Journal of Cryptographic Engineering (2021) 11:369–397
https://doi.org/10.1007/s13389-020-00254-3

REGULAR PAPER

Timing attacks and local timing attacks against Barrett's modular multiplication algorithm

Johannes Mittmann[1] · **Werner Schindler**[1]

Received: 19 May 2020 / Accepted: 20 December 2020 / Published online: 3 February 2021

Abstract
Montgomery's and Barrett's modular multiplication algorithms are widely used in modular exponentiation algorithms, e.g. to compute RSA or ECC operations. While Montgomery's multiplication algorithm has been studied extensively in the literature and many side-channel attacks have been detected, to our best knowledge no thorough analysis exists for Barrett's multiplication algorithm. This article closes this gap. For both Montgomery's and Barrett's multiplication algorithm, differences of the execution times are caused by conditional integer subtractions, so-called extra reductions. Barrett's multiplication algorithm allows even two extra reductions, and this feature increases the mathematical difficulties significantly. We formulate and analyse a two-dimensional Markov process, from which we deduce relevant stochastic properties of Barrett's multiplication algorithm within modular exponentiation algorithms. This allows to transfer the timing attacks and local timing attacks (where a second side-channel attack exhibits the execution times of the particular modular squarings and multiplications) on Montgomery's multiplication algorithm to attacks on Barrett's algorithm. However, there are also differences. Barrett's multiplication algorithm requires additional attack substeps, and the attack efficiency is much more sensitive to variations of the parameters. We treat timing attacks on RSA with CRT, on RSA without CRT, and on Diffie–Hellman, as well as local timing attacks against these algorithms in the presence of basis blinding. Experiments confirm our theoretical results.

Keywords Timing attacks · Local timing attacks · Barrett modular multiplication · RSA · RSA-CRT · Diffie–Hellman · Stochastic modelling · Statistical decision theory

1 Introduction

In his famous pioneer paper [19], Kocher introduced timing analysis. Two years later, [13] presented a timing attack on an early version of the Cascade chip. Both papers attacked unprotected RSA implementations which did not apply the Chinese remainder theorem (CRT). While in [19], the execution times of the particular modular multiplications and squarings are at least approximately normally distributed, this is not the case for the implementation in [13] since the Cascade chip applied the wide-spread Montgomery multiplication algorithm [21]. Due to conditional integer subtractions (so-called extra reductions), the execution times can only attain two values, and the probability whether an extra reduction occurs depends on the preceding Montgomery operations within the modular exponentiation. This fact caused substantial additional mathematical difficulties.

In [24], the random behaviour of the occurrence of extra reductions within a modular exponentiation was studied. The random extra reductions were modelled by a non-stationary time-discrete stochastic process. The analysis of this stochastic process (combined with an efficient error detection and correction strategy) allowed to drastically reduce the sample size, i.e. the number of timing measurements, namely from 200,000 to 300,000 [13] down to 5000 [28].

The analysis of the above-mentioned stochastic process turned out to be very fruitful also beyond this attack scenario. First, the insights into the probabilistic nature of the occurrence of extra reductions within modular exponentiations enabled the development of a completely new timing attack against RSA with CRT and Montgomery's multiplication algorithm [22]. This attack was extended to an attack on the sliding-window-based RSA implementation in OpenSSL

✉ Werner Schindler
werner.schindler@bsi.bund.de

Johannes Mittmann
johannes.mittmann@bsi.bund.de

[1] Bundesamt für Sicherheit in der Informationstechnik (BSI), Godesberger Allee 185–189, 53175 Bonn, Germany

v.0.9.7b [8], which caused a patch. The efficiency of this attack (in terms of the sample size) was increased by a factor of ≈ 10 in [3]. Years later, it was shown that exponent blinding (cf. [19], Sect. 10) does not suffice to prevent this type of timing attack [26,27].

Moreover, in [2,14,23] local timing attacks were considered. There, a side-channel attack (e.g. a power attack or an instruction cache attack) is carried out first, which yields the execution times of the particular Montgomery operations. This plus of information (compared to 'pure' timing attacks) allows to overcome basis blinding (a.k.a. message blinding, cf. [19], Sect. 10), and the attack works against both RSA with CRT and RSA without CRT. We mention that [2] led to a patch of OpenSSL v.0.9.7e.

Barrett's (modular) multiplication algorithm (a.k.a. Barrett reduction) [4] is a well-known alternative to Montgomery's algorithm. It is described in several standard manuals covering RSA, Diffie–Hellman (DH) or elliptic curve cryptosystems (e.g. [10,20]). The efficiency (e.g. running time) of Barrett's algorithm compared to Montgomery's algorithm has been analysed for both software implementations [6] and hardware implementations [18]. However, to our knowledge, there do not exist thorough security evaluations of Barrett's multiplication algorithm. In this paper, we close this gap. For the sake of comparison with previous work on Montgomery's algorithm, we focus again on RSA with and without CRT. In addition, we cover static DH, which can be handled almost identically to RSA without CRT.

Similar to Montgomery's algorithm, timing differences in Barrett's multiplication algorithm are caused by conditional subtractions (so-called extra reductions), which suggests to apply similar mathematical methods. However, for Barrett's algorithm, the mathematical challenges are significantly greater. One reason is that more than one extra reduction may occur. In particular, in place of a stochastic process over $\{0, 1\}$, a two-dimensional Markov process over $[0, 1) \times \{0, 1, 2\}$ has to be analysed and understood. Again, probabilities can be expressed by multidimensional integrals over $[0, 1)^\ell$, but the integrands are less suitable for explicit computations than in the Montgomery case. This causes additional numerical difficulties in particular for the local attacks, where ℓ is usually very large. Our results show many parallels to the Montgomery case, and after suitable modifications, all the known attacks on Montgomery's algorithm can be transferred to Barrett's multiplication algorithm. However, there are also significant differences. First of all, for Barrett's multiplication algorithm the attack efficiency is very sensitive to deviations of the modulus (i.e. of the RSA primes p_1 and p_2 if the CRT is applied), and attacks on RSA with CRT require additional attack steps.

The paper is organized as follows: in Sect. 2, we study the stochastic behaviour of the execution times of Barrett's multiplication algorithm in the context of the square & multiply exponentiation algorithm. We develop, prove and collect results, which will be needed later to perform the attacks. In Sect. 3, properties of Montgomery's and Barrett's multiplication algorithms are compared, and furthermore, a variant of Barrett's algorithm is investigated. In Sect. 4, the particular attacks are described and analysed, while Sect. 5 provides experimental results which confirm the theoretical considerations. Interesting in its own right is also an efficient look-ahead strategy. Finally, Sect. 6 discusses countermeasures.

