{Z}{2^n} \right\rfloor.$$

For the case where $M \in \mathrm{S}_2$, we have, according to Lemma 2, that $\mu = 2^{\alpha+1} - 1$. Now, $\hat{q}$ becomes equal to

$$\hat{q} = \left\lfloor \frac{\lfloor \frac{Z}{2^{n+\beta}} \rfloor (2^{\alpha+1}-1)}{2^{\alpha-\beta}} \right\rfloor = \left\lfloor \left\lfloor \frac{Z}{2^{n+\beta}} \right\rfloor 2^{\beta+1} \left(1 - \frac{1}{2^{\alpha+1}}\right) \right\rfloor.$$

To further simplify the proof, we choose $\beta = -2$ and the previous equation becomes equivalent to

$$\hat{q} = \left\lfloor \left\lfloor \frac{Z}{2^{n-2}} \right\rfloor \frac{1}{2} \left(1 - \frac{1}{2^{\alpha+1}}\right) \right\rfloor.$$

If we choose $\alpha$ such that

$$2^{\alpha+1} > \max\left\{ \left\lfloor \frac{Z}{2^{n-2}} \right\rfloor \right\}, \tag{6}$$

the expression of $\hat{q}$ simplifies to

$$\hat{q} = \begin{cases} \left\lfloor \dfrac{Z}{2^{n-1}} \right\rfloor - 1, & \text{if } 2 \mid \left\lfloor \dfrac{Z}{2^{n-2}} \right\rfloor, \\ \left\lfloor \dfrac{Z}{2^{n-1}} \right\rfloor, & \text{if } 2 \nmid \left\lfloor \dfrac{Z}{2^{n-2}} \right\rfloor. \end{cases} \tag{7}$$

The inequality (6) can be written as

$$2^{\alpha+1} > \left\lfloor \frac{\max\{Z\}}{2^{n-2}} \right\rfloor,$$

where $\max\{Z\}$ is evaluated in the analysis of Algorithm 1 and given as $\max\{Z\} = (2+e)2^{n+w}$. To have the minimal error, we choose $e = 1$ and get the following relation:

$$2^{\alpha+1} > \left\lfloor \frac{3 \cdot 2^{n+w}}{2^{n-2}} \right\rfloor = \lfloor 3 \cdot 2^{w+2} \rfloor.$$

The latter inequality is satisfied for $\alpha \ge w+3$.

If, instead of (7), we use only $\hat{q} = \lfloor \frac{Z}{2^{n-1}} \rfloor$, the evaluation of the intermediate quotient $\hat{q}$ will, for $2 \mid \lfloor \frac{Z}{2^{n-2}} \rfloor$, become greater than or equal to the real intermediate quotient $q$. Due to this fact, $Z$ can become negative at the end of the current iteration. Hence, we need to consider the case where $Z < 0$. Let us prevent $Z$ from an uncontrollable decrease by putting a lower bound with $Z > -2^{n+\gamma}$, where $\gamma \in \mathbb{Z}$. Since $\frac{A}{B} \ge \lfloor \frac{A}{B} \rfloor > \frac{A}{B} - 1$ for any $A$, $B \in \mathbb{Z}$, we can write the following inequality (note that $Z < 0$ and $M > 0$):

$$\begin{aligned} \hat{q} = \left\lfloor \frac{\lfloor \frac{Z}{2^{n+\beta}} \rfloor \lfloor \frac{2^{n+\alpha}}{M} \rfloor}{2^{\alpha-\beta}} \right\rfloor &\le \frac{\lfloor \frac{Z}{2^{n+\beta}} \rfloor \lfloor \frac{2^{n+\alpha}}{M} \rfloor}{2^{\alpha-\beta}} \\ &< \frac{\frac{Z}{2^{n+\beta}} \left( \frac{2^{n+\alpha}}{M} - 1 \right)}{2^{\alpha-\beta}} \\ &= \frac{Z}{M} - \frac{Z}{2^{n+\alpha}} \\ &< \left\lfloor \frac{Z}{M} \right\rfloor + 1 - \frac{Z}{2^{n+\alpha}} \\ &= q + 1 - \frac{Z}{2^{n+\alpha}} \\ &< q + 1 + 2^{\gamma-\alpha}. \end{aligned}$$

Now, since $q, \hat{q}, e \in \mathbb{Z}$, we choose $\alpha \ge \gamma + 1$ and the quotient error gets estimated as $-1 \le e \le 0$. If in the next iteration, it again happens that $2 \mid \lfloor \frac{Z}{2^{n-2}} \rfloor$, the quotient error will become $-2 \le e \le 0$.

