Modular Reduction without Precomputational Phase

Miroslav Knežević, Lejla Batina and Ingrid Verbauwhede
Katholieke Universiteit Leuven
ESAT/SCD-COSIC and IBBT
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{Miroslav.Knezevic, Lejla.Batina, Ingrid.Verbauwhede}@esat.kuleuven.be

***Abstract*—In this paper we show how modular reduction for integers with Barrett and Montgomery algorithms can be implemented efficiently without using a precomputational phase. We propose four distinct sets of moduli for which this method is applicable. The proposed modifications of existing algorithms are very suitable for fast software and hardware implementations of some public-key cryptosystems and in particular of Elliptic Curve Cryptography. Additionally, our results show substantial improvement when a small number of reductions with a single modulus is performed.**

***Index Terms*—Modular reduction, modular multiplication, Barrett reduction, Montgomery reduction, public-key cryptography, elliptic curve cryptography (ECC).**

I. Introduction

As embedded systems evolve from isolated devices to always-on networked devices, security becomes a paramount issue. After more than two decades of extensive research on both, theoretical and practical aspects it is evident that Elliptic curve cryptography (ECC) [1], [2] offers equivalent security as RSA for much smaller key sizes. This results in hardware of smaller footprint and lower power consumption. These features are very important for applications with very firm constraints on area, power, energy etc. *i.e.* for embedded security.

At the same time, a distributed computing starts to play very important role in the embedded systems field and we need more and more powerful servers that will take over the computationally expensive tasks from the constrained devices. In contrast to most of the research work that is focused on providing efficient algorithms on a constrained device itself, here we target servers in mobile network applications as possible subjects of optimization. A modular multiplication, which is the core operation for many public-key cryptosystems (*e.g.* ECC), is one of the computationally expensive algorithms where the high-throughput implementation of modular reduction is necessary.

In this work, we are focusing on the arithmetic. We propose four distinct sets of moduli for which either Barrett or Montgomery reduction algorithm can be performed without using a precomputational step. While using pseudo-Mersenne primes in software implementations brings obvious advantages, our method is more suitable for the hardware implementations. It uses Barrett and Montgomery reductions that are known to be the most efficient reduction algorithms so far. Based on the proposed reduction methods, a modular multiplication algorithm can be implemented at a very high throughput. Moreover, most of the NIST recommended moduli for ECC are also included in the proposed sets (curves P-192, P-224, P-384, P-521) [3].

Our previous work [4] showed that the reduction algorithms of Barrett and Montgomery over binary fields can be performed without using a precomputational step. The set of moduli was accordingly extended by defining two distinct sets for which either reduction method is applicable. Due to the similarity between the two algorithms, a hardware architecture for the fast modular multiplier handling both types of moduli was also proposed. In this work, we extend the approach to the ring of integers.

II. Related Work

Barrett reduction algorithm was introduced by P. D. Barrett in 1986 [5]. This algorithm computes $r = a \bmod m$ for an input a and a modulus m and is given in Alg. 1. The algorithm uses μ, a precomputed reciprocal of m, to avoid computationally expensive divisions that are necessary to compute the quotient q such that $a = qm + r$.

Algorithm 1 Barrett reduction for integers.

INPUT: positive integers $a = (a_{2n-1} \dots a_0)_b$ and $m = (m_{n-1} \dots m_0)_b$, where $m_{n-1} \neq 0$, $b \geq 3$ and PRECOMPUTED $\mu = \lfloor b^{2n}/m \rfloor$.
OUTPUT: $r = a \bmod m$.
$\hat{q} \Leftarrow \left\lfloor \frac{\lfloor \frac{a}{b^{n-1}} \rfloor \mu}{b^{n+1}} \right\rfloor$,
$r \Leftarrow a \bmod b^{n+1} - m\hat{q} \bmod b^{n+1}$,
if $r < 0$ **then**
  $r \Leftarrow r + b^{n+1}$.
end if
while $r \geq m$ **do**
  $r \Leftarrow r - m$.
end while
RETURN: r.

Based on this algorithm, a modular multiplication can be performed at a very high throughput by providing $2n$-digit product and performing the modular reduction afterwards.

Montgomery algorithm [6] is one of the most commonly used reduction algorithms. The result of reduction has a form $aR \bmod m$. Similar to Barrett reduction, this algorithm uses a precomputed value $m' = -m^{-1} \bmod R$. For the sake of efficient implementation one usually uses $R = b^n$ where b

is a radix. Algorithm 2 shows the Montgomery reduction in short. Similar to Barrett reduction, modular multiplication of two n-bit inputs can also be done based on Alg. 2.