2 Stochastic modelling of modular exponentiation

In this section, we analyse the stochastic timing behaviour of modular exponentiation algorithms when Barrett's multiplication is applied. We consider a basic version of Barrett's multiplication algorithm (cf. Algorithm 1). The slightly optimized version of this algorithm due to Barrett [4] will be discussed in Sect. 3.2.2 (cf. Algorithm 5), where we show that the same analysis applies, albeit with additional algorithmic noise. Since we assume that Steps 1 to 3 of Algorithm 1 run in constant time for fixed modulus length and base (see Justification of Assumption 2 in Sect. 3.2.1), we focus on the stochastic behaviour of the number of extra reductions in Algorithm 1. This knowledge will be needed in Sect. 4 when we consider concrete attacks. In Sect. 2.1, we investigate the Barrett's multiplication algorithm in isolation, and in Sect. 2.2, we use our results to study the square & multiply exponentiation algorithm if Barrett's multiplication is applied. This approach can be transferred to table-based exponentiation algorithms. Finally, we aim at equivalent results, which are already known for Montgomery's multiplication algorithm. We reach this goal, but the technical difficulties are significantly larger than they are for Montgomery's multiplication algorithm (cf. Sect. 3.1). In Sect. 2.3, we summarize the facts that are relevant for the attacks in Sect. 4. This allows to skip the technical Sects. 2.1 and 2.2 during the first reading of this paper.

2.1 Barrett's modular multiplication algorithm

In this subsection, we study the basic version of Barrett's modular multiplication algorithm (cf. Algorithm 1).

Definition 1 Let $\mathbb{N} := \{0, 1, \dots\}$. For $M \in \mathbb{N}_{\geq 2}$, let $\mathbb{Z}_M := \{0, 1, \dots, M-1\}$. Given $x \in \mathbb{Z}$, we denote by $x \bmod M$ the unique integer in $\mathbb{Z}_M$ congruent to x modulo M, i.e. $x \bmod M = x - \lfloor x/M \rfloor M$. We define the fractional part of a real number $x \in \mathbb{R}$ by $\{x\} := x - \lfloor x \rfloor \in [0, 1)$.

Let $M \in \mathbb{N}_{\geq 2}$ be a modulus of length $k = \lfloor \log_b M \rfloor + 1$ in base $b \in \mathbb{N}_{\geq 2}$, i.e. we have $b^{k-1} \leq M \leq b^k - 1$. The multipli-

cation modulo M of two integers $x, y \in \mathbb{Z}_M$ can be computed by an integer multiplication, followed by a modular reduction. The resulting remainder is $r := (x \cdot y) \bmod M = z - qM$, where $z := xy$ and $q := \lfloor z/M \rfloor$. The computation of q is the most expensive part, because it involves an integer division. The idea of Barrett's multiplication algorithm is to approximate q by

$$\widetilde{q} := \left\lfloor \frac{\lfloor z/b^{k-1} \rfloor \cdot \lfloor b^{2k}/M \rfloor}{b^{k+1}} \right\rfloor \lesssim \left\lfloor \frac{(z/b^{k-1}) \cdot (b^{2k}/M)}{b^{k+1}} \right\rfloor = q.$$

If the integer reciprocal $\mu := \lfloor b^{2k}/M \rfloor$ of M has been precomputed and if b is a power of 2, then $\widetilde{q}$ can be computed using only multiplications and bit shifts, which on common computer architectures are cheaper operations than divisions. From $\widetilde{q}$ an approximation $\widetilde{r} := z - \widetilde{q}M$ of r can be obtained. Since $\widetilde{q}$ can be smaller than q, it may be necessary to correct $\widetilde{r}$ by some conditional subtractions of M, which we call extra reductions. This leads to Algorithm 1.

Algorithm 1 Barrett modular multiplication.

Input: A modulus $M \in \mathbb{N}_{\geq 2}$ of length $k = \lfloor \log_b M \rfloor + 1$ in base $b \in \mathbb{N}_{\geq 2}$, the integer reciprocal $\mu = \lfloor b^{2k}/M \rfloor$ of M, and integers $x, y \in \mathbb{Z}_M$.
Output: $(x \cdot y) \bmod M$.

```
z ← x · y
q̃ ← ⌊⌊z/b^{k-1}⌋ μ / b^{k+1}⌋
r ← z − q̃ · M
while r ≥ M do
    r ← r − M                          ▷ Extra reduction
end while
return r
```

Algorithm 2 Left-to-right binary (a.k.a. square & multiply) exponentiation.

Input: A modulus $M \in \mathbb{N}_{\geq 2}$, a basis $y \in \mathbb{Z}_M$, and an exponent $d = (d_{\ell-1}, \ldots, d_0)_2 \in \mathbb{N}_{\geq 1}$ with $d_{\ell-1} = 1$.
Output: $y^d \bmod M$.

```
x ← y
for i ← ℓ − 2, ..., 0 do
    x ← x^2 mod M
    if d_i = 1 then
        x ← (x · y) mod M
    end if
end for
return x
```

Barrett showed that at most two extra reductions are required in Algorithm 1. The following lemma provides an exact characterization of the number of extra reductions and is at the heart of our subsequent analysis. In particular, the lemma identifies two important constants $\alpha \in [0, 1)$ and $\beta \in (b^{-1}, 1]$ associated with M and b.

Lemma 1 *On input $x, y \in \mathbb{Z}_M$, the number of extra reductions carried out in Algorithm 1 is*

$$\left\lceil \alpha \frac{x}{M} \frac{y}{M} + \beta \left\{ \frac{xy}{b^{k-1}} \right\} - \left\{ \frac{xy}{M} \right\} \right\rceil \in \mathbb{Z}_3, \tag{1}$$

where

$$\alpha := (M^2/b^{2k})\{b^{2k}/M\} \in [0, 1) \quad \textit{and}$$
$$\beta := \lfloor b^{2k}/M \rfloor / b^{k+1} \in (b^{-1}, 1].$$

Proof Set $z := xy, q := \lfloor z/M \rfloor$, and $\widetilde{q} := \lfloor \lfloor z/b^{k-1} \rfloor \mu / b^{k+1} \rfloor$. Since $z \bmod M = z - qM$, the number of extra reductions is

$$\begin{aligned}
q - \widetilde{q} &= \left\lfloor \frac{z}{M} \right\rfloor - \left\lfloor \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \frac{\mu}{b^{k+1}} \right\rfloor \\
&= \left\lceil \left\lfloor \frac{z}{M} \right\rfloor - \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \frac{\mu}{b^{k+1}} \right\rceil \\
&= \left\lceil \left(\frac{z}{M} - \left\{ \frac{z}{M} \right\} \right) - \left(\frac{z}{b^{k-1}} - \left\{ \frac{z}{b^{k-1}} \right\} \right) \frac{\mu}{b^{k+1}} \right\rceil \\
&= \left\lceil \frac{z}{b^{2k}} \left(\frac{b^{2k}}{M} - \mu \right) + \left\{ \frac{z}{b^{k-1}} \right\} \frac{\mu}{b^{k+1}} - \left\{ \frac{z}{M} \right\} \right\rceil \\
&= \left\lceil \frac{z}{b^{2k}} \left\{ \frac{b^{2k}}{M} \right\} + \left\{ \frac{z}{b^{k-1}} \right\} \frac{\mu}{b^{k+1}} - \left\{ \frac{z}{M} \right\} \right\rceil \\
&= \left\lceil \alpha \frac{z}{M^2} + \beta \left\{ \frac{z}{b^{k-1}} \right\} - \left\{ \frac{z}{M} \right\} \right\rceil.
\end{aligned}$$