Finally, to assure that $Z$ will remain within the bounds during the $i$th iteration, we write

$$\begin{aligned} Z_i &= Z_{i-1}2^w + XY_i \\ &= (Z_{i-2} - qM + eM)2^w + XY_i \\ &> (0 + eM)2^w + 0 \\ &> e2^{n+w} > -2^{n+\gamma}. \end{aligned}$$

The worst case is when $e = -2$, and then, it must hold $\gamma > w+1$. By choosing $\alpha = w+3$ and $\beta = -2$, all conditions are satisfied, and hence, $\hat{q}$ is indeed a good estimate of $q$. At

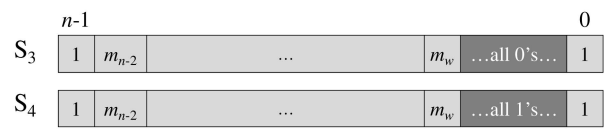


Fig. 2. Binary representation of the proposed sets $S_3$ and $S_4$.

most one subtraction or two additions at the correction step are required to obtain $Z = XY \bmod M$. □

### 3.2 Speeding Up Montgomery Modular Multiplication

Similar to Lemmas 1 and 2, we also have Lemmas 3 and 4 that are at the heart of the proposed modular multiplication algorithm based on Montgomery reduction.

**Lemma 3.** *Let $M = \Delta 2^w + 1$ be an $n$-digit positive integer in radix 2 representation, i.e., $2^{n-w-1} \le \Delta < 2^{n-w}$, and let $M' = -M^{-1} \bmod 2^w$, where $w \in \mathbb{N}$, then*

$$M' = -1. \tag{8}$$

**Proof of Lemma 3.** Since $M \equiv 1 \bmod 2^w$, we clearly have $-M^{-1} \equiv -1 \bmod 2^w$. □

**Lemma 4.** *Let $M = \Delta 2^w - 1$ be an $n$-digit positive integer in radix 2 representation, i.e., $2^{n-w-1} < \Delta \le 2^{n-w}$ and let $M' = -M^{-1} \bmod 2^w$, where $w \in \mathbb{N}$, then*

$$M' = 1. \tag{9}$$

**Proof of Lemma 4.** Since $M \equiv -1 \bmod 2^w$, we clearly have $-M^{-1} \equiv 1 \bmod 2^w$. □

According to the previous two lemmas, we can easily find two sets of moduli for which the precomputation step in Montgomery multiplication can be excluded. The resulting algorithm is shown in Algorithm 4. The proposed sets are of type

$$\begin{array}{ll} \mathrm{S}_3 : M = \Delta 2^w + 1, & \text{where } 2^{n-w-1} \le \Delta < 2^{n-w}, \\ \mathrm{S}_4 : M = \Delta 2^w - 1, & \text{where } 2^{n-w-1} < \Delta \le 2^{n-w}. \end{array} \tag{10}$$

Fig. 2 further illustrates the properties of the two proposed sets $S_3$ and $S_4$. As can be seen in the figure, $w-1$ bits of the modulus are fixed to be all 0s or all 1s, while the other $n-w+1$ bits are arbitrarily chosen. To fulfill the condition $\gcd(M, b) = 1$ (see Algorithm 2), the least significant bit of $M$ is set to 1.

**Algorithm 4.** Proposed interleaved modular multiplication based on Montgomery modular reduction.

**Input:** $X = (X_{n_w-1} \dots X_0)_r$, $Y = (Y_{n_w-1} \dots Y_0)_r$, $M \in \mathrm{S}_3 \cup \mathrm{S}_4$ where $0 \le X, Y < M$, $r = 2^w$ and $n_w = \lceil n/w \rceil$.

