Double-Speed Barrett Moduli

Rémi Géraud, Diana Maimuţ, David Naccache

Double-Speed Barrett Moduli

Rémi Géraud, Diana Maimuţ, and David Naccache

École normale supérieure
Équipe de cryptographie, 45 rue d'Ulm, F-75230 Paris CEDEX 05, France
given_name.family_name@ens.fr

Abstract. Modular multiplication and modular reduction are the atomic constituents of most public-key cryptosystems. Amongst the numerous algorithms for performing these operations, a particularly elegant method was proposed by Barrett. This method builds the operation a mod b from bit shifts, multiplications and additions in $\mathbb{Z}$. This allows to build modular reduction at very marginal code or silicon costs by leveraging existing hardware or software multipliers.

This paper presents a method allowing to double the speed of Barrett's algorithm by using specific composite moduli. This is particularly useful for lightweight devices where such an optimization can make a difference in terms of power consumption, cost and processing time. The generation of composite moduli with a predetermined portion is a well-known technique and the use of such moduli is considered, *in statu scientiæ*, as safe as using randomly generated composite moduli.

1 Introduction

Modular multiplication and modular reduction are the atomic constituents of most public-key cryptosystems. Amongst the numerous algorithms for performing these operations (*e.g.* [3, 4, 9, 12]), a particularly elegant method was proposed by Barrett in [1]. This method assembles the operation a mod b from bit shifts, multiplications and additions in $\mathbb{Z}$. This allows to build modular reduction at very marginal code or silicon costs by leveraging existing hardware or software multipliers. For a very detailed comparison of the principal modular reduction strategies, we refer the reader to [3].

This paper presents a method allowing to double the speed of Barrett's algorithm by using specific composite moduli. This is particularly useful for lightweight devices where such an optimization can make a difference in terms of power consumption, cost and processing time. The generation of composite moduli with a predetermined portion is a well-known technique [6, 10, 17] and the use of such moduli is considered, *in statu scientiæ*, as safe as using randomly generated composite moduli.

Related work: Douguet and Dupaquis [5] describe a modified Barrett modular reduction algorithm whose purpose is the acceleration of this type of operation

in certain (elliptic curve) groups of known moduli. Thus, the approach they consider implies moduli with a given form, *e.g.* the recommended ones from [13]. Estimations of the speed-ups are not provided, but the resistance of various architectures to different physical attacks is discussed. A general form of the Barrett constant and of the quotients (when certain moduli are used) are described. As an example of the proposed techniques, the Elliptic Curve Digital Signature Algorithm (ECDSA) [14] is taken into account.

We stress that no specific modulus generation algorithm is presented in [5]. The approach of [5] is rather a practical one, whereas our goal is to provide formal mathematical models for moduli with a predetermined portion generation.

Knežević, Batina and Verbauwhede [7] propose two sets of moduli for which Barrett's modular reduction algorithm can be implemented by avoiding the pre-computation of the Barrett constant. The types of moduli considered throughout this paper do not fall into those sets.

Structure of the paper: Section 2 starts by introducing notations and describing Barrett's original algorithm. Section 3 recalls background concerning composite moduli a predetermined portion. Section 4 introduces our core idea, that leverages Section 3 to generate Barrett-friendly RSA moduli. In Section 5, we apply this idea to other cryptographic primitives, such as DSA [14].

2 Barrett's Algorithm

For a given a, let $\|a\| = 1 + \lfloor \log_2 a \rfloor = \lceil \log_2 (a+1) \rceil$. That is, $\|a\|$ will denote the bit-length of a throughout this paper. $a|b$ will represent the concatenation of the bit-strings a and b.

$x \gg y$ will denote binary shift-to-the-right of x by y places *i.e.*:

$$x \gg y = \left\lfloor \frac{x}{2^y} \right\rfloor$$

Barrett's algorithm (Algorithm 1) approximates the result $c = d \bmod n$ by a quasi-reduced number $c + \epsilon n$ where $0 \leq \epsilon \leq 2$. We denote $N = \|n\|, D = \|d\|$ and set a *maximal bit-length reduction capacity* L such that $N \leq D \leq L$. The algorithm will function as long as $D \leq L$. In most implementations $D = L = 2N$. The algorithm uses the pre-computed constant $\kappa = \lfloor 2^L / n \rfloor$ that depends only on n and L. The reader is referred to [1] for a proof and a thorough analysis of this algorithm.