Algorithm 2 Montgomery reduction for integers.

INPUT: positive integers $a = (a_{2n-1} \dots a_0)_b$, $m = (m_{n-1} \dots m_0)_b$, $R = b^n$, where $\gcd(b, m) = 1$ and PRECOMPUTED $m' = -m^{-1} \bmod R$.
OUTPUT: $t = aR^{-1} \bmod m$.
$s \Leftarrow (a \bmod R)m' \bmod R$,
$t \Leftarrow (a + ms)/R$,
if $t \geq m$ **then**
$t \Leftarrow t - m$
end if
RETURN: t.

Both described, Barrett and Montgomery algorithms have one property in common. In order to perform modular reduction, they need a precomputed value of the reciprocal/inverse of modulus. This reduces flexibility of the system forcing one to use fixed modulus and its precomputed reciprocal/inverse. From the implementation point of view, this requires extra computational time and memory space to calculate the precomputed value. Hence, both algorithms are suitable only for the case when many reductions are performed with a single modulus. In the next section we show how these shortcomings can be overcome using the special sets of moduli.

III. The Proposed Modular Reduction Methods

Here, we first provide two special sets of moduli for which the precomputational step in Barrett reduction can be efficiently avoided. Second, we propose a modular reduction algorithm that is based on Barrett reduction and show how using a modulus from the defined sets can be beneficial for skipping the precomputational step. Finally, we show how Montgomery reduction, with using complementary sets of moduli, can also be performed very efficiently without precomputation. A modular multiplication based on these algorithms can be implemented at a very high throughput in hardware. Due to the big integers used in cryptographic applications, the algorithms are not suitable for embedded devices as they require the use of big multipliers (*e.g.* 163×163-bit multiplier)[1]. On the other hand, they are very suitable for powerful servers where the fast modular multiplication is necessary.

Before describing the actual algorithms, we give some mathematical background to make the following explanations easier. Starting with the basic idea of the proposed modular reduction based on Barrett algorithm we give next two Lemmata as follows:

Lemma 1: Let $m = b^{n-1} + \Delta$ be n-word positive integer in radix b representation, such that $0 \leq \Delta < \lfloor b^{(n-1)/2} \rfloor$ and $\mu = \lfloor b^{2n-2}/m \rfloor$. Then it holds:

$$\mu = b^{n-1} - \Delta \ . \tag{1}$$

Proof of Lemma 1: Rewrite b^{2n-2} as

$$b^{2n-2} = (b^{n-1} - \Delta)m + \Delta^2 \, .$$

Since it is given that $0 \leq \Delta < \lfloor b^{(n-1)/2} \rfloor$, we conclude that $0 \leq \Delta^2 < m$. By definition of Euclidean division, this shows that $\mu = b^{n-1} - \Delta$. ■

Lemma 2: Let $m = b^n - \Delta$ be n-word positive integer in radix b representation, such that $0 < \Delta < \lfloor b^{n/2} \rfloor$ and $\mu = \lfloor b^{2n}/m \rfloor$. Then it holds:

$$\mu = b^n + \Delta \ . \tag{2}$$

Proof of Lemma 2: Rewrite b^{2n} as

$$b^{2n} = (b^n + \Delta)m + \Delta^2 \, .$$

Since it is given that $0 < \Delta < \lfloor b^{n/2} \rfloor$, we conclude that $0 < \Delta^2 < m$. By definition of Euclidean division, this shows that $\mu = b^n + \Delta$. ■

The Barrett modular reduction algorithm for integers is given in Sect. II. Now, according to Lemmata 1 and 2, we can define two sets of primes for which the Barrett reduction described in Alg. 1 can be performed without using a precomputational phase. These sets are of type:

$$\begin{array}{ll} \mathrm{S}_1 : m = b^{n-1} + \Delta_1 & \text{where } 0 \leq \Delta_1 < \lfloor b^{(n-1)/2} \rfloor; \\ \mathrm{S}_2 : m = b^n - \Delta_2 & \text{where } 0 < \Delta_2 < \lfloor b^{n/2} \rfloor. \end{array} \tag{3}$$

To further illustrate the properties of the two proposed sets, we give Fig. 1 where the moduli from each set are represented in radix 2 representation. Note that here $k = \left\lfloor \frac{n-1}{2} \right\rfloor$, $m_i \in \{0, 1\}$ and, additionally, Eq. (3) has to be satisfied.

