

Speeding Up Barrett and Montgomery Modular Multiplications

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

Abstract—This paper proposes two improved 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 art the attacks on public-key cryptosystems, we prove 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 for more than 50 %. Additionally, our design consumes less area compared to the standard solutions. Furthermore, we adapt these algorithms for finite fields of characteristic 2.

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

1 INTRODUCTION

PUBLIC-KEY cryptography (PKC), a concept introduced by Diffie and Hellman [13] in the mid 70's, 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], [16]. 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. Hence, an efficient implementation of PKC relies on efficient modular multiplication. The most popular algorithm for modular multiplication is Montgomery's method [21]. The approach of Montgomery avoids the time consuming trial division that is the common bottleneck of other algorithms. For efficient implementation of modular multiplication the crucial operation is modular reduction. Algorithms that are most commonly used for this purpose are Barrett reduction [3] and Montgomery reduction [21].

In this study we propose two 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. By considering state of art the attacks on public-key cryptosystems, we prove 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 for more than 50 %. Additionally, our design consumes less area comparing to the standard solutions. Furthermore, we adapt these algorithms for finite fields of characteristic 2.

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. In Section 6 we adapt these algorithms to finite fields $\text{GF}(2^n)$. Section 7 concludes the paper.

2 PRELIMINARIES

In the 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*; 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.

The authors are with the Department of Electrical Engineering, Katholieke Universiteit Leuven, ESAT/SCD-COSIC, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium.

* Postdoctoral Fellow of the Research Foundation - Flanders (FWO)

E-mail: {mknezevi,fvercaut,iverbauw}@esat.kuleuven.be.

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 $0 \leq Z < M$. The integer q is called the *quotient* and is denoted by the floor function as

$$q = \lfloor U/M \rfloor. \quad (1)$$

The integer Z is called the *remainder*. Note here that the floor function always rounds towards 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 Barrett 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 it is shown in Algorithm 1. A value q is called an

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 $0 \leq X, Y < M$, $2^{n-1} \leq M < 2^n$, $r = 2^w$ 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: $Z \leftarrow Z - q_C M$
 - 6: **end for**
 - 7: **Return** Z .
-

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 [2], [3] in 1984. The original algorithm considers only reduction, assuming that the multiplication is performed beforehand.

To reduce the number of correction steps, Dhem [10] introduces a generalized Barrett algorithm. The algorithm is given in Algorithm 2 where α and β are added parameters. As will be shown later, these parameters are of great importance for the quotient evaluation. The analysis of the generalized Barrett algorithm is given in [11]. To make the following explanations easier, we outline a

similar analysis and give some additional notations that are used in the later algorithms.

Algorithm 2 Generalized Barrett modular reduction for integers [10].

Input: positive integers $U = (U_{n+\gamma-1} \dots U_0)_2$ and $M = (M_{n-1} \dots M_0)_2$ where $M_{n-1} \neq 0$, $\gamma \leq n$, $\mu = \lfloor 2^{n+\alpha}/M \rfloor$.

Output: $Z = U \bmod M$.

$$\hat{q} \leftarrow \left\lfloor \frac{\lfloor \frac{U}{2^{n+\beta}} \rfloor \mu}{2^{\alpha-\beta}} \right\rfloor$$

$$Z \leftarrow U - \hat{q}M$$

if $Z \geq M$ **then**

$$Z \leftarrow Z - M$$

end if

return Z .

Analysis of Algorithm 2 (see also [11]): The quotient $q = \lfloor \frac{U}{M} \rfloor$ can be written as

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

where α and β are two variables. The estimation of the given quotient is now equal to

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

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

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

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

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

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

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

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

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

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

Modular multiplication of two n -digit inputs using Barrett reduction can be done by reducing the result of $n \times n$ -digit multiplication (input U in Algorithm 2). This algorithm implemented in hardware can perform modular multiplication at a very high throughput. However, due to the big integers used in cryptographic applications, this algorithm is not suitable for embedded devices as it requires the use of big multipliers (e.g. 512×512 -bit multiplier). To make the multiplication more efficient, interleaved reduction and multiplication can be performed. An interleaved digit-serial modular multiplication with generalized Barrett reduction is given in Algorithm 3. The quotient evaluation is done in the same way as in the generalized Barrett reduction algorithm.

Algorithm 3 Interleaved digit-serial modular multiplication with generalized Barrett reduction [11].