Work Factor: $\|c_1\| = D - N + 1 \simeq D - N$ and $\|\kappa\| = L - N$ hence their product requires $w = (D-N)(L-N)$ elementary operations. $\|c_3\| = (D - N) + (L - N) - (L - N + 1) = D - N - 1 \simeq D - N$. The product nc_3 will therefore claim $w' = (D-N)N$ elementary operations. All in all, work amounts to $w + w' = (D-N)(L-N) + (D-N)N = (D-N)L$. The goal of this paper is to halve this work factor.

Algorithm 1: Barrett's Algorithm

Input: $n < 2^N, d < 2^D, \kappa = \left\lfloor \frac{2^L}{n} \right\rfloor$ where $N \leq D \leq L$
Output: $c = d \bmod n$

$c_1 \leftarrow d \gg (N-1)$;
$c_2 \leftarrow c_1 \kappa$;
$c_3 \leftarrow c_2 \gg (L-N+1)$;
$c_4 \leftarrow d - nc_3$;
while $c_4 \geq n$ **do**
  $c_4 \leftarrow c_4 - n$;
end while
return c_4

3 Moduli with a Predetermined Portion

RSA [15] moduli with a predetermined portion are used to reduce storage requirements or computations. As mentioned before, such moduli are presently not known to be cryptographically weaker than randomly chosen ones. The first techniques for generating composite moduli were proposed by Vanstone and Zuccherato [17] who presented various ways of specifying $N/4 \leq \ell \leq N/2$ bits of n. Lenstra [10] proposed more advanced techniques for specifying up to $N/2$ bits. Based on Lenstra's algorithms, Joye proposed new techniques in [6]. Further works in the area include, for instance, [8, 11, 16]. We will hereafter recall the folklore method described by Joye (Algorithm 2), that perfectly fits our purpose[1].

Folklore method. The purpose of the folklore technique recalled by Joye is to obtain an RSA modulus n with a predetermined leading part n_h. Letting $\|n_h\| = H$, we have:

$$n = n_h 2^{N-H} + n_\ell, \text{ for some } 0 < n_\ell < 2^{N-H} \tag{1}$$

The algorithm uses the function NextPrime(x) that returns the prime following x (if x is prime then $x = \text{NextPrime}(x)$). Note that because the gap between x and NextPrime(x) is unpredictable, the algorithm may fail to return an n of the form $n = n_h 2^{N-H} + n_\ell$ and will have to be re-launched. We refer the reader to [10] for a more formal analysis of this process.

Lemma 1 (Bounding n and ω). *Consider the parameters used in Algorithm 2 and let $m = q - \omega$. Then, $n < n_h 2^{N-H} + (1+m)(2^{N-H} - 1)$ and $\omega < 2^{H+1} + 1$.*

Proof. By definition:

$$\omega = \left\lceil \frac{\eta}{p} \right\rceil \Rightarrow \exists \alpha < p \text{ such that } \omega = \frac{\eta}{p} + \frac{\alpha}{p}$$

[1] For the sake of clarity we remove all tests meant to enforce the condition $\text{GCD}(e, \phi(n)) = 1$.

τ	0	1	2	3	4
$\|\bar{n}_h\|$	503	502	501	500	499
success rate	85.66%	97.96%	99.96%	100%	100%

Fig. 1. Success rates of Algorithm 2 for $N = 1024, H = 512$ and 10^4 experiments.