**Output:** $Z = XYr^{-n_w} \bmod M$.

$Z \Leftarrow 0$
**for** $i = 0$ to $n_w - 1$ **do**
  $Z \Leftarrow Z + XY_i$
  $$q_M = \begin{cases} -Z \bmod r & \text{if } M \in \mathrm{S}_3; \\ Z \bmod r & \text{if } M \in \mathrm{S}_4. \end{cases}$$
  $Z \Leftarrow (Z + q_M M)/r$
**end for**
**if** $Z \ge M$ **then**
  $Z \Leftarrow Z - M$
**end if**
return $Z$.

Due to the use of special type of moduli, the evaluation of the intermediate Montgomery quotient is simplified compared to the original algorithm given in Algorithm 2. As in our case, the value of $M'$ is simply equal to 1 or $-1$, the Montgomery quotient $q_M = (Z \bmod r)M' \bmod r$ becomes now

$$q_M = \begin{cases} -Z \bmod r, & \text{if } M \in \mathrm{S}_3, \\ Z \bmod r, & \text{if } M \in \mathrm{S}_4. \end{cases}$$

Since $r = 2^w$, the evaluation of $q$ basically comes for free.

**Proof of Algorithm 4.** Follow immediately from Lemmas 3 and 4. □

## 4 Security Considerations

In this section, we analyze the security implications of choosing primes in one of the sets $S_1, S_2, S_3, S_4$ for use in ECC/HECC and in RSA.

In the current state of the art, the security of ECC/HECC over finite fields GF($q$) only depends on the extension degree of the field [2]. Therefore, the security does not depend on the precise structure of the prime $p$. This is illustrated by the particular choices for $p$ that have been made in several standards such as SEC [26], NIST [22], ANSI [1]. In particular, the following primes have been proposed: $p_{192} = 2^{192} - 2^{64} - 1$, $p_{224} = 2^{224} - 2^{96} + 1$, $p_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, $p_{384} = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$, and $p_{521} = 2^{521} - 1$. It is easy to verify that for $w \le 28$, all primes are in one of the proposed sets. As such at least one of our methods applies for all primes included in the standards. In conclusion, choosing a prime of prescribed structure has no influence on the security of ECC/HECC.

The case of RSA requires a more detailed analysis than ECC/HECC. First, we assume that the modulus $N$ is chosen from one of the proposed sets. This is a special case of the security analysis given in [16] followed by the conclusion that the resulting moduli do not seem to offer less security than regular RSA moduli.

Next, we assume that the primes $p$ and $q$ that constitute the modulus $N = pq$ both are chosen in one of the sets $S_i$. To analyze the security implications of the restricted choice of $p$ and $q$, we first make a trivial observation. The number of $n$-bit primes in the sets $S_i$ for $n > 259 + w$ is large enough such that exhaustive listing of these sets is impossible, since a maximum of $w + 3$ bits are fixed.

The security analysis then corresponds to attacks on RSA with partially known factorization. This problem has been analyzed extensively in the literature and the first results come from Rivest and Shamir [24] in 1985. They describe an algorithm that factors $N$ in polynomial time if $2/3$ of the bits of $p$ or $q$ are known. In 1995, Coppersmith [6] improves this bound to $3/5$.

Today's best attacks all rely on variants of Coppersmith's algorithm published in 1996 [8], [7]. A good overview of these algorithms is given in [17], [18]. The best results in this area are as follows: Let $N$ be an $n$ bit number, which is a product of two $n/2$-bit primes. If half of the bits of either $p$ or $q$ (or both) are known, then $N$ can be factored in polynomial time. If less than half of the bits are known, say $n/4 - \varepsilon$ bits, then the best algorithm simply guesses $\varepsilon$ bits, and then, applies the polynomial-time algorithm, leading to a running time exponential in $\varepsilon$. In practice, the values of $w$ (typically, $w \le 64$) and $n$ ($n \ge 1{,}024$) are always such that our proposed moduli remain secure against Coppersmith's factorization algorithm, since at most $w + 3$ bits of $p$ and $q$ are known.