Since $\alpha, \beta \in [0, 1]$, this number of extra reductions is in $\{0, 1, 2\}$. Finally, we note that $\lfloor b^{2k}/M \rfloor \geq \lfloor b^{2k}/(b^k - 1) \rfloor \geq b^k + 1$, hence $\beta > b^{-1}$. □

Remark 1 Note that $\alpha = 0$ if and only if M divides b^{2k}. In order to exclude this corner case (which is not relevant to our applications anyway), we assume $\alpha > 0$ for the remainder of this paper. Typically, $b = 2^{\mathsf{ws}}$ for some word size $\mathsf{ws} \geq 1$ and $\alpha = 0$ can only happen if M is a power of two, and then, modular multiplication is easy anyway. More special cases of α and β will be discussed in Sect. 3.2.3.

We first study the distribution of the number of extra reductions which are needed in Algorithm 1 for random inputs. To this end, we introduce the following stochastic model. Random variables are denoted by capital letters, and realizations of these random variables (i.e. values taken on by these random variables) are denoted with the corresponding small letters.

Stochastic Model 1 Let $s, t \in [0, 1)$. We define the random variable

$$R(s, t) := \lceil \alpha st + \beta U - V \rceil, \tag{2}$$

where U, V are independent, uniformly distributed random variables on $[0, 1)$.

A realization r of $R(s, t)$ expresses the random quantity of extra reductions which is required in Algorithm 1 for normalized inputs $x/M, y/M \in M^{-1}\mathbb{Z}_M$ within a small neighbourhood of s and t in $[0, 1)$.

Justification of Stochastic Model 1: Assume that $N(s)$ and $N(t)$ are small neighbourhoods of s and t in $[0, 1)$, respectively. Let $s_{\min}, s' \in N(s) \cap M^{-1}\mathbb{Z}_M$ such that $s_{\min}$ is minimal. Analogously, let $t_{\min}, t' \in N(t) \cap M^{-1}\mathbb{Z}_M$ such that $t_{\min}$ is minimal. Then, there are $m, n \in \mathbb{Z}$ such that $s' = s_{\min} + m/M$ and $t' = t_{\min} + n/M$. Then, $x' := Ms', x_{\min} := Ms_{\min}, y' := Mt', y_{\min} := Mt_{\min}$ are integers in $\mathbb{Z}_M$ such that

$$x'y' - x_{\min}y_{\min} \equiv my_{\min} + nx_{\min} + mn \pmod{M}.$$

The integers m and n assume values in

$$\{0, \ldots, |N(s) \cap M^{-1}\mathbb{Z}_M| - 1\} \text{ and}$$
$$\{0, \ldots, |N(t) \cap M^{-1}\mathbb{Z}_M| - 1\},$$

respectively. For cryptographically relevant modulus sizes, these numbers are very large so that one may assume that the admissible terms $\{(my_{\min} + nx_{\min} + mn) \bmod M\}$ are essentially uniformly distributed on $\mathbb{Z}_M$, justifying the model assumption that the random variable V is uniformly distributed on $[0, 1)$. The assumptions on the uniformity of U and the independence of U and V have analogous justifications. □

Remark 2 (i) In Stochastic Model 1 and Stochastic Model 2, we follow a strategy which has been very successful in the analysis of Montgomery's multiplication algorithm. Of course, the number of extra reductions needed for a particular computation $(xy) \bmod M$ is deterministic. On the other hand, by Lemma 1 the number of extra reductions only depends on the fact whether in (1) the sum within the brackets $\lceil\cdot\rceil$ is contained in $(-1, 0]$, $(0, 1]$ or $(1, 2)$. We exploit the fact that concerning the number of extra reductions $x, x' \in \mathbb{Z}_M$ have similar stochastic properties if $|x/M - x'/M|$ is small in $\mathbb{R}$. Furthermore, for moduli M that are used in cryptography even very small intervals in $[0, 1)$ contain a gigantic number of elements of $\mathbb{Z}_M/M$.

(ii) For extremely small input x or y of Algorithm 1, Stochastic Model 1 may not apply. In particular, if one of the inputs x, y is 0 or 1, the integer product $x \cdot y$ is already reduced modulo M; hence, $\widetilde{q} = 0$ and no extra reductions occur (cf. Remark 4).

Definition 2 For $x \in \mathbb{R}$, we write $(x)_+ := \max\{0, x\}$. Moreover, we set $(x)_+^0 := 1_{\{x \geq 0\}}$ and $(x)_+^n := ((x)_+)^n$ for $n \in \mathbb{N}_{>0}$.

Lemma 2 *Let $s, t \in [0, 1)$.*

(i) *The term $(an)^{-1}(ax + b)_+^n$ is an antiderivative of $(ax + b)_+^{n-1}$ for all $a, b \in \mathbb{R}$ with $a \neq 0$ and all $n \in \mathbb{N}_{>0}$.*

(ii) *Let $\mathscr{B}([0, 1))$ be the Borel σ-algebra on $[0, 1)$. For $r \in \mathbb{Z}_3$ and $B \in \mathscr{B}([0, 1))$, we have*

$$\Pr\big(R(s, t) \leq r, V \in B\big) = \beta^{-1} \int_B \big((r - \alpha st + v)_+ - (r - \alpha st - \beta + v)_+\big)\mathrm{d}v.$$

(iii) *For $r \in \mathbb{Z}_3$, we have*

$$\Pr\big(R(s, t) \leq r\big) = (2\beta)^{-1}\big(-(r - \alpha st)_+^2 + (r - \alpha st - \beta)_+^2 + (r - \alpha st + 1)_+^2 - (r - \alpha st - \beta + 1)_+^2\big).$$

(iv) *We have $\mathrm{E}\big(R(s, t)\big) = \alpha st + \beta/2$.*

(v) *We have $\mathrm{Var}\big(R(s, t)\big) = \alpha st + \beta/2 - (\alpha st + \beta/2)^2 + \beta^{-1}(\alpha st + \beta - 1)_+^2$.*

Proof For assertion (i), see, for example, [9]. Now let $r \in \mathbb{Z}_3$ and $v \in [0, 1)$. By (2), we have

$$\begin{aligned}\Pr\big(R(s, t) \leq r \mid V = v\big) &= \int_0^1 1_{\{\alpha st + \beta u - v \leq r\}}\mathrm{d}u \\ &= \int_0^1 (r - \alpha st - \beta u + v)_+^0\mathrm{d}u \\ &= \beta^{-1}\big((r - \alpha st + v)_+ - (r - \alpha st - \beta + v)_+\big).\end{aligned}$$