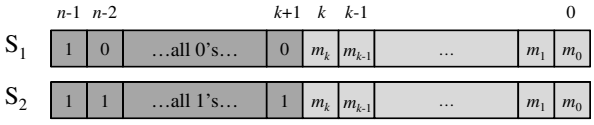

Fig. 1. Binary representation of the proposed sets S_1 and S_2.

The precomputed reciprocal needed for the Barrett algorithm can be easily formed together with the modulus as:

$$\mu = \begin{cases} b^{n-1} - \Delta_1 & \text{if } m \in \mathrm{S}_1; \\ b^n + \Delta_2 & \text{if } m \in \mathrm{S}_2. \end{cases}$$

The proposed modular reduction based on Barrett algorithm is shown in Alg. 3. It is important to note here that using this algorithm with the modulus m we can efficiently reduce an integer a that is less than m^2. Since we assume that, a priori multiplication, all the inputs are reduced and less than m, this does not make any additional constraints.

Proof of Alg. 3: To show the correctness of the algorithm we first assume that $m \in \mathrm{S}_1$. The case when $m \in \mathrm{S}_2$ is

[1]We consider only the case where the multiplication is performed at the first place, followed by the reduction afterwards. Implemented in this way, the algorithm is performed faster than the bit/digit serial multiplication interleaved with reduction.

Algorithm 3 Modular reduction without precomputation based on Barrett algorithm.

INPUT: positive integers $a = (a_{2n-1} \dots a_0)_b$, $m \in \mathrm{S}_1 \cup \mathrm{S}_2$, μ, $b \geq 3$.
OUTPUT: $r = a \bmod m$.

$$\hat{q} \Leftarrow \begin{cases} \left\lfloor \frac{\lfloor \frac{a}{b^{n-1}} \rfloor \mu}{b^{n-1}} \right\rfloor & \text{if } m \in \mathrm{S}_1; \\ \left\lfloor \frac{\lfloor \frac{a}{b^{n-1}} \rfloor \mu}{b^{n+1}} \right\rfloor & \text{if } m \in \mathrm{S}_2. \end{cases}$$

$r \Leftarrow a \bmod b^{n+1} - m\hat{q} \bmod b^{n+1}$,
CORRECTION STEP:
if $r < 0$ **then**
  $r \Leftarrow r + b^{n+1}$.
end if
while $r \geq m$ **do**
  $r \Leftarrow r - m$.
end while
RETURN: r.

completely analogous. Let $q = \lfloor a/m \rfloor$ and $r = a \bmod m = a - qm$. In the algorithm above, $\hat{q}$ is an estimate of q since

$$\frac{a}{m} = \frac{a}{b^{n-1}} \cdot \frac{b^{2n-2}}{m} \cdot \frac{1}{b^{n-1}} \ .$$

We now show that $q - 3 \leq \hat{q} \leq q$. The right part of the inequality is straightforward to prove as it is

$$\hat{q} = \left\lfloor \frac{\lfloor \frac{a}{b^{n-1}} \rfloor \mu}{b^{n-1}} \right\rfloor \leq \left\lfloor \frac{a}{b^{n-1}} \cdot \frac{b^{2n-2}}{m} \cdot \frac{1}{b^{n-1}} \right\rfloor = \left\lfloor \frac{a}{m} \right\rfloor = q \ .$$

Next, we prove the left part of the inequality. Since $\frac{A}{B} \geq \left\lfloor \frac{A}{B} \right\rfloor > \frac{A}{B} - 1$ for any $A, B \in \mathbb{N}$, we can write the following inequality

$$\begin{aligned} q &= \left\lfloor \frac{\frac{a}{b^{n-1}} \cdot \frac{b^{2n-2}}{m}}{b^{n-1}} \right\rfloor \\ &\leq \left\lfloor \frac{\left(\lfloor \frac{a}{b^{n-1}} \rfloor + 1\right)\left(\lfloor \frac{b^{2n-2}}{m} \rfloor + 1\right)}{b^{n-1}} \right\rfloor \\ &= \left\lfloor \frac{\lfloor \frac{a}{b^{n-1}} \rfloor \mu}{b^{n-1}} + \frac{\lfloor \frac{a}{b^{n-1}} \rfloor + \lfloor \frac{b^{2n-2}}{m} \rfloor + 1}{b^{n-1}} \right\rfloor \ . \end{aligned}$$