Substituting the value of η, we get:

$$\omega = \frac{n_h 2^{N-H}}{p} + \frac{\alpha}{p} \Rightarrow q = \omega + m = \frac{n_h 2^{N-H}}{p} + \frac{\alpha}{p} + m$$

Thus:

$$n = pq = n_h 2^{N-H} + \alpha + mp < n_h 2^{N-H} + (1+m)p < n_h 2^{N-H} + (1+m)(2^{N-H} - 1)$$

And upper bounding ω we get:

$$\omega < \frac{\eta}{p} + 1 = \frac{n_h 2^{N-H}}{p} + 1 < \frac{n_h 2^{N-H}}{2^{N-H-1}} + 1 = 2n_h + 1 < 2^{H+1} + 1$$

Note that the most significant bit of p must be set to 1, *i.e.* $2^{N-H-1} < p < 2^{N-H} - 1$. □

It follows directly from Lemma 1 that:

$$q = \text{NextPrime}[\omega] \leq \text{NextPrime}[2^{H+1} + 1].$$

Applying the Prime Number Theorem, we find that $m \simeq \ln(2^{H+1} + 1) \simeq 0.7(H + 1)$. In other words, the $\log_2(m + 1) \simeq \log_2(0.7H + 1.7) < \log_2 H$ least significant bits of n_h are likely to get polluted. We hence rectify the size of n_h to $H - \tau - \log_2 H$ where $\tau \in \mathbb{N}$ is a parameter allowing to reduce the failure probability of Algorithm 2 at the cost of further shortening n_h. For the sake of clarity, we do not integrate these fine-tunings in the description of Algorithm 2 but consider that n_h is composed of a "real" prescribed pattern $\bar{n}_h$ of size $H - \tau - \lceil \log_2 H \rceil$ bits right-padded with $\tau + \lceil \log_2 H \rceil$ zero bits. Various success rates for $N = 1024, H = 512$ are given in Figure 1. Based on those we recommend to set $\tau = 0$ or $\tau = 1$ and re-launch the generation process if the algorithm fails.

Note: The algorithm's theoretical analysis could be simplified and the failure rate improved if step (4) of Figure 1 is replaced by: "If ω is composite then goto 1; else $q \leftarrow \omega$". The quality of the generated primes will also become theoretically uniform because NextPrime favors primes p_i whose distance from the previous prime p_{i-1} is large. This modification will, however, come at the cost of more computation time. The same note is applicable to Algorithm 3 as well.

Algorithm 2: Folklore method

Input: $N, H \leq N/2, n_h < 2^H$
Output: $n = n_h 2^{N-H} + n_\ell$, such that $0 < n_\ell < 2^{N-H}$

Generate a random prime p, such that $2^{N-H-1} < p < 2^{N-H} - 1$;
$\eta \leftarrow n_h 2^{N-H}$;
$\omega \leftarrow \left\lceil \frac{\eta}{p} \right\rceil$;
$q \leftarrow \text{NextPrime}(\omega)$;
$n \leftarrow pq$;
return n

Algorithm 3: Barrett-friendly modulus generator

Input: $L = 2N = 4U$
Output: n, an RSA modulus such that $2^{N-1} < n < 2^{N-1} + (0.7U + 2)(2^U - 1)$ whose associated κ is such that $2^{N+1} - 2^{U+1}(1 + 0.7U) < \kappa < 2^{N+1}$

Generate a random integer r such that $2^{U-1} < r < 2^U - 1$;
$\eta \leftarrow 2^{N-1} + r$;
Generate a random prime p such that $2^{U-1} < p < 2^U - 1$;
$\omega \leftarrow \left\lceil \frac{\eta}{p} \right\rceil$;
$q \leftarrow \text{NextPrime}(\omega)$;
$n \leftarrow pq$;
return n

4 Barrett-Friendly Moduli

We note that both multiplications in Algorithm 1 are multiplications by constants. Namely by n and κ. It is known (*e.g.* [2]) that multiplications by constants can be performed faster than multiplications by arbitrary integers. Our goal is to generate ① a composite n ② whose leading bits do not need to be multiplied and ③ whose associated κ also features a most significant part that does not need to be multiplied. As for the least significant parts of n and κ, these are constants and can hence *independently* benefit of speedup techniques such as [2]. The algorithm is given for the very common setting $L = D = 2N$. For convenience we introduce a bitlength unit U such that $L = 2N = 4U$.

Example 1. Let $N = 100$ and $L = 200$:

$r =$ `1ace38e78e29f`
$p =$ `322a28626f0a7`
$q =$ `51a6acec7fcd5`
$\eta =$ `8000000000001ace38e78e29f`
$\omega =$ `28d356763fe4a`
$n =$ `80000000000a8c93071ac14d9`
$\kappa =$ `1fffffffffd5cdb3e394fe440`

Lemma 2. *If* $0 < x < 2^{P/2-1}$, *then* $\left\lfloor \frac{2^{2P}}{2^{P-1}+x} \right\rfloor = 2^{P+1} - 4x$.