Now let $B \in \mathscr{B}([0, 1))$. Since V is uniformly distributed on $[0, 1)$, we obtain

$$\begin{aligned}\Pr\big(R(s, t) \leq r, V \in B\big) &= \int_B \Pr\big(R(s, t) \leq r \mid V = v\big)\mathrm{d}v \\ &= \beta^{-1}\int_B \big((r - \alpha st + v)_+ - (r - \alpha st - \beta + v)_+\big)\mathrm{d}v.\end{aligned}$$

Setting $B = [0, 1)$, we get

$$\begin{aligned}\Pr\big(R(s, t) \leq r\big) &= \beta^{-1}\int_0^1 \big((r - \alpha st + v)_+ - (r - \alpha st - \beta + v)_+\big)\mathrm{d}v \\ &= (2\beta)^{-1}\big(-(r - \alpha st)_+^2 + (r - \alpha st - \beta)_+^2 \\ &\quad + (r - \alpha st + 1)_+^2 - (r - \alpha st - \beta + 1)_+^2\big).\end{aligned}$$

Distinguishing the cases $\alpha st + \beta \leq 1$ and $\alpha st + \beta > 1$, the expectation and variance of $R(s, t)$ can be determined by careful but elementary computations. □

Lemma 3 *Let S be a uniformly distributed random variable on $[0, 1)$ and let $t \in (0, 1)$.*

(i) *For $r \in \mathbb{Z}_3$, we have*

$$\begin{aligned}\Pr\big(R(S,t) \le r\big) &= (6\alpha\beta t)^{-1}\big(-(r)_+^3 + (r-\alpha t)_+^3 + (r-\beta)_+^3 \\ &+ (r+1)_+^3 - (r-\alpha t-\beta)_+^3 - (r-\alpha t+1)_+^3 \\ &- (r-\beta+1)_+^3 + (r-\alpha t-\beta+1)_+^3\big).\end{aligned}$$

(ii) *We have* $\mathrm{E}\big(R(S,t)\big) = \alpha t/2 + \beta/2$.

(iii) *We have* $\mathrm{Var}\big(R(S,t)\big) = \alpha t/2 + \beta/2 - (\alpha t/2 + \beta/2)^2 + (3\alpha\beta t)^{-1}(\alpha t + \beta - 1)_+^3$.

Proof Let $r \in \mathbb{Z}_3$. We have

$$\Pr\big(R(S,t) \le r\big) = \int_0^1 \Pr\big(R(s,t) \le r\big)\mathrm{d}s.$$

By Lemma 2 (iii), we obtain

$$\begin{aligned}&\Pr\big(R(S,t) \le r\big) \\ &= (2\beta)^{-1}\int_0^1 \big(-(r-\alpha st)_+^2 + (r-\alpha st-\beta)_+^2 \\ &\quad + (r-\alpha st+1)_+^2 - (r-\alpha st-\beta+1)_+^2\big)\mathrm{d}s \\ &= (6\alpha\beta t)^{-1}\big(-(r)_+^3 + (r-\alpha t)_+^3 + (r-\beta)_+^3 \\ &\quad + (r+1)_+^3 - (r-\alpha t-\beta)_+^3 - (r-\alpha t+1)_+^3 \\ &\quad - (r-\beta+1)_+^3 + (r-\alpha t-\beta+1)_+^3\big).\end{aligned}$$

Distinguishing the cases $\alpha t + \beta \le 1$ and $\alpha t + \beta > 1$, the expectation and variance of $R(S,t)$ can be determined by careful but elementary computations. □

Lemma 4 *Let S be a uniformly distributed random variable on $[0,1)$.*

(i) *For $r \in \mathbb{Z}_3$, we have*

$$\begin{aligned}\Pr\big(R(S,S) \le r\big) &= (30\alpha^{1/2}\beta)^{-1}\big(-15r^2\min\{\alpha,(r)_+\}^{1/2} \\ &+ 10r\min\{\alpha,(r)_+\}^{3/2} \\ &- 3\min\{\alpha,(r)_+\}^{5/2} \\ &+ 15(r-\beta)^2\min\{\alpha,(r-\beta)_+\}^{1/2} \\ &- 10(r-\beta)\min\{\alpha,(r-\beta)_+\}^{3/2} \\ &+ 3\min\{\alpha,(r-\beta)_+\}^{5/2} \\ &+ 15(r+1)^2\min\{\alpha,(r+1)_+\}^{1/2} \\ &- 10(r+1)\min\{\alpha,(r+1)_+\}^{3/2} \\ &+ 3\min\{\alpha,(r+1)_+\}^{5/2} \\ &- 15(r-\beta+1)^2\min\{\alpha,(r-\beta+1)_+\}^{1/2} \\ &+ 10(r-\beta+1)\min\{\alpha,(r-\beta+1)_+\}^{3/2} \\ &- 3\min\{\alpha,(r-\beta+1)_+\}^{5/2}\big).\end{aligned}$$

(ii) *We have* $\mathrm{E}\big(R(S,S)\big) = \alpha/3 + \beta/2$.

(iii) *If $\alpha+\beta \le 1$, then* $\mathrm{Var}\big(R(S,S)\big) = \alpha/3 + \beta/2 - (\alpha/3 + \beta/2)^2$. *If $\alpha+\beta > 1$, then* $\mathrm{Var}\big(R(S,S)\big) = \alpha/3 + \beta/2 - (\alpha/3+\beta/2)^2 + (15\beta)^{-1}\big(15(1-\beta)^2 - 10\alpha(1-\beta) + 3\alpha^2\big) - 8(15\alpha^{1/2}\beta)^{-1}(1-\beta)^{5/2}$.

Proof Let $r \in \mathbb{Z}_3$. We have

$$\Pr\big(R(S,S) \le r\big) = \int_0^1 \Pr\big(R(s,s) \le r\big)\mathrm{d}s.$$

By Lemma 2 (iii), we obtain

$$\begin{aligned}&\Pr\big(R(S,S) \le r\big) \\ &= (2\beta)^{-1}\int_0^1 \big(-(r-\alpha s^2)_+^2 + (r-\alpha s^2-\beta)_+^2 \\ &\quad + (r-\alpha s^2+1)_+^2 - (r-\alpha s^2-\beta+1)_+^2\big)\mathrm{d}s.\end{aligned}$$

For $c \in \{r, r-\beta, r+1, r-\beta+1\}$, we have

$$\begin{aligned}\int_0^1 (c-\alpha s^2)_+^2 &= (15\alpha^{1/2})^{-1}\big(15c^2\min\{\alpha,(c)_+\}^{1/2} \\ &- 10c\min\{\alpha,(c)_+\}^{3/2} + 3\min\{\alpha,(c)_+\}^{5/2}\big).\end{aligned}$$

This implies (i). Distinguishing the cases $\alpha + \beta \le 1$ and $\alpha + \beta > 1$, the expectation and variance of $R(S,S)$ can be determined by careful but elementary computations. □

The number of extra reductions in Barrett's multiplication algorithm depends decisively on the parameters α and β. In Table 1, we illustrate this phenomenon with several numerical examples.