Since $a < m^2$ and $m = b^{n-1} + \Delta_1 \geq b^{n-1}$, where $0 \leq \Delta_1 < \lfloor b^{(n-1)/2} \rfloor$, it follows that

$$\begin{aligned} \left\lfloor \frac{a}{b^{n-1}} \right\rfloor + \left\lfloor \frac{b^{2n-2}}{m} \right\rfloor + 1 &\leq \left\lfloor \frac{m^2}{b^{n-1}} \right\rfloor + b^{n-1} + 1 \\ &= b^{n-1} + 2\Delta_1 + \left\lfloor \frac{\Delta_1^2}{b^{n-1}} \right\rfloor + b^{n-1} + 1 \\ &\leq 2b^{n-1} + 2\Delta_1 + 2 \ . \end{aligned}$$

Finally, we have

$$\begin{aligned} q &\leq \left\lfloor \frac{\lfloor \frac{a}{b^{n-1}} \rfloor \mu}{b^{n+1}} + \frac{2b^{n-1} + 2\Delta_1 + 2}{b^{n-1}} \right\rfloor \\ &= \left\lfloor \frac{\lfloor \frac{a}{b^{n-1}} \rfloor \mu}{b^{n+1}} + 2 + \frac{2\Delta_1 + 2}{b^{n-1}} \right\rfloor \\ &\leq \hat{q} + 3 \ . \end{aligned}$$

Similarly, for the case when $m \in \mathrm{S}_2$ we have[2]

$$\begin{aligned} q &= \left\lfloor \frac{\frac{a}{b^{n-1}} \cdot \frac{b^{2n}}{m}}{b^{n+1}} \right\rfloor \\ &\leq \left\lfloor \frac{\left(\lfloor \frac{a}{b^{n-1}} \rfloor + 1\right)\left(\lfloor \frac{b^{2n}}{m} \rfloor + 1\right)}{b^{n+1}} \right\rfloor \\ &\leq \left\lfloor \frac{\lfloor \frac{a}{b^{n-1}} \rfloor \mu}{b^{n+1}} + 2 + \frac{2\Delta_2 b + 2}{b^{n+1}} \right\rfloor \\ &\leq \hat{q} + 3 \ . \end{aligned}$$

Hence, $\hat{q}$ is indeed a good estimate of q and at most 3 subtractions at the correction step are required to obtain $r = a \bmod m$. This concludes the proof. ■

In contrast to the original Barrett algorithm for integers (see Alg. 1), our proposed algorithm differs not only in the lack of the precomputational phase, but also in the number of the correction steps (see Alg. 3). While in the original Barrett reduction algorithm, the number of correction steps is at most 2, in our modified reduction algorithm this number can be at most 3. One can further reduce the number of redundant subtractions by increasing a precision of μ for two or more digits. The same approach was applied to the original Barrett algorithm, resulting in the improved Barrett reduction where at most one subtraction needs to be performed at the correction step [7].

Similar to Lemmata 1 and 2 we also give Lemmata 3 and 4 that are the base points for the proposed modular reduction based on Montgomery algorithm.

***Lemma** 3:* Let $m = \Delta b^k + 1$ be n-word positive integer in radix b representation where $b^{n-k-1} \leq \Delta < b^{n-k}$, $k = \left\lceil \frac{n-1}{2} \right\rceil$ and let $m' = -m^{-1} \bmod b^{n-1}$. Then it holds:

$$m' = \Delta b^k - 1 \ . \tag{4}$$

Proof of Lemma 3: In order to prove Eq. (4) we need to show that

$$mm' \equiv -1 \bmod b^{n-1} \ .$$

Indeed, if we express product mm' as

$$\begin{aligned} mm' &= (\Delta b^k + 1)(\Delta b^k - 1) \\ &= \Delta^2 b^{2k} - 1 \ , \end{aligned}$$

it becomes obvious that $mm' \equiv -1 \bmod b^{n-1}$. ■

[2]Due to the space limitation, we skip a few steps of the following computation. These steps are equivalent to the steps from the case when $m \in \mathrm{S}_1$.