Input: $X = (X_{n_w-1} \dots X_0)_r$, $Y = (Y_{n_w-1} \dots Y_0)_r$, $M = (M_{n_w-1} \dots M_0)_r$, $\mu = \lfloor 2^{n+\alpha}/M \rfloor$ where $0 \leq X, Y < M$, $2^{n-1} \leq M < 2^n$, $r = 2^w$ 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:    $\hat{q} \leftarrow \lfloor \frac{\lfloor \frac{Z}{2^{n+\beta}} \rfloor \mu}{2^{\alpha-\beta}} \rfloor$ 
5:    $Z \leftarrow Z - \hat{q}M$ 
6: end for
7: if  $Z \geq M$  then
8:    $Z \leftarrow Z - M$ 
9: end if
10: Return  $Z$ .
```

Analysis of Algorithm 3 (see also [11]): Starting from the first iteration of Algorithm 3 ($i = 0$), we can find an integer γ such that

$$Z_0 = XY_{n_w-1} < 2^{n+\gamma}.$$

Since $X < M$, $Y_i < 2^w$, $Z_i < M + eM$ and $M < 2^n$, after i iterations we have

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

where $e = e(\alpha, \beta, \gamma)$ is the quotient error as a function of α , β and γ . 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 γ such that

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

In other words, we choose $\gamma \geq w + 2$. Now, according to the analysis of Algorithm 2 we can conclude that for $\alpha = w + 3$, $\beta = -2$ and $\gamma = w + 2$ we may realize a modular multiplication with only one correction step at the end of the whole process. ■

Montgomery's algorithm is one of the most commonly used reduction algorithms. In contrast to Barrett reduction it utilizes right to left divisions. The result of the reduction has the form $Z = UR^{-1} \bmod M$ for an input U and a modulus M . Similar to Barrett reduction, this algorithm uses a precomputed value $M' = -M^{-1} \bmod R$. For the sake of efficient implementation one usually uses $R = 2^n$. Algorithm 4 shows the Montgomery reduction in short. The interleaved modular multiplication can also be done based on Montgomery reduction algorithm and is shown in Algorithm 5.

Algorithm 4 Montgomery reduction for integers [21].

Input: positive integers $U = (U_{2n-1} \dots U_0)_2$, $M = (M_{n-1} \dots M_0)_2$, $R = 2^n$ where $\gcd(M, 2) = 1$, $M' = -M^{-1} \bmod R$.

Output: $Z = UR^{-1} \bmod M$.

```

 $q \leftarrow (U \bmod R)M' \bmod R$ 
 $Z \leftarrow (U + qM)/R$ 
if  $Z \geq M$  then
   $Z \leftarrow Z - M$ 
end if
return  $Z$ .
```

Algorithm 5 Interleaved digit-serial modular multiplication with Montgomery reduction.

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 $0 \leq X, Y < M$, $2^{n-1} \leq M < 2^n$, $r = 2^w$, $\gcd(M, r) = 1$ 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:    $q \leftarrow (Z \bmod r)M' \bmod r$ 
5:    $Z \leftarrow (Z + qM)/r$ 
6: end for
7: if  $Z \geq M$  then
8:    $Z \leftarrow Z - M$ 
9: end if
10: Return  $Z$ .
```

2.2 Related Work

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 multiplications are performed with the same modulus.

In [14] 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.

Besides the simplified quotient evaluation, our algorithms do not require any additional preprocessing. Instead, we propose four large sets of moduli for which the proposed algorithms work. 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 any general moduli if the preprocessing as described in [14] is performed beforehand.

3 THE PROPOSED MODULAR MULTIPLICATION METHODS FOR INTEGERS

In both Barrett and Montgomery modular multiplication algorithms, the precomputed values of either modulus reciprocal (μ) 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 3 and 5). This especially concerns the hardware implementations, as the multiplication with the precomputed values 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 μ and M' are both of type $\pm 2^\delta - \varepsilon$ where $\delta \in \mathbb{Z}$ and $\varepsilon \in \{0, 1\}$. By tuning μ 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 $\pm 2^\delta - \varepsilon$.

3.1 Speeding Up Barrett Modular Multiplication

Before describing the actual algorithm, we provide two lemmata 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

$$\mu = 2^\alpha . \quad (2)$$

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 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

$$\mu = 2^{\alpha+1} - 1 . \quad (3)$$

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 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 Lemmata 1 and 2 we can define two sets of moduli for which the modular multiplication based on Barrett modular reduction described in Algorithm 3 can be improved. These sets are of type

$$\begin{aligned} S_1 : M &= 2^n - \Delta \quad \text{where } 0 < \Delta \leq \lfloor \frac{2^n}{1+2^\alpha} \rfloor; \\ S_2 : M &= 2^{n-1} + \Delta \quad \text{where } 0 < \Delta \leq \lfloor \frac{2^{n-1}}{2^{\alpha+1}-1} \rfloor. \end{aligned} \quad (4)$$

Figure 1 further illustrates the properties of the two proposed sets S_1 and S_2 . As we can see from the figure, approximately α bits of the modulus are fixed to be all 0's or all 1's, while the other $n - \alpha$ bits are arbitrarily chosen¹.

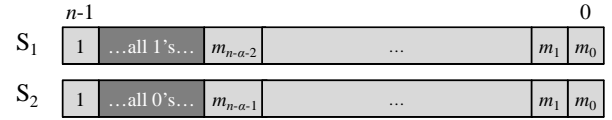


Fig. 1. Binary representation of the proposed sets S_1 and S_2 .

The proposed modular multiplication algorithm is shown in Algorithm 6. Again, α and β are parameters, important for the quotient evaluation. As we show later, to minimize the error in quotient evaluation, α and β are chosen such that $\alpha = w + 3$ and $\beta = -2$.