Proof. Observe that:

$$\frac{2^{2P}}{2^{P-1}+x} - (2^{P+1} - 4x) = \frac{4x^2}{2^{P-1}+x}. \tag{2}$$

Furthermore,

$$\frac{4x^2}{2^{P-1}+x} < 1 \Leftrightarrow 4x^2 - x < 2^{P-1}$$

This is a polynomial of degree 2, that has one positive and one negative root. We assumed $x > 0$, therefore we only need to consider the positive root $x_{\max}$:

$$x_{\max} = \frac{1}{8}\left(1 + \sqrt{1 + 2^{P+4}}\right) > 2^{P/2-1}$$

Therefore, if $x < 2^{P/2-1}$, then the fraction in eq. (2) is smaller than one. As a consequence, we have

$$\left\lfloor \frac{2^{2P}}{2^{P-1}+x} - (2^{P+1} - 4x) \right\rfloor = \left\lfloor \frac{2^{2P}}{2^{P-1}+x} \right\rfloor - (2^{P+1} - 4x) = 0,$$

as $2^{P+1} - 4x$ is an integer. □

Lemma 3 (Bounding n, ω and κ in Algorithm 3). *Consider the parameters used in Algorithm 3 and let $m = q - \omega$. Then, $n < 2^{N-1} + (2+m)(2^U - 1)$, $2^{N+1} - 2^{U+1}(1+m) < \kappa < 2^{N+1}$ and $\omega < 2^U + 2$.*

Proof. By definition:

$$\omega = \left\lceil \frac{\eta}{p} \right\rceil, \text{ thus } \exists \alpha < p \text{ such that } \omega = \frac{\eta}{p} + \frac{\alpha}{p}.$$

Substituting the value of η, we get:

$$\omega = \frac{2^{N-1} + r}{p} + \frac{\alpha}{p} \Rightarrow q = \omega + m = \frac{2^{N-1}}{p} + \frac{r}{p} + \frac{\alpha}{p} + m.$$

Thus:

$$n = pq = 2^{N-1} + r + \alpha + mp$$

$$\Downarrow$$

$$n < 2^{N-1} + r + (1+m)p < 2^{N-1} + 2^U - 1 + (1+m)(2^U - 1) < 2^{N-1} + (2+m)(2^U - 1).$$

We observe that

$$2^{N-1} + r + \alpha + mp \leq 2^{N-1} + r + mp \Rightarrow \frac{1}{2^{N-1} + r + \alpha + mp} \geq \frac{1}{2^{N-1} + r + mp}.$$

Bounding κ we obtain:

$$\kappa = \left\lfloor \frac{2^L}{n} \right\rfloor > \frac{2^L}{n} - 1 \geq \frac{2^L}{2^{N-1} + r + mp} - 1,$$

Now observe that $r + mp < 2^{N-1}$, therefore we can write

$$\frac{2^L}{2^{N-1} + r + mp} = \frac{2^{2N}}{2^{N-1} + r + mp} = 2^{N+1} \frac{1}{1 + 2^{1-N}(r + mp)} = 2^{N+1} \sum_{\ell=0}^{\infty} (-2)^{\ell(1-N)} (r+mp)^{\ell}$$

This series is convergent, alternating, and the term is strictly decreasing, therefore its sum is bounded below (resp. above) by the partial sum of odd (resp. even) degree S_ℓ. As a consequence,

$$\kappa > S_1 - 1 = 2^{N+1} \left(1 - 2^{1-N}(r + pm)\right) - 1 = 2^{N+1} - 4(r+pm) - 1 > 2^{N+1} - 2^{U+1}(1+m).$$

We observe that

$$2^{N-1} + r + \alpha + mp > 2^{N-1} \Rightarrow \frac{1}{2^{N-1} + r + \alpha + mp} < \frac{1}{2^{N-1}}.$$

Thus:

$$\kappa \leq \frac{2^L}{n} = \frac{2^L}{2^{N-1} + r + \alpha + mp} < \frac{2^L}{2^{N-1}} < 2^{N+1}.$$