Finally, we consider a similar approach extended to moduli of the form $N = p^r q$, where $p$ and $q$ have the same bit size. This extension was proposed by Boneh et al. [5]. Assuming that $p$ and $q$ are of the same bit size, one needs a $1/(r+1)$-fraction of the most significant bits of $p$ in order to factor $N$ in polynomial time. In other words, for the case $r = 1$, we need half of the bits, whereas for, e.g., $r = 2$, we need only a third of the most significant bits of $p$. These results show that the primes $p, q \in \mathrm{S}$, assembling an RSA modulus of the form $N = p^r q$, should be used with care. This is especially

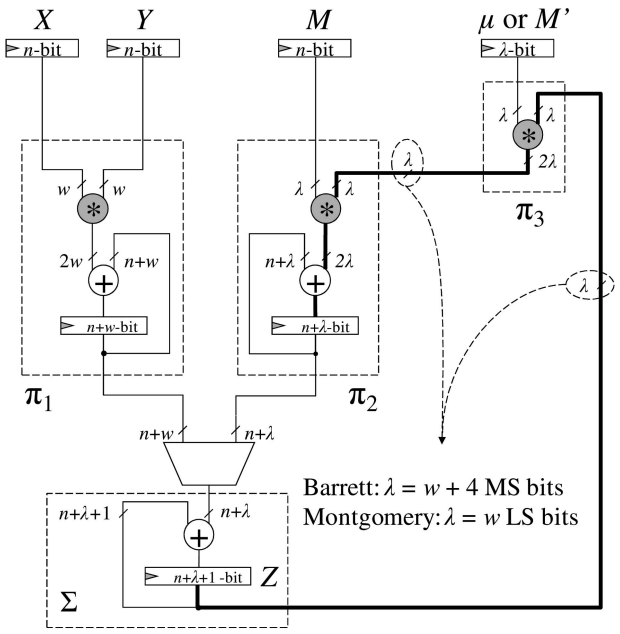


Fig. 3. Architecture for an interleaved modular multiplier based on Barrett or Montgomery reduction.

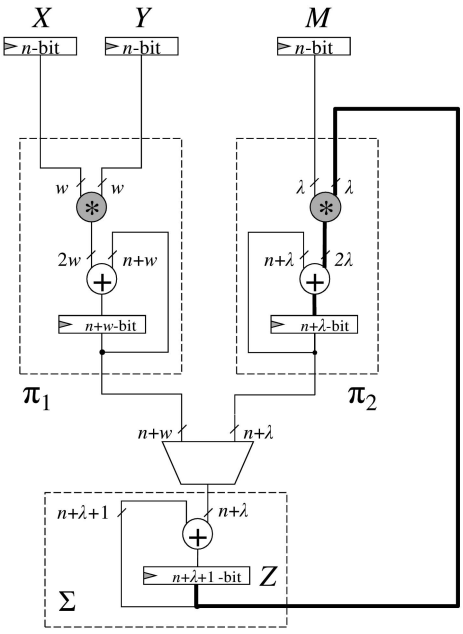


Fig. 4. Architecture for an interleaved modular multiplier based on modified Barrett or modified Montgomery reduction.

true when $r$ is large. Note that if $r \approx \log p$, the latter factoring method factors $N$ in polynomial time for any primes $p, q \in \mathbb{N}$.

## 5 Hardware Implementation of the Proposed Algorithms

A typical architecture that describes an interleaved modular multiplier is shown in Fig. 3. Both Barrett and Montgomery algorithms can be implemented based on this architecture. The architecture consists of two multiple-precision multipliers ($\pi_1$ and $\pi_2$) and one single-precision multiplier ($\pi_3$). Apart from the multipliers, the architecture contains an additional adder denoted by $\Sigma$. Having two multiple-precision multipliers may seem redundant at first glance, but the multiplier $\pi_1$ uses data from $x$ and $y$ that are fixed during a single modular multiplication. Now, by running $\pi_1$ and $\pi_2$ in parallel, we speed up the whole multiplication process. If the target is a more compact design, one can also use a single multiple-precision multiplier which does not reduce the generality of our discussion.