In contrast to the original Barrett algorithm where the quotient is evaluated as $\hat{q} = \lfloor \frac{\lfloor \frac{Z}{2^{n+\beta}} \rfloor \lfloor \frac{2^{n+\alpha}}{M} \rfloor}{2^{\alpha-\beta}} \rfloor$, in our proposed algorithm the evaluation is simplified to

$$\hat{q} = \begin{cases} \lfloor \frac{Z}{2^n} \rfloor & \text{if } M \in S_1; \\ \lfloor \frac{Z}{2^{n-1}} \rfloor & \text{if } M \in S_2. \end{cases}$$

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

Proof of Algorithm 6: To prove the correctness of the algorithm, we need to show that for $\alpha = w + 3$ and $\beta = -2$, \hat{q} can indeed be represented as

$$\hat{q} = \begin{cases} \lfloor \frac{Z}{2^n} \rfloor & \text{if } M \in S_1; \\ \lfloor \frac{Z}{2^{n-1}} \rfloor & \text{if } M \in S_2. \end{cases}$$

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 Equation 4 is satisfied.

Algorithm 6 Proposed digit-serial modular multiplication based on Barrett modular reduction ($\alpha = w + 3$ and $\beta = -2$).

Input: $X = (X_{n_w-1} \dots X_0)_r$, $Y = (Y_{n_w-1} \dots Y_0)_r$, $M \in S_1 \cup S_2$ where $0 \leq X, Y < M$, $r = 2^w$ and $n_w = \lceil n/w \rceil$.

Output: $Z = XY \bmod M$.

```

 $Z \leftarrow 0$ 
for  $i = n_w - 1$  downto  $0$  do
   $Z \leftarrow Z2^w + XY_i$ 
   $\hat{q} = \begin{cases} \lfloor \frac{Z}{2^n} \rfloor & \text{if } M \in S_1; \\ \lfloor \frac{Z}{2^{n-1}} \rfloor & \text{if } M \in S_2. \end{cases}$ 
   $Z \leftarrow Z - \hat{q}M$ 
end for
if  $Z \geq M$  then
   $Z \leftarrow Z - M$ 
end if
while  $Z < 0$  do
   $Z \leftarrow Z + M$  // At most 2 additions are needed.
end while
return  $Z$ .

```

Let us first assume that $M \in S_1$. According to Lemma 1 it follows that $\mu = 2^\alpha$. Now, \hat{q} becomes equal to

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

For $\beta \leq 0$ the previous equation becomes equivalent to

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

For the case where $M \in S_2$ we have, according to Lemma 2, that $\mu = 2^{\alpha+1} - 1$. Now, \hat{q} becomes equal to

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

For $\beta = -2$ 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 α such that

$$2^{\alpha+1} > \max \left\{ \left\lfloor \frac{Z}{2^{n-2}} \right\rfloor \right\}, \quad (5)$$

the expression of \hat{q} simplifies to

$$\hat{q} = \begin{cases} \left\lfloor \frac{Z}{2^{n-1}} \right\rfloor - 1 & \text{if } 2 \mid \left\lfloor \frac{Z}{2^{n-2}} \right\rfloor; \\ \left\lfloor \frac{Z}{2^{n-1}} \right\rfloor & \text{if } 2 \nmid \left\lfloor \frac{Z}{2^{n-2}} \right\rfloor. \end{cases} \quad (6)$$

The inequality (5) 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 3 and given as

$$\max\{Z\} = (2 + e)2^{n+w}.$$

To have the minimal error, we choose $e = 1$ and get the following relation

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

Latter inequality is satisfied for $\alpha \geq w + 3$.

If, instead of Equation 6 we use \hat{q} as $\hat{q} = \left\lfloor \frac{Z}{2^{n-1}} \right\rfloor$, the evaluation of the intermediate quotient \hat{q} will, for $2 \mid \left\lfloor \frac{Z}{2^{n-2}} \right\rfloor$, 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 $\frac{A}{B} \geq \left\lfloor \frac{A}{B} \right\rfloor > \frac{A}{B} - 1$ for any $A, B \in \mathbb{Z}$, we can write the following inequality (note that $Z < 0$ and $M > 0$)

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

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

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 most one subtraction or 2 additions at the correction step are required to obtain $Z = XY \bmod M$. This concludes the proof. ■

3.2 Speeding Up Montgomery Modular Multiplication

Similar to Lemmata 1 and 2 we also have Lemmata 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. $2^{n-w-1} \leq \Delta < 2^{n-w}$, and let $M' = -M^{-1} \bmod 2^w$ where $w \in \mathbb{N}$, then

$$M' = -1. \quad (7)$$

Proof of Lemma 3: Since $M \equiv 1 \pmod{2^w}$ we clearly have $-M^{-1} \equiv -1 \pmod{2^w}$. ■

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

$$M' = 1. \quad (8)$$

Proof of Lemma 4: Since $M \equiv -1 \pmod{2^w}$ we clearly have $-M^{-1} \equiv 1 \pmod{2^w}$. ■

According to the previous two lemmata 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 7. The proposed sets are of type

$$\begin{aligned} S_3 : M &= \Delta 2^w + 1 \quad \text{where } 2^{n-w-1} \leq \Delta < 2^{n-w}; \\ S_4 : M &= \Delta 2^w - 1 \quad \text{where } 2^{n-w-1} < \Delta \leq 2^{n-w}. \end{aligned} \quad (9)$$

Figure 2 further illustrates the properties of the two proposed sets S_3 and S_4 . As we can see from the figure, $w-1$ bits of the modulus are fixed to be all 0's or all 1's, while the other $n-w+1$ bits are arbitrarily chosen. To fulfill the condition $\gcd(M, b) = 1$ (see Algorithm 5), the least significant bit of M is set to 1.