Remark 3 (i) For a random modulus M that is uniformly distributed on $\{b^{k-1}, \ldots, b^k - 1\}$, the parameters (α, β) can be modelled as realizations of the random vector $(A, B) := (X^2Y, (bX)^{-1})$, where X and Y denote independent random variables that are uniformly distributed on $[b^{-1}, 1)$ and $[0, 1)$, respectively. We have

$$\begin{aligned}\mathrm{E}(A) &= \frac{1}{2}\frac{1}{1-b^{-1}}\int_{b^{-1}}^1 x^2\mathrm{d}x = \frac{1}{6}(1+b^{-1}+b^{-2}), \\ \mathrm{E}(B) &= \frac{b^{-1}}{1-b^{-1}}\int_{b^{-1}}^1 x^{-1}\mathrm{d}x = \frac{\log(b)}{b-1}.\end{aligned}$$

For example, we get $\alpha \approx 0.29$ and $\beta \approx 0.69$ on average if $b = 2$.

(ii) By (2), two extra reductions can only occur if $\alpha + \beta > 1$ and the probability of this event increases for larger sums

$\alpha + \beta$. This sum can be bounded by

$$b^{-1} < \alpha + \beta \leq \frac{M^2}{b^{2k}} + \frac{b^{k-1}}{M} < 1 + b^{-1},$$

in particular α and β cannot attain their individual maxima simultaneously. For $b = 2$, we numerically determined $\Pr(A + B > 1) \approx 0.5$, which means that two extra reductions do occur for roughly one half of the moduli in this case.

(iii) Although the probability of two extra reductions can be very small or even 0, it does not simplify the stochastic representations (2) and (3) in Stochastic Model 1 and Stochastic Model 2. In particular, it does not simplify the analysis unless $\alpha \approx 0$ or $\beta \approx 0$ (cf. Sect. 3.2.3).

(iv) Although the impact of the extra reductions in terms of running time efficiency is only marginal for both Barrett's multiplication algorithm and Montgomery's multiplication algorithm, extra reductions are the source of several side-channel attacks (cf. Sect. 4).

2.2 Modular exponentiation (square and multiply algorithms)

Now we consider the left-to-right binary exponentiation algorithm (see Algorithm 2), where modular squarings and multiplications are performed using Barrett's algorithm (see Algorithm 1). Our goal is to define and analyse a stochastic process which allows to study the stochastic behaviour of the execution time of Algorithm 2. Sect. 2.2 provides a sequence of technical lemmata which be needed later.

Definition 3 Let $d \in \mathbb{N}_{>0}$. The binary representation of d is denoted by $(d_{\ell-1}, \ldots, d_0)_2$ with $d_{\ell-1} = 1$ and ℓ is called bit-length of d. The Hamming weight of d is defined as $\mathrm{ham}(d) := d_0 + \cdots + d_{\ell-1}$.

Let $y \in \mathbb{Z}_M$ be an input basis of Algorithm 2. We denote the intermediate values computed in the course of Algorithm 2 by $x_0, x_1, \ldots \in \mathbb{Z}_M$ and associate the sequence of squaring and multiplication operations with a string $\mathtt{O}_1\mathtt{O}_2\cdots$ over the alphabet $\{\mathtt{S}, \mathtt{M}\}$. For the sake of defining an infinite stochastic process, we assume that Algorithm 2 may run forever; hence, $x_0, x_1, \ldots$ is an infinite sequence and $\mathtt{O}_1\mathtt{O}_2\cdots \in \{\mathtt{S}, \mathtt{M}\}^\omega$. Consequently, we have $x_0 = y$ and

$$x_{n+1} = \begin{cases} x_n^2 \bmod M, & \text{if } \mathtt{O}_{n+1} = \mathtt{S}, \\ (x_n \cdot y) \bmod M, & \text{if } \mathtt{O}_{n+1} = \mathtt{M}, \end{cases}$$

for all $n \in \mathbb{N}$. Note that $\mathtt{O}_1\mathtt{O}_2\cdots$ does not contain the substring $\mathtt{MM}$. We will refer to strings $\mathtt{O}_1\mathtt{O}_2\cdots$ without substring $\mathtt{MM}$ as operation sequences.

Note that the square and multiply algorithm applied to an exponent d corresponds to a particular finite (d-specific) operation sequence $\mathtt{O}_1\mathtt{O}_2\cdots\mathtt{O}_{\ell+\mathrm{ham}(d)-2}$, where ℓ is the bit-length of d.

Stochastic Model 2 Let $t \in [0, 1)$ and let $\mathtt{O}_1\mathtt{O}_2\cdots \in \{\mathtt{S}, \mathtt{M}\}^\omega$ be an operation sequence. We define a stochastic process $(S_n, R_n)_{n\in\mathbb{N}}$ on the state space $\mathscr{S} := [0, 1) \times \mathbb{Z}_3$ as follows. Let $S_0, S_1, \ldots$ be independent random variables on $[0, 1)$. The random variable S_0 is uniformly distributed in $N(s_0) \cap M^{-1}\mathbb{Z}_M$, where $N(s_0)$ is a small neighbourhood of $s_0 = y/M$. Further, the random variables $S_1, S_2, \ldots$ are uniformly distributed random variables on $[0, 1)$, while R_0 is arbitrary (e.g. $R_0 = 0$), and for $n \in \mathbb{N}$ we define

$$R_{n+1} := \begin{cases} \lceil \alpha S_n^2 + \beta U_{n+1} - S_{n+1} \rceil, & \text{if } \mathtt{O}_{n+1} = \mathtt{S}, \\ \lceil \alpha S_n t + \beta U_{n+1} - S_{n+1} \rceil, & \text{if } \mathtt{O}_{n+1} = \mathtt{M}, \end{cases} \tag{3}$$

where $U_1, U_2, \ldots$ are independent, uniformly distributed random variables on $[0, 1)$.

The value t represents a normalized input y/M of Algorithm 2, realizations $s_0, s_1, \ldots$ of $S_0, S_1, \ldots$ represent the normalized intermediate values $x_0/M, x_1/M, \ldots \in M^{-1}\mathbb{Z}_M$, and a realization r_{n+1} of R_{n+1} represents the number of extra reductions which are necessary to compute x_{n+1} from x_n (and, additionally, from y if $\mathtt{O}_{n+1} = \mathtt{M}$).

Justification of Stochastic Model 2: This follows from the justification of Stochastic Model 1 by observing that, for inputs $x, y \in \mathbb{Z}_M$ of Algorithm 1, the term $\{xy/M\}$ in (1) equals the normalized value $\big((xy) \bmod M\big)/M$ of the product modulo M. As in Stochastic Model 1, we first conclude that U_1 and S_1 are uniformly distributed on $[0, 1)$. It follows by induction that $S_2, S_3, \ldots$ are uniformly distributed on $[0, 1)$, and $S_0, S_1, S_2, \ldots$ are independent. □

Remark 4 Remark 2 (ii) applies to the stochastic Stochastic Model 2 as well. In Sect. 4.3, we will adjust the stochastic process $(S_n, R_n)_{n\in\mathbb{N}}$ to a table-based exponentiation algorithm (fixed window exponentiation), where multiplications by 1 occur frequently. Multiplications by 1 will therefore be handled separately.