Multipliers $\pi_1$ and $\pi_2$ perform multiplications at lines 3 and 5 of both Algorithms 1 and 2, respectively. A multiplication performed in step 4 of both algorithms is done by multiplier $\pi_3$. An eventual shift of the register $Z$ is handled by controller. The exact schedule of the functional parts of the multiplier is as follows: $\pi_1 \rightarrow \Sigma \rightarrow \pi_1\pi_2\pi_3 \rightarrow \Sigma \rightarrow \Sigma \rightarrow \pi_1\pi_2\pi_3 \rightarrow \Sigma \rightarrow \Sigma \rightarrow \cdots$ In case of generalized Barrett reduction [9], the precomputed value $\mu$ is $\lambda = w + 4$-bits long, while for the case of Montgomery, the precomputed value $M'$ is $\lambda = w$-bits long. Due to the generalized Barrett's algorithm, the multiplier $\pi_2$ uses the most significant $\lambda$ bits of the product calculated by $\pi_3$, while for the case of Montgomery, it uses the least significant $\lambda$ bits of the same product. This is indeed a reason for Montgomery's multiplier being superior compared to the one of Barrett.

The critical path of the whole design occurs from the output of the register $Z$ to the input of the temporary register in $\pi_2$, passing through two single-precision multipliers and one adder (bold line). To show this, in practice, we have synthesized 192, 256, and 512-bit multipliers, each with the digit size of 32 bits. The code was first written in GEZEL [11] and tested for its functionality, and

TABLE 1
Synthesis Results for the Hardware Architectures of 192, 256, and 512-Bit Modular Multipliers

| Architecture | Input size $n$ [bit] | Digit size $w$ [bit] | Area [kGE] | Frequency [MHz] | Relative speed gain compared to the standard methods [%] |
|---|---|---|---|---|---|
| Classical (Algorithm 1) | 192 | 32 | 48.510 | 215 | - |
| | 256 | 32 | 54.037 | 217 | - |
| | 512 | 32 | 77.407 | 215 | - |
| Algorithm 3 | 192 | 32 | 40.756 | 327 | 52 |
| | 256 | 32 | 46.463 | 320 | 47 |
| | 512 | 32 | 71.833 | 313 | 45 |
| Montgomery (Algorithm 2) | 192 | 32 | 41.739 | 254 | - |
| | 256 | 32 | 47.278 | 251 | - |
| | 512 | 32 | 72.189 | 256 | - |
| Algorithm 4 | 192 | 32 | 39.465 | 333 | 31 |
| | 256 | 32 | 44.976 | 321 | 27 |
| | 512 | 32 | 69.616 | 303 | 18 |

then, translated to VHDL and synthesized using the Synposys Design Compiler version Y-2006.06. The library we used was UMC $0.13\ \mu m$ CMOS High-Speed standard cell library. The results can be found in Table 1. The size of the designs is given as the number of NAND gate equivalences (GEs).

A major improvement of the new algorithms is the simplified quotient evaluation. This fact results in the new proposed architecture for the efficient modular multiplier, as shown in Fig. 4. It consists of two multiple-precision multipliers ($\pi_1$ and $\pi_2$) only. The most important difference is that there are no multiplications with the precomputed values, and hence, the critical path contains one single-precision multiplier and one adder only (bold line). To compare the performance with the architecture proposed in Fig. 3, we have again synthesized a number of multipliers using the same standard cell library.

The results are given in Table 1 and show that frequencywise our proposed architecture outperforms the modular multiplier based on standard Barrett's reduction up to 52 percent. The architecture based on Montgomery's reduction results in a relative speedup up to 31 percent. Additionally, designs based on our algorithms demonstrate area savings in range from 3.5 to 14 percent. Note here that the obtained results are based on the synthesis only. After the place and route are performed, we expect a decrease of the performance for both implemented multipliers, and hence, believe that the relative speedup will approximately remain the same.

Finally, it is interesting to consider a choice of the digit size. As discussed in the previous section, the upper bound is decided by security margins. A typical digit size of 8, 16, 32, or 64 bits seems to provide a reasonable security margin for the RSA modulus of 512 bits or more. On the other side, with the increase of digit size, the number of cycles decreases for the whole design and the overall speedup is increasing. It is also obvious that the larger digit size implies the larger circuit, and thus, the performance trade-off concerning throughput and area would be interesting to explore.

## 6 Conclusion

In this work, we proposed two interleaved modular multiplication algorithms based on Barrett and Montgomery modular reductions. We introduced two sets of moduli for the algorithm based on Barrett and two sets of moduli for the algorithm based on Montgomery algorithm. These sets contain moduli with a prescribed number (typically, the digit size) of zero/one bits, either in the most significant or least significant part. Due to this choice, our algorithms have no precomputational phase and have a simplified quotient evaluation, which makes them more flexible and efficient than existing solutions.