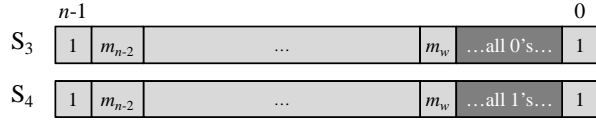


Fig. 2. Binary representation of the proposed sets S_3 and S_4 .

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 5. As in our case the value of M' is simply equal to 1 or -1 , the Montgomery quotient $q = (Z \bmod R)M' \bmod R$ becomes now

$$q = \begin{cases} -Z \bmod R & \text{if } M \in S_3; \\ Z \bmod R & \text{if } M \in S_4. \end{cases}$$

Similar to the proposed modular multiplication algorithm based on Barrett reduction, this fact also increases the overall performance of the proposed algorithm.

Proof of Algorithm 7: Follows immediately from Lemma 3 and Lemma 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.

4.1 ECC/HECC

In the current state of the art, the security of ECC/HECC over prime fields $\text{GF}(p)$ does not depend at all on the precise structure of the prime p . This is illustrated by

Algorithm 7 Proposed digit-serial modular multiplication based on Montgomery modular reduction.

Input: $X = (X_{n_w-1} \dots X_0)_r, Y = (Y_{n_w-1} \dots Y_0)_r, M \in S_3 \cup S_4$ where $0 \leq X, Y < M, r = 2^w$ and $n_w = \lceil n/w \rceil$.

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

```

Z ← 0
for i = 0 to n_w - 1 do
  Z ← Z + XY_i
  q = { -Z mod r  if M ∈ S_3;
        Z mod r   if M ∈ S_4.
  Z ← (Z + qM)/r
end for
if Z ≥ M then
  Z ← Z - M
end if
return Z.

```

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 $w \leq 28$ all primes are in one of the proposed sets. As such at least one of our methods applies for all primes included in the standards². In conclusion: choosing a prime of prescribed structure has no influence on the security of ECC/HECC.

4.2 RSA

The case of RSA requires a more detailed analysis than ECC/HECC. We will assume that the primes p and q that constitute the modulus $N = pq$ both are chosen in one of the sets S_i . Note that if Δ is chosen two times smaller than the maximum allowed by Lemma 1 or 2, then for $p, q \in S_1$ or S_2 , we have $N \in S_1$ or S_2 . Furthermore, if $p, q \in S_3$ or S_4 , then the modulus $N \in S_4$, so our methods not only apply for p and q , but also for N itself.

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 [7] improves this bound to $3/5$.

Today's best attacks all rely on variants of Coppersmith's algorithm published in 1996 [9], [8]. A good overview of these algorithms is given in [17], [18]. The

² Please note that for the moduli used in all common ECC cryptosystems the modular reduction can be done even faster (without any multiplication). This is the reason behind standardizing generalized Mersenne prime moduli (sums/differences of a few powers of 2) [5].

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 ε bits and then applies the polynomial time algorithm, leading to a running time exponential in ε . In practice, the values of w (typically $w \leq 64$) and n ($n \geq 1024$) 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, Durfee and Howgrave-Graham [4]. 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 third of the most significant bits of p . These results show that the primes $p, q \in S$, assembling an RSA modulus of the form $N = p^r q$, should be used with care. This is especially 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 Figure 3. Both Barrett and Montgomery algorithms can be implemented based on this architecture. The architecture consists of two multiple-precision multipliers (π_1 and π_2) and one single-precision multiplier (π_3). Having two multiple-precision multipliers may seem redundant at first glance, but the multiplier π_1 uses data from x and y that are fixed during a single modular multiplication. Now, by running π_1 and π_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 π_1 and π_2 perform multiplications at the lines 3 and 5 of both Algorithms 3 and 5, respectively. A multiplication performed in step 4 of both algorithms is done by multiplier π_3 . In case of generalized Barrett reduction [10], the precomputed value μ 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 π_2 uses the most significant λ bits of the product calculated by π_3 , while for the case of Montgomery, it uses the least significant λ bits of the same product. This is indeed a reason for Montgomery's multiplier being superior comparing to the one of Barrett's.

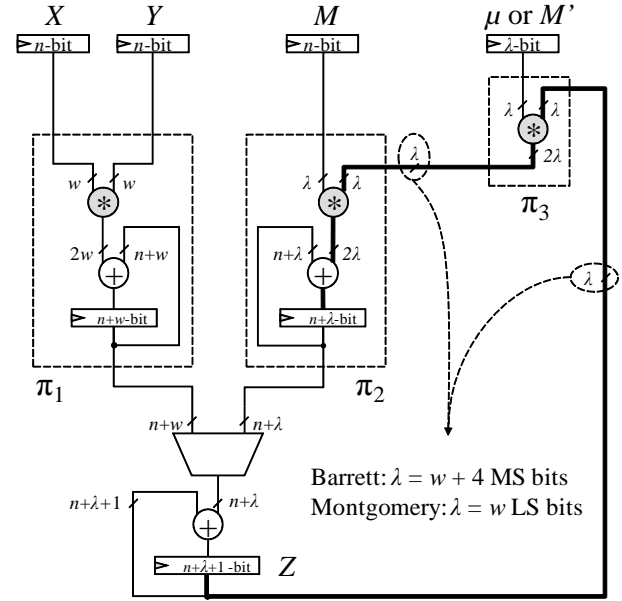


Fig. 3. Architecture for an interleaved modular multiplier based on Barrett or Montgomery reduction.