Lemma 5 *Let λ be the Lebesgue measure on the Borel σ-algebra $\mathscr{B}([0, 1))$, let η be the counting measure on the power set $\mathscr{P}(\mathbb{Z}_3)$ (i.e. $\eta(0) = \eta(1) = \eta(2) = 1$), and let $\lambda \otimes \eta$ be the product measure on the product σ-algebra $\mathscr{B}([0, 1)) \otimes \mathscr{P}(\mathbb{Z}_3)$.*

(i) *The stochastic process $(S_n, R_n)_{n\in\mathbb{N}}$ is a non-homogeneous Markov process on $\mathscr{S}$.*

(ii) *The random vector (S_n, R_n) has a density $f_n(s_n, r_n)$ with respect to $\lambda \otimes \eta$.*

Table 1 Numerical examples for the probabilities of extra reductions in Algorithm 1 for parameters α, β of several moduli M with base $b = 2$. Here, S and T denote independent random variables that are uniformly distributed on $[0, 1)$. We consider the cases of multiplication of a random input with a fixed input (with normalized value $t = 0.9$ and $t = 0.7$), squaring of a random input, and multiplication of two random inputs

α	β	$\alpha+\beta$	$\Pr(R(S,t=0.9)=r)$			$\Pr(R(S,t=0.7)=r)$			$\Pr(R(S,S)=r)$			$\Pr(R(S,T)=r)$		
			$r=0$	$r=1$	$r=2$	$r=0$	$r=1$	$r=2$	$r=0$	$r=1$	$r=2$	$r=0$	$r=1$	$r=2$
0.01	0.50	0.51	0.75	0.25	0.0	0.75	0.25	0.0	0.75	0.25	0.0	0.75	0.25	0.0
0.29	0.69	0.98	0.52	0.48	0.0	0.55	0.45	0.0	0.56	0.44	0.0	0.58	0.42	0.0
0.25	1.0	1.25	0.40	0.59	0.0082	0.42	0.58	0.0049	0.42	0.57	0.0061	0.44	0.55	0.0033
0.69	0.56	1.25	0.41	0.58	0.0028	0.48	0.52	0.0	0.49	0.50	0.0035	0.55	0.45	0.00066
0.99	0.50	1.49	0.33	0.65	0.023	0.41	0.59	0.0036	0.44	0.54	0.021	0.51	0.49	0.0056

(iii) *For $B \in \mathscr{B}([0,1))$, $r_{n+1} \in \mathbb{Z}_3$, and fixed $(s_n, r_n) \in \mathscr{S}$, we have*

$$\begin{aligned}\Pr(S_{n+1} \in B,\, R_{n+1} = r_{n+1} \mid S_n = s_n,\, R_n = r_n) &= \int_{B\times\{r_{n+1}\}} h_{n+1}(s_{n+1}, r \mid s_n)\mathrm{d}s_{n+1}\mathrm{d}\eta(r)\\ &= \int_B h_{n+1}(s_{n+1}, r_{n+1} \mid s_n)\mathrm{d}s_{n+1},\end{aligned}$$

where the conditional density is given by

$$\begin{aligned}&h_{n+1}(s_{n+1}, r_{n+1} \mid s_n)\\ &\quad= \beta^{-1}\big((r_{n+1} - \alpha s_n^2 + s_{n+1})_+\\ &\quad- (r_{n+1} - \alpha s_n^2 + s_{n+1} - \beta)_+\\ &\quad- (r_{n+1} - \alpha s_n^2 + s_{n+1} - 1)_+\\ &\quad+ (r_{n+1} - \alpha s_n^2 + s_{n+1} - \beta - 1)_+\big)\end{aligned}$$

if $\mathtt{O}_{n+1} = \mathtt{S}$ (i.e. the $(n+1)$-th operation is squaring) and

$$\begin{aligned}&h_{n+1}(s_{n+1}, r_{n+1} \mid s_n)\\ &\quad= \beta^{-1}\big((r_{n+1} - \alpha s_n t + s_{n+1})_+\\ &\quad- (r_{n+1} - \alpha s_n t + s_{n+1} - \beta)_+\\ &\quad- (r_{n+1} - \alpha s_n t + s_{n+1} - 1)_+\\ &\quad+ (r_{n+1} - \alpha s_n t + s_{n+1} - \beta - 1)_+\big)\end{aligned}$$

if $\mathtt{O}_{n+1} = \mathtt{M}$ (i.e. the $(n+1)$-th operation is multiplication by y).
In particular, the conditional density $h_{n+1}(s_{n+1}, r_{n+1} \mid s_n)$ does not depend on r_n. (This justifies the omission of r_n from the list of arguments.)

Proof Assertion (i) is an immediate consequence of the definition of Stochastic Model 2. To show (ii), let $C \subseteq [0,1)\times\mathbb{Z}_3$ be a $(\lambda\otimes\eta)$-zero set. Then, $C = \bigcup_{j\in\mathbb{Z}_3} B_j \times \{j\}$ (disjoint union) for measurable λ-zero sets B_0, B_1 and B_2. Hence, $\lambda(B_u) = 0$ for $B_u := B_0 \cup B_1 \cup B_2$. Finally, we get

$$\begin{aligned}\Pr((S_n, R_n) \in C) &\le \Pr((S_n, R_n) \in B_u \times \mathbb{Z}_3)\\ &= \Pr(S_n \in B_u) = \lambda(B_u) = 0.\end{aligned}$$

This shows that the pushforward measure of (S_n, R_n) is absolutely continuous with respect to $\lambda\otimes\eta$; therefore, assertion (ii) follows from the Radon–Nikodym theorem. Assertion (iii) follows from Lemma 2 (ii) with $(v, r) = (s_{n+1}, r_{n+1})$ and $(s, t) = (s_n, s_n)$ (if $\mathtt{O}_{n+1} = \mathtt{S}$) or $(s, t) = (s_n, t)$ (if $\mathtt{O}_{n+1} = \mathtt{M}$) and $V = S_{n+1}$. □

Lemma 6 (i) *For $r_{n+1} \in \mathbb{Z}_3$, we have*

$$\begin{aligned}&\Pr(R_{n+1} = r_{n+1})\\ &\quad= \int_{[0,1)\times\mathbb{Z}_3} \left(\int_0^1 h_{n+1}(s_{n+1}, r_{n+1} \mid s_n)\mathrm{d}s_{n+1}\right)\\ &\quad f_n(s_n, r_n)\mathrm{d}s_n\mathrm{d}\eta(r_n)\\ &\quad= \int_0^1 \left(\int_0^1 h_{n+1}(s_{n+1}, r_{n+1} \mid s_n)ds_{n+1}\right)\mathrm{d}s_n.\end{aligned}$$

In particular, the distribution of R_{n+1} does not depend on n but only on the operation type $\mathtt{O}_{n+1} \in \{\mathtt{S}, \mathtt{M}\}$.