Upper bounding ω we get:

$$\omega < \frac{\eta}{p} + 1 = \frac{2^{N-1} + r}{p} + 1 < \frac{2^{N-1} + 2^{U-1}}{2^{U-1}} + 1 = 2^{N-1-U+1} + 1 + 1 < 2^U + 2.$$

Note that the most significant bit of p must be set to 1, *i.e.* $2^{U-1} < p < 2^U - 1$. □

It follows directly from Lemma 3 that:

$$q = \text{NextPrime}[\omega] \leq \text{NextPrime}[2^U + 2] = \text{NextPrime}[2^U + 1].$$

Let n_h denote the predetermined portion of n, *i.e.* $n_h = 2^{U-1}$. Applying the Prime Number Theorem, we obtain $m \simeq \ln(2^U + 1) \simeq 0.7U$. Put differently, the $\log_2(m + 2) \simeq \log_2(0.7U + 2) < \log_2 U$ least significant bits of n_h are likely to get polluted. We hence rectify the size of n_h to $U - \tau - \log_2 U$ where $\tau \in \mathbb{N}$ is a parameter allowing to reduce the failure probability of Algorithm 3 at the cost of further shortening n_h. For the sake of clarity, we do not integrate these fine-tunings in the description of Algorithm 3 but consider that n_h is composed of a "real" prescribed pattern $\bar{n}_h$ of size $U - \tau - \lceil \log_2 U \rceil$ bits right-padded with $\tau + \lceil \log_2 U \rceil$ zero bits. Various success rates for $N = 1024, U = 512$ are given in Figure 2. Based on those we recommend to set $\tau = 0$ or $\tau = 1$ and re-launch the generation process if the algorithm fails.

It is easy to see that multiplication by both n and κ is not costly at all. To be more specific, n and κ satisfy the inequalities:

$$2^{N-1} < n < 2^{N-1} + (0.7U + 2)(2^U - 1) \text{ and } 2^{N+1} - 2^{U+1}(1 + 0.7U) < \kappa < 2^{N+1}.$$

As a result, this can double the speed of Barrett reduction[2].

[2] A few more complexity bits can be grabbed if the variant described in the note at the end of section 3 is used.

τ	0	1	2	3	4
$\|\bar{n}_h\|$	503	502	501	500	499
success rate	85.16%	97.51%	99.91%	100%	100%

Fig. 2. Success rates of Algorithm 3 for $N = 1024, U = 512$ and 10^4 experiments.

Algorithm 4: DSA prime generation

Input: Key lengths P and $Q \leq P$.
Output: Parameters (p, q).

1 Choose a $Q-$bit prime q;
2 Choose a $P-$bit prime modulus p such that $p-1$ is a multiple of q;
3 **return** (p, q)

5 Extensions

The parameter generation phase of DL cryptosystems requires the generation of two primes (*e.g.* p and q). Computations modulo these two primes represent important steps within the algorithms. Thus, a modular reduction speedup is necessary. It is thus desirable that both p and q to contain significantly long patterns (*i.e.* many successive 1s or 0s). We will now propose a Barrett-friendly parameter generation approach to do so. For the sake of clarity, we choose a particular algorithm to describe our method: the Digital Signature Algorithm (DSA).

5.1 Barrett-Friendly DSA Parameters Generation

DSA's parameter generation is presented in Algorithm 4. For the complete description of the DSA, we refer the reader to [14].

We suggest a modified DSA prime generation process leveraging the idea of Section 4. The procedure is described in Algorithm 5.

Lemma 4 (Structure of κ_q). *Let κ_q be the κ associated to q. With the notations of Algorithm 5, we have $\kappa_q = 2^{Q+1} - 4\omega$, assuming that $\omega < 2^{\frac{Q}{2}-1}$.*

Proof. Let $z = \frac{p-1}{q}$ and $\omega = q - 2^{Q-1}$. We observe that $\|z\| = P - Q$ and $q = 2^{Q-1}|\omega$. By definition, $\kappa_q = \left\lfloor \frac{2^{L_q}}{q} \right\rfloor$, where $L_q = 2Q$. As we assumed $\omega < 2^{\frac{Q}{2}-1}$, using Lemma 2 we have:

$$\kappa_q = \left\lfloor \frac{2^{L_Q}}{q} \right\rfloor = \left\lfloor \frac{2^{2Q}}{2^{Q-1} + \omega} \right\rfloor = 2^{Q+1} - 4\omega.$$

□

Algorithm 5: Barrett-friendly DSA prime generation

Input: Key lengths P and $Q \leq P$.
Output: Parameters (p, q).