The critical path of the whole design occurs from the output of the register Z to the input of the temporary register in π_2 , passing through two single-precision multipliers and one adder (bold line). To show this in practice, we have synthesized 192-bit, 256-bit and 512-bit multipliers, each with the digit size of 8, 16 and 32 bits. The code was synthesized in $0.13 \mu\text{m}$ CMOS technology, using Synopsys Design Vision tool and the standard cell library. The results can be found in Tables 1 and 2.

A main improvement of the new algorithms is a simplified quotient evaluation. This fact results in the new proposed architecture for the efficient modular multiplier as shown in Figure 4. It consists of two multiple-precision multipliers (π_1 and π_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 Figure 3, we have again synthesized a number of multipliers in $0.13 \mu\text{m}$ CMOS technology, using Synopsys Design Vision tool and the same standard cell library.

The results are given in Tables 1 and 2 and show that frequency-wise our proposed architecture outperforms the modular multiplier based on standard Barrett's reduction for more than 50 %. The architecture based on Montgomery's reduction gets a relative speed-up of around 30 %. To further illustrate the obtained results, we provide two charts in Figure 5 and Figure 6 where the critical path delay versus occupied area is given. Designs that are closer to the origin are the more superior ones.

Finally, we would like to note that the size of some of the proposed designs with $w = 8$ and $w = 16$ bits

TABLE 1

Synthesis results for the hardware architectures of 192-bit, 256-bit and 512-bit modular multipliers based on Barrett reduction.

Architecture	Input size n [bit]	Digit size w [bit]	Area [kGE]	Frequency [MHz]	Relative speed gain compared to the standard methods [%]
Barrett	192	8	24.93	135.6	-
		16	30.31	110.0	-
		32	51.52	87.1	-
	256	8	33.15	137.6	-
		16	37.61	109.1	-
		32	58.00	131.0	-
	512	8	62.69	127.6	-
		16	67.08	107.9	-
		32	86.63	84.5	-
Proposed Algorithm 6	192	8	25.01	180.5	33.1
		16	28.03	156.5	42.3
		32	43.51	130.4	49.7
	256	8	33.40	177.3	28.9
		16	35.97	153.8	41.0
		32	50.77	131.0	53.4
	512	8	67.07	159.8	25.2
		16	66.13	149.0	38.1
		32	79.20	121.7	44.0

TABLE 2

Synthesis results for the hardware architectures of 192-bit, 256-bit and 512-bit modular multipliers based on Montgomery reduction.

Architecture	Input size n [bit]	Digit size w [bit]	Area [kGE]	Frequency [MHz]	Relative speed gain compared to the standard methods [%]
Montgomery	192	8	22.72	171.5	-
		16	26.00	132.1	-
		32	41.57	101.3	-
	256	8	29.94	173.6	-
		16	33.14	133.9	-
		32	48.60	100.5	-
	512	8	58.53	157.5	-
		16	58.35	127.6	-
		32	74.46	100.7	-
Proposed Algorithm 7	192	8	23.00	190.1	10.8
		16	25.30	162.3	22.9
		32	38.59	130.9	29.2
	256	8	29.84	181.5	4.5
		16	32.60	157.0	17.3
		32	45.68	127.4	26.9
	512	8	60.13	163.7	3.9
		16	58.09	144.5	13.2
		32	71.87	121.7	20.9

is larger than the size of the designs based on standard Barrett and standard Montgomery method (see Table 1 and Table 2). This is due to the synthesis tool that uses larger gates in order to achieve a higher frequency. To prove this, we have also synthesized all designs with the digit size of $w = 8$ and $w = 16$ bits, using the lower frequency and the final size was at least 10 % smaller in all cases.

6 THE PROPOSED MULTIPLICATION METHODS IN $\text{GF}(2^n)$

Following the same principles described in Section 3, we first show how the original Barrett multiplication can be adapted for the interleaved digit-serial multiplication over $\text{GF}(2^n)$. Second, we also provide a special set of moduli for which the digit-serial multiplication in $\text{GF}(2^n)$

based on Barrett reduction has no precomputation and has a simplified quotient evaluation. Finally, we show how the interleaved digit-serial multiplication based on Montgomery reduction for a complementary set of moduli, can also be performed without precomputation and with simplified quotient evaluation. As these algorithms operate in a binary field, they are specially suitable for efficient hardware implementations.