(ii) *There exist real numbers $\mu_\mathtt{S}$, $\mu_\mathtt{M}$, $\sigma_\mathtt{S}^2$, $\sigma_\mathtt{M}^2 \in \mathbb{R}$ such that $\mathrm{E}(R_{n+1}) = \mu_{\mathtt{O}_{n+1}}$ and $\mathrm{Var}(R_{n+1}) = \sigma_{\mathtt{O}_{n+1}}^2$ for all $n \in \mathbb{N}$. In particular, we have*

$$\begin{aligned}\mu_\mathtt{S} &= \alpha/3 + \beta/2,\\ \mu_\mathtt{M} &= \alpha t/2 + \beta/2,\end{aligned} \tag{4}$$

$$\begin{aligned}\sigma_\mathtt{S}^2 &= \alpha/3 + \beta/2 - (\alpha/3 + \beta/2)^2, \quad \text{if } \alpha+\beta \le 1,\\ \sigma_\mathtt{S}^2 &= \alpha/3 + \beta/2 - (\alpha/3 + \beta/2)^2\\ &\quad + (15\beta)^{-1}\big(15(1-\beta)^2 - 10\alpha(1-\beta) + 3\alpha^2\big)\\ &\quad - 8(15\alpha^{1/2}\beta)^{-1}(1-\beta)^{5/2}, \quad \text{if } \alpha+\beta > 1,\end{aligned} \tag{5}$$

$$\begin{aligned}\sigma_\mathtt{M}^2 &= \alpha t/2 + \beta/2 - (\alpha t/2 + \beta/2)^2\\ &\quad + (3\alpha\beta t)^{-1}(\alpha t + \beta - 1)_+^3.\end{aligned} \tag{6}$$

The expectation $\mu_{\mathtt{M}}$ is strictly monotonously increasing in $t = y/M$.

Proof Let $r_{n+1} \in \mathbb{Z}_3$. Since

$$\begin{aligned}\Pr(R_{n+1} = r_{n+1}) &= \Pr\big((S_n, R_n) \in \mathscr{S}, (S_{n+1}, R_{n+1}) \in [0,1) \times \{r_{n+1}\}\big),\end{aligned}$$

the first equation of assertion (i) follows from Lemma 5. The second equation of (i) follows from the fact that

$$\sum_{r=0}^{2} f_n(s_n, r) = 1 \quad \text{for all } s_n \in [0,1),$$

because S_n is uniformly distributed on $[0, 1)$. Assertion (ii) is an immediate consequence of (i). The formulae (4), (5), and (6) follow from Lemma 4 (ii), Lemma 3 (ii), Lemma 4 (iii), and Lemma 3 (iii). The final assertion of (ii) is obvious since $\alpha > 0$. □

Definition 4 A sequence $X_1, X_2, \ldots$ of random variables is called m-dependent if the random vectors $(X_1, \ldots, X_u)$ and $(X_v, \ldots, X_n)$ are independent for all $1 \le u < v \le n$ with $v - u > m$.

Lemma 7 (i) *For $r_{n+1}, \ldots, r_{n+u} \in \mathbb{Z}_3$, we have*

$$\begin{aligned}&\Pr(R_{n+1} = r_{n+1}, \ldots, R_{n+u} = r_{n+u}) \\ &= \int_{[0,1)\times\mathbb{Z}_3} \Bigg(\int_0^1 \Bigg(\cdots \\ &\quad \int_0^1 h_{n+u}(s_{n+u}, r_{n+u} \mid s_{n+u-1}) \mathrm{d}s_{n+u} \cdots \Bigg) \\ &\quad h_{n+1}(s_{n+1}, r_{n+1} \mid s_n) \mathrm{d}s_{n+1} \Bigg) f_n(s_n, r_n) \mathrm{d}s_n \mathrm{d}\eta(r_n) \\ &= \int_0^1 \Bigg(\int_0^1 \Bigg(\cdots \\ &\quad \int_0^1 h_{n+u}(s_{n+u}, r_{n+u} \mid s_{n+u-1}) \mathrm{d}s_{n+u} \cdots \Bigg) \\ &\quad h_{n+1}(s_{n+1}, r_{n+1} \mid s_n) \mathrm{d}s_{n+1} \Bigg) \mathrm{d}s_n.\end{aligned}$$

In particular, the joint distribution of $R_{n+1}, \ldots, R_{n+u}$ does not depend on n but only on the operation types $\mathtt{O}_{n+1}, \ldots, \mathtt{O}_{n+u} \in \{\mathtt{S}, \mathtt{M}\}$.

(ii) *There exist real numbers $\mathrm{cov}_{\mathtt{SS}}, \mathrm{cov}_{\mathtt{SM}}, \mathrm{cov}_{\mathtt{MS}} \in \mathbb{R}$ such that $\mathrm{Cov}(R_n, R_{n+1}) = \mathrm{cov}_{\mathtt{O}_n\mathtt{O}_{n+1}}$ for all $n \in \mathbb{N}_{>0}$.*

(iii) *The sequence $R_1, R_2, \ldots$ is 1-dependent. In particular, we have $\mathrm{Cov}(R_n, R_{n+s}) = 0$ for all $s \ge 2$.*

Proof Let $r_{n+1}, \ldots, r_{n+u} \in \mathbb{Z}_3$. Then,

$$\begin{aligned}&\Pr(R_{n+1} = r_{n+1}, \ldots, R_{n+u} = r_{n+u}) \\ &= \Pr\big((S_n, R_n) \in \mathscr{S}, \\ &\quad (S_{n+i}, R_{n+i}) \in [0,1) \times \{r_{n+i}\} \text{ for } i = 1, \ldots, u\big)\end{aligned}$$

and assertion (i) follows from Lemma 5, Lemma 6 (i), and the Ionescu–Tulcea theorem. Assertion (ii) is an immediate consequence of (i). To prove (iii), let $1 \le u < v \le n$ such that $v - u > 1$, let $r_1, \ldots, r_u, r_v, \ldots, r_n \in \mathbb{Z}_3$, and define the events

$$\begin{aligned}E_1 &:= \big\{(S_0, R_0) \in \mathscr{S}\big\}, \\ E_2 &:= \big\{(S_i, R_i) \in [0,1) \times \{r_i\} \text{ for } i = 1, \ldots, u\big\}, \\ E_3 &:= \big\{(S_{v-1}, R_{v-1}) \in \mathscr{S}\big\}, \\ E_4 &:= \big\{(S_i, R_i) \in [0,1) \times \{r_i\} \text{ for } i = v, \ldots, n\big\}.\end{aligned}$$