Generate a $Q-$bit prime as follows:
$q \leftarrow \text{NextPrime}(2^{Q-1})$;
Construct a $P-$bit prime modulus p such that $p-1$ is a multiple of q in the following way:
$p \leftarrow 4$;
$i \leftarrow 1$;
$F \leftarrow 2^{P-Q-1}$;
while p is composite **do**
    $p \leftarrow 2q(F+i)+1$;
    $i++$;
end while
return (p, q)

The key consequence of Lemma 4 is that κ_q consists of a long pattern concatenated to a short different sequence, with a predetermined portion that is the complement of $q_h = 2^{Q-\Omega}$. The computation of κ_q is easy.

Let $L_p = 2P$. By definition, $\kappa_p = \left\lfloor \frac{2^{L_p}}{p} \right\rfloor$.

Lemma 5. *Let* $m(n) = \frac{1}{8}\left(n + \sqrt{n^2 + 2^{P+3}n}\right)$. *Let* x *be a positive integer such that* $0 < x < 2^{P-1}$ *and* $m(n) \leq x < m(n+1)$. *Then,*

$$\left\lfloor \frac{2^{2P}}{2^{P-1}+x} \right\rfloor = 2^{P+1} - 4x + n \quad \textit{and} \quad 0 \leq n < 2^P.$$

Proof. The proof consists of writing the fraction as a geometric series:

$$\begin{aligned}
\kappa = \left\lfloor \frac{2^{2P}}{2^{P-1}+x} \right\rfloor &= \left\lfloor 2^{P+1} \sum_{n=0}^{\infty} (-x)^n 2^{n(1-P)} \right\rfloor \\
&= \left\lfloor 2^{P+1}\left(1 - 2^{1-P}x + 2^{2-2P}x^2 - 2^{3-3P}x^3 + \dots\right) \right\rfloor \\
&= \left\lfloor 2^{P+1} - 4x + 2^{3-P}x^2 - 2^{4-2P}x^3 + \dots \right\rfloor
\end{aligned}$$

Now, $2^{P+1} - 4x$ is always a positive integer, it can therefore be safely taken out of the floor function. None of the remaining terms of the sum is an integer. We have:

$$\kappa = 2^{P+1} - 4x + \left\lfloor \sum_{n=2}^{\infty} (-x)^n 2^{n(1-P)} \right\rfloor.$$

The rightmost term is essentially a sum of shifted versions of powers of x. If x is small, then this contribution quickly vanishes. We now provide an exact value

for this sum, by rewriting:

$$\begin{aligned}\kappa &= 2^{P+1} - 4x + \left\lfloor 2^{2-P} x^2 \frac{2^{P-1}}{2^{P-1}+x} \right\rfloor \\ &= 2^{P+1} - 4x + \left\lfloor \frac{4x^2}{2^{P-1}+x} \right\rfloor .\end{aligned}$$

For any positive integer n, we have:

$$\frac{4x^2}{2^{P-1}+x} = n \Leftrightarrow x = \frac{1}{8}\left(n + \sqrt{n^2 + 2^{P+3}n}\right).$$

We assumed $x > 0$, thus we only need to consider the positive root. The leftmost fraction is a strictly increasing function of x as its derivative is > 0. Therefore, the rightmost formula strictly increases with n.
Let $m(n) = \frac{1}{8}\left(n + \sqrt{n^2 + 2^{P+3}n}\right)$ and assume that $m(n) \leq x < m(n+1)$. Then, we have:

$$n \leq \frac{4x^2}{2^{P-1}+x} < n+1$$

Therefore:

$$\left\lfloor \frac{4x^2}{2^{P-1}+x} \right\rfloor = n.$$

Finally, $x < 2^{P-1}$ implies an upper bound on the value of n, which must therefore be smaller than 2^P.