Before describing the actual algorithms, we give some mathematical background of finite field arithmetic. Each element of the field $\text{GF}(2^n)$ can be represented as a polynomial of degree less than or equal to $n - 1$, written as $A(x) = \sum_{i=0}^{n-1} a_i x^i$ where $a_i \in \text{GF}(2)$. Similarly, the same element $A(x)$ can be written in digit-representation as

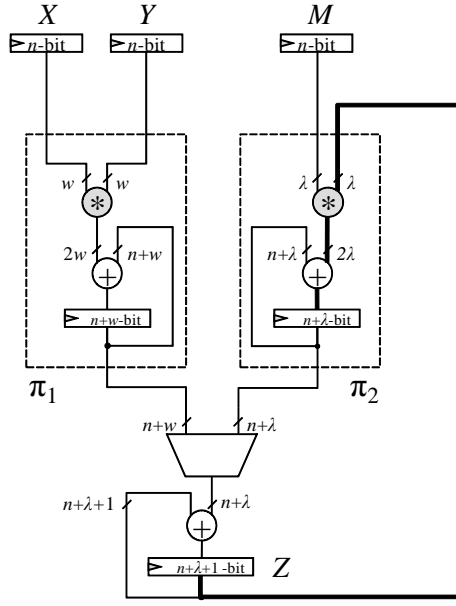


Fig. 4. Architecture for an interleaved modular multiplier based on modified Barrett or modified Montgomery reduction.

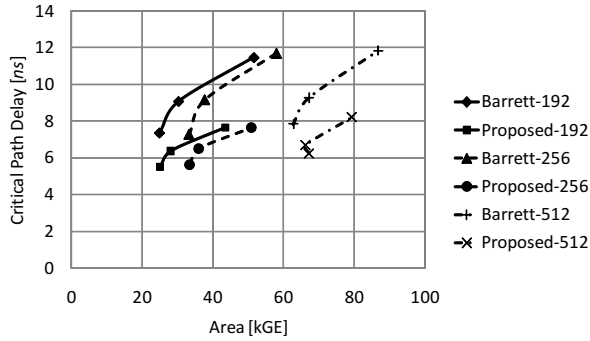


Fig. 5. Critical Path Delay vs Area graph for multipliers based on Barrett and our proposed method.

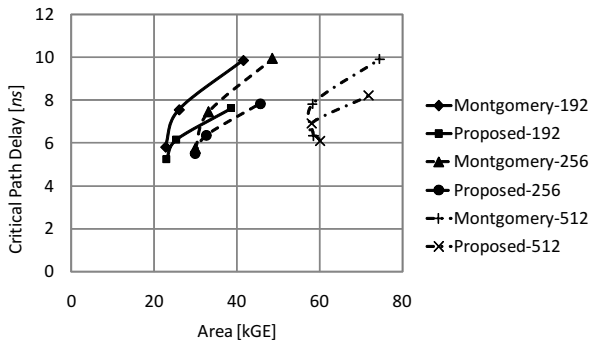


Fig. 6. Critical Path Delay vs Area graph for multipliers based on Montgomery and our proposed method.

$$A(x) = \sum_{i=0}^{n_w-1} A_i(x)x^{iw} \text{ where } n_w = \lceil n/w \rceil \text{ is the number}$$

of digits and $A_i(x)$ is a polynomial of degree $w-1$ such that $A_i(x) = \sum_{j=0}^{w-1} a_{iw+j}x^j$.

A multiplication in $\text{GF}(2^n)$ is multiplication modulo an irreducible polynomial that is used to define the field. Let $A(x)$ and $B(x)$ represent two elements in $\text{GF}(2^n)$ and let $M(x)$ be an irreducible polynomial of degree n over $\text{GF}(2)$ used to define the finite field. The multiplication of $A(x)$ and $B(x)$ over $\text{GF}(2^n)$ is defined as

$$A(x) \times B(x) \triangleq A(x) \cdot B(x) \bmod M(x) . \quad (10)$$

To make the following discussion easier we introduce the floor function for polynomials in the following manner. Let $U(x)$ be a polynomial over $\text{GF}(2)$, then there exist unique polynomials $q(x)$ and $Z(x)$ over $\text{GF}(2)$ where the degree of $Z(x)$ is less or equal to $n-1$, such that $U(x) = q(x)M(x) + Z(x)$. The polynomial $q(x)$ is called the quotient and is denoted by the floor function as

$$q(x) = \lfloor U(x)/M(x) \rfloor = U(x) \text{ div } M(x) . \quad (11)$$

6.1 Improved Multiplication in $\text{GF}(2^n)$ Based on Barrett Reduction

The following lemma is the analogue of Lemma 2.

Lemma 5: Let $M(x) = x^n + \Delta(x)$ be an irreducible polynomial over $\text{GF}(2)$ such that $\Delta(x) = \sum_{i=0}^{n-w} m_i x^i$ where $1 < w < n$, $m_i \in \text{GF}(2)$ and let $\mu(x) = \lfloor x^{n+w-1}/M(x) \rfloor$. Then it holds:

$$\mu(x) = x^{w-1} . \quad (12)$$

Proof of Lemma 5: Rewrite x^{n+w-1} as

$$x^{n+w-1} = x^{w-1}M(x) + x^{w-1}\Delta(x) .$$

Since $\deg(x^{w-1}\Delta(x)) \leq n-1$ and $\deg(M(x)) = n$, we conclude that the quotient is indeed $\mu(x) = x^{w-1}$. ■

In [12] the authors present a digit-serial multiplication in $\text{GF}(2^n)$ based on Barrett modular reduction. Here we outline the same algorithm in Algorithm 8 and based on it, we propose the improved digit-serial multiplication algorithm in Algorithm 9.