Following the same principles as described in the paper, this approach can be easily extended to finite fields of characteristic two.

## Acknowledgments

This work is supported in part by the IAP Programme P6/26 BCRYPT of the Belgian State, by FWO project G.0300.07, by the European Commission under contract number ICT-2007-216676 ECRYPT NoE phase II, and by K.U. Leuven-BOF (OT/06/40).

## References

[1] ANSI, "ANSI X9.62 The Elliptic Curve Digital Signature Algorithm (ECDSA)," http://www.ansi.org, 2010.
[2] R.M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren, *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* CRC Press, 2005.
[3] P. Barrett, "Communications Authentication and Security Using Public Key Encryption—A Design for Implementation," master's thesis, Oxford Univ., 1984.
[4] P. Barrett, "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '86),* pp. 311-323, 1986.
[5] D. Boneh, G. Durfee, and N. Howgrave-Graham, "Factoring $N = p^r q$ for Large r," *Proc. 19th Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '99),* pp. 326-337, 1999.
[6] D. Coppersmith, "Factoring with a Hint," IBM Research Report RC 19905, 1995.
[7] D. Coppersmith, "Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known," *Proc. Int'l Conf. Theory and Application of Cryptographic Techniques (Eurocrypt '96),* 1996.
[8] D. Coppersmith, "Small Solutions to Polynomial Equations, and Low Exponent Vulnerabilities," *J. Cryptology,* vol. 10, no. 4, pp. 233-260, 1996.
[9] J.-F. Dhem, "Modified Version of the Barrett Algorithm," technical report, 1994.
[10] W. Diffie and M.E. Hellman, "New Directions in Cryptography," *IEEE Trans. Information Theory,* vol. IT-22, no. 6, pp. 644-654, Nov. 1976.
[11] GEZEL, http://www.ee.ucla.edu/~schaum/gezel, 2010.
[12] L. Hars, "Long Modular Multiplication for Cryptographic Applications," *Proc. Int'l Workshop Cryptographic Hardware and Embedded Systems (CHES '04),* pp. 218-254, 2004.
[13] M. Joye, "RSA Moduli with a Predetermined Portion: Techniques and Applications," *Proc. Information Security Practice and Experience Conf.,* pp. 116-130, 2008.
[14] N. Koblitz, "A Family of Jacobians Suitable for Discrete Log Cryptosystems," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '88),* pp. 94-99, 1988.
[15] N. Koblitz, "Elliptic Curve Cryptosystem," *Math. of Computation,* vol. 48, pp. 203-209, 1987.
[16] A. Lenstra, "Generating RSA Moduli with a Predetermined Portion," *Proc. Advances in Cryptology (ASIACRYPT '98),* pp. 1-10, 1998.
[17] A. May, "New RSA Vulnerabilities Using Lattice Reduction Methods," PhD thesis, Univ. of Paderborn, 2003.
[18] A. May, "Using LLL-Reduction for Solving RSA and Factorization Problems: A Survey," http://www.informatik.tu-darmstadt.de/KP/publications/07/lll.pdf, 2007.
[19] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1997.
[20] V. Miller, "Uses of Elliptic Curves in Cryptography," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO '85),* pp. 417-426, 1985.
[21] P. Montgomery, "Modular Multiplication without Trial Division," *Math. of Computation,* vol. 44, no. 170, pp. 519-521, 1985.
[22] National Institute of Standards and Technology. FIPS 186-2: Digital Signature Standard, Jan. 2000.
[23] J.-J. Quisquater, "Encoding System According to the So-Called RSA Method, by Means of a Microcontroller and Arrangement Implementing This System," US Patent #5,166,978, 1992.
[24] R.L. Rivest and A. Shamir, "Efficient Factoring Based on Partial Information," *Proc. Workshop Theory and Application of Cryptographic Techniques on Advances in Cryptology—EUROCRYPT '85,* pp. 31-34, 1986.
[25] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Comm. ACM,* vol. 21, no. 2, pp. 120-126, 1978.
[26] Standards for Efficient Cryptography, "Elliptic Curve Cryptography, Version 1.5, Draft," http://www.secg.org, 2005.