Lemma 4: Let $m = \Delta b^k - 1$ be n-word positive integer in radix b representation where $b^{n-k-1} < \Delta \leq b^{n-k}$, $k = \left\lceil \frac{n-1}{2} \right\rceil$ and let $m' = -m^{-1} \bmod b^{n-1}$. Then it holds:

$$m' = \Delta b^k + 1 \ . \tag{5}$$

Proof of Lemma 4: Analogous to the proof of Lemma 3, we write

$$\begin{aligned} mm' &= (\Delta b^k - 1)(\Delta b^k + 1) \\ &= \Delta^2 b^{2k} - 1 \\ &\equiv -1 \bmod b^{n-1} \ . \end{aligned}$$

This concludes the proof. ∎

According to Lemmata 3 and 4, we can easily find two sets of moduli for which the precomputational step in Montgomery reduction can be excluded. The proposed sets are of type

$$\begin{aligned} &\mathrm{S}_3 : m = \Delta_3 b^k + 1 \quad \text{where } b^{n-k-1} \leq \Delta_3 < b^{n-k}; \\ &\mathrm{S}_4 : m = \Delta_4 b^k - 1 \quad \text{where } b^{n-k-1} < \Delta_4 \leq b^{n-k}, \end{aligned} \tag{6}$$

where $k = \left\lceil \frac{n-1}{2} \right\rceil$. To further illustrate the properties of the two proposed sets, we give Fig. 2 where the moduli from each set are represented in radix 2 representation. Note that here $k = \left\lceil \frac{n-1}{2} \right\rceil$ and $m_i \in \{0, 1\}$.

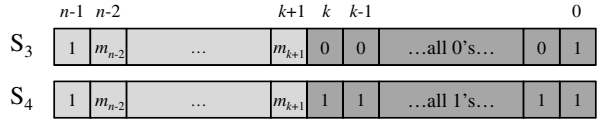

Fig. 2. Binary representation of the proposed sets S_3 and S_4.

The precomputed inverse needed for the Montgomery algorithm can be easily formed together with the modulus as:

$$m' = \begin{cases} \Delta_3 b^k - 1 & \text{if } m \in \mathrm{S}_3; \\ \Delta_4 b^k + 1 & \text{if } m \in \mathrm{S}_4. \end{cases}$$

In contrast to the modified Barrett algorithm, the Montgomery algorithm is easier to adopt to the special moduli, because it always computes $aR^{-1} \bmod m$, no matter what R is. The modified algorithm is shown in Alg.4.

Algorithm 4 Modular reduction without precomputation based on Montgomery algorithm.

INPUT: positive integers $a = (a_{2n-1} \ldots a_0)_b$, $m \in \mathrm{S}_3 \cup \mathrm{S}_4$, m', $R = b^n$.
OUTPUT: $t = aR^{-1} \bmod m$.
$s \Leftarrow (a \bmod R)m' \bmod R$
$t \Leftarrow (a + ms)/R$,
CORRECTION STEP:
if $t \geq m$ **then**
$t \Leftarrow t - m$
end if
RETURN: t.

IV. On The Security Of The Proposed Sets

As we can see from Figs. 1 and 2, one half of the modulus (either the most or the least significant half) is always filled with all 0's or all 1's. In the current state of the art, the security of ECC/HECC over prime fields GF(p) does not depend at all on the precise structure of the prime p. Hence, choosing a prime from the proposed sets has no influence on the security of ECC/HECC.

The security analysis for the RSA corresponds to attacks on RSA with partially known factorization. This problem has been analyzed extensively in the literature and the best attacks all rely on variants of Coppersmith's algorithm [8]. 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 (p and q). If half of the bits of either p or q (or both) are known, then N can be factored in polynomial time. Hence, it is important to stress here that, due to the Coppersmith's method of factoring, this type of integer should not be used as a prime factor for the RSA modulus.

V. Conclusion

The four distinct sets of primes, for which the precomputational step in modular reduction algorithms can be excluded, are introduced in this work. Choosing the modulus from the proposed sets gives more flexibility when implementing a high-speed modular multiplication algorithms. The proposed modular reduction methods are very suitable for fast hardware implementations of some public-key cryptosystems and in particular of Elliptic Curve Cryptography.

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.

References

[1] 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.

[2] N. Koblitz. Elliptic curve cryptosystem. *Math. Comp.*, 48:203–209, 1987.

[3] National Institute of Standards and Technology. FIPS 186-2: Digital Signature Standard, January 2000.

[4] M. Knežević, K. Sakiyama, J. Fan, and I. Verbauwhede. Modular Reduction in GF(2^n) Without Pre-Computational Phase. In *International Workshop on the Arithmetic of Finite Fields*, LNCS, page 11. Springer–Verlag, 2008.

[5] 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.

[6] P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170):519–521, 1985.

[7] J.-F. Dhem. *Design of an efficient public-key cryptographic library for RISC-based smart cards*. PhD Thesis, 1998.

[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.