An illustrative example for $P = 1024$ and $Q = 160$ is given next.

Example 2.

$\omega = 299$

$i_p = 1$

$L_q = 2 \cdot 160$

$q = 2^{159} + 299$

$\kappa_q = 2^{163} - 4 \cdot 299$

$L_p = 2 \cdot 1024 = 2^{11}$

$p = (2^{864} + 2)q + 1 = (2^{864} + 2)(2^{159} + 299) + 1$

$x = 2^{60} + 299 \cdot 2^{864} + 2 \cdot 299 + 1$

$\kappa_p = 2^{71} \sum_{k=0}^{5} 2^{159k}(-299)^{6-k} - 2^{162} + 2387$

Thus, multiplication by p, q, κ_p and κ_q is easy.

References

1. P. Barrett. Implementing the Rivest, Shamir and Adleman Public-Key Encryption Algorithm on a Standard Digital Signal Processor. In *Advances in Cryptology: Proceedings of Crypto '86*, Lecture Notes in Computer Science, Vol. 263, Springer, pp. 311–323, 1987.
2. R. Bernstein. Multiplication by Integer Constants. In *Software - Practice and Experience*, 16(7), pp. 641–652, 1986.
3. A. Bosselaers, R. Govaerts, and J. Vandewalle. Comparison of Three Modular Reduction Functions. In *Advances in Cryptology: Proceedings of Crypto '93*, Lecture Notes in Computer Science, Vol. 773, Springer, pp. 175–186, 1994.
4. E. F. Brickell. A Fast Modular Multiplication Algorithm with Applications to Two Key Cryptography. In *Advances in Cryptology: Proceedings of Crypto '82*, Plenum, pp. 51–60, 1983.
5. M. Douguet and V. Dupaquis. Modular Reduction Using a Special Form of the Modulus. In *U.S. Patent Application 12/033,512*, filed February 19, Atmel Corporation, 2008.
6. M. Joye. RSA Moduli with a Predetermined Portion: Techniques and Applications. In *Information Security Practice and Experience*, Lecture Notes in Computer Science, Vol. 4991, Springer, pp. 116–130, 2008.
7. M. Knežević, L. Batina, I. Verbauwhede. Modular Reduction without Precomputational Phase. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, pp. 1389–1392, 2009.
8. H.-J. Knobloch. A Smart Card Implementation of the Fiat-Shamir Identification Scheme. In *Advances in Cryptology: Proceedings of Eurocrypt '88*, Lecture Notes in Computer Science, Vol. 330, Springer, pp. 87–95, 1988.
9. D. E. Knuth. The Art of Computer Programming. In *Volume 2, Seminumerical Algorithms*, 2nd Edition, Addison Wesley, Reading, Mass., 1981.
10. A. K. Lenstra. Generating RSA Moduli with a Predetermined Portion. In *Advances in Cryptology: Proceedings of Asiacrypt '98*, Lecture Notes in Computer Science, Vol. 1514, Springer, pp. 1–10, 1998.
11. G. Meister. On an Implementation of the Mohan-Adiga Algorithm. In *Advances in Cryptology: Proceedings of Eurocrypt '90*, Lecture Notes in Computer Science, Vol. 473, Springer, pp. 496–500, 1991.
12. P. L. Montgomery. Modular Multiplication Without Trial Division. In *Mathematics of Computation*, Vol. 44, No. 170, pp. 519–521, 1985.
13. National Institute of Standards and Technology (NIST). Digital Signature Standard. In *FIPS PUB 186-2*, 2013.
14. National Institute of Standards and Technology (NIST). Digital Signature Standard. In *FIPS PUB 186-4*, 2013.
15. R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In *Communications of the Association for Computing Machinery*, 21(2), pp. 120–126, 1978.
16. I. E. Shparlinski. On RSA Moduli with Prescribed Bit Patterns. In *Designs, Codes and Cryptography*, 39(1), pp. 113–122, 2006.
17. S. A. Vanstone and R. J. Zuccherato. Short RSA Keys and Their Generation. In *Journal of Cryptology*, 8(2), pp. 101–114, 1995.