Proof of Algorithm 9: The correctness of the algorithm follows directly from Lemma 5 since $\mu = x^{w-1}$ and the quotient evaluation thus becomes

$$\hat{q}(x) = \left\lfloor \frac{\lfloor \frac{U_i(x)}{x^{n-1}} \rfloor \mu(x)}{x^{w-1}} \right\rfloor = \left\lfloor \frac{\lfloor \frac{U_i(x)}{x^{n-1}} \rfloor x^{w-1}}{x^{w-1}} \right\rfloor = \left\lfloor \frac{U_i(x)}{x^{n-1}} \right\rfloor . \quad \blacksquare$$

6.2 Improved Multiplication in $\text{GF}(2^n)$ Based on Montgomery Reduction

Similar to Lemma 5 we also give Lemma 6 that is at the heart of the improved multiplication in $\text{GF}(2^n)$ based on Montgomery reduction.

Lemma 6: Let $M(x) = x^n + x^w\Delta(x) + 1$ be an irreducible polynomial over $\text{GF}(2)$ such that $\Delta(x) =$

Algorithm 8 Digit-serial multiplication in $GF(2^n)$ based on Barrett reduction [12].

Input: $A(x) = \sum_{i=0}^{n_w-1} A_i(x)x^{iw}$, $B(x) = \sum_{i=0}^{n_w-1} B_i(x)x^{iw}$, irreducible polynomial $M(x) = \sum_{i=0}^n m_i x^i$ where $m_n = 1$, $n_w = \lceil n/w \rceil$ and $\mu(x) = \lfloor x^{n+w-1}/M(x) \rfloor$.

Output: $Z(x) = A(x)B(x) \bmod M(x)$.

$Z(x) \leftarrow 0$

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

$Z(x) \leftarrow Z(x)x^w + A(x)B_i(x)$

$\hat{q}(x) \leftarrow \lfloor \frac{\lfloor \frac{Z(x)}{x^{n-1}} \rfloor \mu(x)}{x^{w-1}} \rfloor$

$Z(x) \leftarrow Z(x) + \hat{q}(x)M(x)$

end for

return $Z(x)$.

Algorithm 9 Proposed multiplication in $GF(2^n)$ based on Barrett reduction.

Input: $A(x) = \sum_{i=0}^{n_w-1} A_i(x)x^{iw}$, $B(x) = \sum_{i=0}^{n_w-1} B_i(x)x^{iw}$, irreducible polynomial $M(x) = x^n + \Delta(x)$ where $\Delta(x) = \sum_{i=0}^{n-w} m_i x^i$, $n_w = \lceil n/w \rceil$.

Output: $Z(x) = A(x)B(x) \bmod M(x)$.

$Z(x) \leftarrow 0$

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

$Z(x) \leftarrow Z(x)x^w + A(x)B_i(x)$

$\hat{q}(x) \leftarrow \lfloor \frac{Z(x)}{x^{n-1}} \rfloor$

$Z(x) \leftarrow Z(x) + \hat{q}(x)M(x)$

end for

return $Z(x)$.

$\sum_{i=0}^{n-w-1} m_i x^i$ where $1 < w < n$, $m_i \in GF(2)$ and let $M'(x) = M(x)^{-1} \bmod x^w$. Then it holds:

$$M'(x) = 1. \quad (13)$$

Proof of Lemma 6: Note that $M(x) \equiv 1 \bmod x^w$, which shows immediately that $M'(x) = 1$. ■

The Montgomery multiplication algorithm for finite fields of characteristic 2 was proposed in [6] and is outlined in Algorithm 10. Based on this algorithm we give an improved digit-serial multiplication in $GF(2^n)$ that skips the precomputation and has a simplified quotient evaluation (see Algorithm 11).

Proof of Algorithm 11: The correctness follows immediately from Lemma 6 since $M'(x) = 1$. ■

7 CONCLUSION

In this work we proposed two improved modular multiplication algorithms based on Barrett and Montgomery modular reductions. We introduced two sets of moduli

Algorithm 10 Digit-serial multiplication in $GF(2^n)$ based on Montgomery reduction [6].

Input: $A(x) = \sum_{i=0}^{n_w-1} A_i(x)x^{iw}$, $B(x) = \sum_{i=0}^{n_w-1} B_i(x)x^{iw}$, irreducible polynomial $M(x) = \sum_{i=0}^n m_i x^i$ where $m_n = 1$, $n_w = \lceil n/w \rceil$, $r(x) = x^w$ and $M'(x) = -M_0(x)^{-1} \bmod r(x)$.

Output: $Z(x) = A(x)B(x)r(x)^{-n_w} \bmod M(x)$.

$Z(x) \leftarrow 0$

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

$Z(x) \leftarrow Z(x) + A(x)B_i(x)$

$q(x) \leftarrow (Z(x) \bmod r(x))M'(x) \bmod r(x)$

$Z(x) \leftarrow (Z(x) + q(x)M(x))/r(x)$

end for

return $Z(x)$.

Algorithm 11 Proposed digit-serial multiplication in $GF(2^n)$ based on Montgomery reduction.

Input: $A(x) = \sum_{i=0}^{n_w-1} A_i(x)x^{iw}$, $B(x) = \sum_{i=0}^{n_w-1} B_i(x)x^{iw}$, irreducible polynomial $M(x) = x^n + \Delta(x) + 1$ where $\Delta(x) = \sum_{i=w}^{n-1} m_i x^i$, $r(x) = x^w$, $n_w = \lceil n/w \rceil$.

Output: $Z(x) = A(x)B(x)r(x)^{-n_w} \bmod M(x)$.

$Z(x) \leftarrow 0$

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

$Z(x) \leftarrow Z(x) + A_i(x)B(x)$

$q(x) \leftarrow Z(x) \bmod r(x)$

$Z(x) \leftarrow (Z(x) + M(x)q(x))/r(x)$

end for

return $Z(x)$.

for the improved Barrett and two sets of moduli for the improved 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. We have also proposed an architecture for a modular multiplier that is based on our method. The results show that, concerning the speed, our proposed architecture outperforms the modular multiplier based on standard Barrett method for more than 50 % with no additional area overhead. Similar sets of moduli are also defined for binary fields.

ACKNOWLEDGMENT

This work is funded partially by IBBT, Katholieke Universiteit Leuven (OT/06/40) and FWO projects (G.0300.07 and G.0450.04). This work was supported in part by the IAP Programme P6/26 BCRYPT of the

Belgian State (Belgian Science Policy), by the EU IST FP6 projects (ECRYPT) and by the IBBT-QoE project of the IBBT.

- [25] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [26] Standards for Efficient Cryptography. Elliptic Curve Cryptography, Version 1.5, draft, 2005. Available at: <http://www.secg.org>.

REFERENCES

- [1] ANSI. ANSI X9.62 The Elliptic Curve Digital Signature Algorithm (ECDSA). <http://www.ansi.org>.
- [2] P. Barrett. Communications authentication and security using public key encryption - A design for implementation. Master's thesis, Oxford University, 1984.
- [3] P. Barrett. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. In *Proc. CRYPTO'86*, pages 311–323, 1986.
- [4] D. Boneh, G. Durfee, and N. Howgrave-Graham. Factoring $n = p^r q$ for large r . In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 326–337, London, UK, 1999. Springer-Verlag.
- [5] M. Brown, D. Hankerson, J. López, and A. Menezes. Software implementation of the NIST elliptic curves over prime fields. *Lecture Notes in Computer Science*, 2020:250–??, 2001.
- [6] Ç.K. Koç and T. Acar. Montgomery multiplication in $GF(2^k)$. *Designs, Codes and Cryptography*, 14:57–69, 1998.
- [7] D. Coppersmith. Factoring with a hint, 1995. IBM Research Report RC 19905.
- [8] D. Coppersmith. Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. In *Proceedings of Eurocrypt '96 vol. 1070*. Lecture Notes in Computer Science, 1996.
- [9] D. Coppersmith. Small solutions to polynomial equations, and low exponent vulnerabilities. In *Journal of Cryptology*, 10(4), pages 233–260, 1996.
- [10] J.-F. Dhem. *Modified version of the Barrett algorithm*. Technical report, 1994.
- [11] J.-F. Dhem. *Design of an efficient public-key cryptographic library for RISC-based smart cards*. PhD Thesis, 1998.
- [12] J.-F. Dhem. Efficient Modular Reduction Algorithm in $\mathbb{F}_q[x]$ and Its Application to Left to Right Modular Multiplication in $\mathbb{F}_2[x]$. In *Proceedings of 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, Lecture Notes in Computer Science, pages 203–213. Springer-Verlag, 2003.
- [13] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.
- [14] L. Hars. Long Modular Multiplication for Cryptographic Applications. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 218 – 254, 2004.
- [15] N. Koblitz. Elliptic curve cryptosystem. *Math. Comp.*, 48:203–209, 1987.
- [16] N. Koblitz. A family of Jacobians suitable for Discrete Log Cryptosystems. In S. Goldwasser, editor, *Advances in Cryptology: Proceedings of CRYPTO'88*, number 403 in *Lecture Notes in Computer Science*, pages 94–99. Springer-Verlag, 1988.
- [17] A. May. *New RSA Vulnerabilities Using Lattice Reduction Methods*. PhD Thesis, University of Paderborn, 2003.
- [18] A. May. Using LLL-Reduction for Solving RSA and Factorization Problems: A Survey, 2007. Available at: <http://www.informatik.tu-darmstadt.de/KP/publications/07/lll.pdf>.
- [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. In H. C. Williams, editor, *Advances in Cryptology: Proceedings of CRYPTO'85*, number 218 in *Lecture Notes in Computer Science*, pages 417–426. Springer-Verlag, 1985.
- [21] P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.
- [22] National Institute of Standards and Technology. FIPS 186-2: Digital Signature Standard, January 2000.
- [23] J.-J. Quisquater. Encoding system according to the so-called RSA method, by means of a microcontroller and arrangement implementing this system, 1992. US Patent #5,166,978.
- [24] R. L. Rivest and A. Shamir. Efficient factoring based on partial information. In *Proc. of a workshop on the theory and application of cryptographic techniques on Advances in cryptology—EUROCRYPT '85*, pages 31–34, New York, NY, USA, 1986. Springer-Verlag New York, Inc.