Using the Markov property of $(S_n, R_n)_{n\in\mathbb{N}}$ and (i), we obtain

$$\begin{aligned}&\Pr(R_1 = r_1, \ldots, R_u = r_u, R_v = r_v, \ldots, R_n = r_n) \\ &= \Pr(E_1, E_2, E_3, E_4) \\ &= \Pr(E_1, E_2) \cdot \Pr(E_3 \mid E_1, E_2) \cdot \Pr(E_4 \mid E_1, E_2, E_3) \\ &= \Pr(E_1, E_2) \cdot 1 \cdot \Pr(E_4 \mid E_3) \\ &= \Pr(E_1, E_2) \cdot \Pr(E_3, E_4) \cdot \Pr(E_3)^{-1} \\ &= \Pr(E_1, E_2) \cdot \Pr(E_3, E_4) \\ &= \Pr(R_1{=}r_1, \ldots, R_u{=}r_u) \cdot \Pr(R_v{=}r_v, \ldots, R_n{=}r_n);\end{aligned}$$

hence, $(R_1, \ldots, R_u)$ and $(R_v, \ldots, R_n)$ are independent. We conclude that $R_1, R_2, \ldots$ is a 1-dependent sequence. □

Definition 5 The normal distribution with mean μ and variance σ^2 is denoted by $\mathscr{N}(\mu, \sigma^2)$, and

$$\Phi(x) := (2\pi)^{-1/2} \int_{-\infty}^{x} e^{-t^2/2} \mathrm{d}t$$

is the cumulative distribution function of $\mathscr{N}(0, 1)$.

For strings $x, y \in \{\mathtt{S}, \mathtt{M}\}^*$, we denote by $\#_x(y) \in \mathbb{N}$ the number of occurrences of x in y.

Below we will use the following version of the central limit theorem for m-dependent random variables due to Hoeffding & Robbins.

Lemma 8 (Cf. [17, Theorem 1]) *Let $X_1, X_2, \ldots$ be an m-dependent sequence of random variables such that $\mathrm{E}(X_i) = 0$ and $\mathrm{E}(|X_i|^3)$ is uniformly bounded for all $i \in \mathbb{N}_{>0}$. For $i \in \mathbb{N}_{>0}$ define $A_i := \mathrm{Var}(X_{i+m}) + 2\sum_{j=1}^{m} \mathrm{Cov}(X_{i+m-j}, X_{i+m})$. If the limit $A := \lim_{u\to\infty} u^{-1} \sum_{h=1}^{u} A_{i+h}$ exists uniformly for all $i \in \mathbb{N}$, then $(X_1 + \cdots + X_s)/\sqrt{s}$ has the limiting distribution $\mathscr{N}(0, A)$ as $s \to \infty$.*

Lemma 9 *Let $\mathtt{O}_1\mathtt{O}_2 \cdots \in \{\mathtt{S}, \mathtt{M}\}^\omega$ be an operation sequence such that the limit*

$$\rho := \lim_{u\to\infty} \frac{\#_{\mathtt{M}}(\mathtt{O}_{i+1} \cdots \mathtt{O}_{i+u})}{u} \tag{7}$$

exists uniformly for all $i \in \mathbb{N}$ and define

$$A := \rho \cdot \sigma_{\mathtt{M}}^2 + (1-\rho) \cdot \sigma_{\mathtt{S}}^2 + 2\rho \cdot \mathrm{cov}_{\mathtt{MS}} + 2\rho \cdot \mathrm{cov}_{\mathtt{SM}} + (2-4\rho) \cdot \mathrm{cov}_{\mathtt{SS}} .$$

Then, $\lim_{s\to\infty} \mathrm{Var}(R_{n+1} + \cdots + R_{n+s})/s = A$ *and*

$$\frac{R_{n+1} + \cdots + R_{n+s} - \mathrm{E}(R_{n+1} + \cdots + R_{n+s})}{\sqrt{s}}$$

has the limiting distribution $\mathcal{N}(0, A)$.

Proof Let $R'_i := R_i - \mathrm{E}(R_i)$ for all $i \in \mathbb{N}_{>0}$. Then, $R'_1, R'_2, \ldots$ is a 1-dependent sequence of random variables with $\mathrm{E}(R'_i) = 0$ and $\mathrm{E}(|R'_i|^3) \le 2^3 = 8$ for all i. Define $A_i := \mathrm{Var}(R'_{i+1}) + 2\,\mathrm{Cov}(R'_i, R'_{i+1})$. For $x \in \{\mathtt{S}, \mathtt{M}\}^*$ and $0 \le i \le j$, we set $\#_x^{i,j} := \#_x(\mathtt{O}_i \cdots \mathtt{O}_j)$. With this notation, we have

$$\sum_{h=1}^{u} A_{i+h} = \#_{\mathtt{M}}^{i+2,i+u+1}\sigma_{\mathtt{M}}^2 + \#_{\mathtt{S}}^{i+2,i+u+1}\sigma_{\mathtt{S}}^2 + 2\#_{\mathtt{MS}}^{i+1,i+u+1}\,\mathrm{cov}_{\mathtt{MS}} + 2\#_{\mathtt{SM}}^{i+1,i+u+1}\,\mathrm{cov}_{\mathtt{SM}} + 2\#_{\mathtt{SS}}^{i+1,i+u+1}\,\mathrm{cov}_{\mathtt{SS}} .$$

Using the identities

$$\begin{aligned}
\#_{\mathtt{MS}}^{i+1,i+u+1} &= \#_{\mathtt{M}}^{i+1,i+u},\\
\#_{\mathtt{S}}^{i+2,i+u+1} &= u - \#_{\mathtt{M}}^{i+2,i+u+1},\\
\#_{\mathtt{SM}}^{i+1,i+u+1} &= \#_{\mathtt{M}}^{i+2,i+u+1},\\
\#_{\mathtt{SS}}^{i+1,i+u+1} &= u - 1 - \#_{\mathtt{M}}^{i+1,i+u} - \#_{\mathtt{M}}^{i+2,i+u+1},
\end{aligned}$$

we obtain

$$\sum_{h=1}^{u} A_{i+h} = \#_{\mathtt{M}}^{i+2,i+u+1}\sigma_{\mathtt{M}}^2 + (u - \#_{\mathtt{M}}^{i+2,i+u+1})\sigma_{\mathtt{S}}^2 + 2\#_{\mathtt{M}}^{i+1,i+u}\,\mathrm{cov}_{\mathtt{MS}} + 2\#_{\mathtt{M}}^{i+2,i+u+1}\,\mathrm{cov}_{\mathtt{SM}} + 2(u - 1 - \#_{\mathtt{M}}^{i+2,i+u+1} - \#_{\mathtt{M}}^{i+1,i+u})\,\mathrm{cov}_{\mathtt{SS}} .$$

As $u \to \infty$, the ratio $\#_{\mathtt{M}}^{i+1,i+u}/u$ converges to ρ uniformly for all i by assumption; therefore, $u^{-1}\sum_{h=1}^{u} A_{i+h}$ converges to A uniformly for all i. Since

$$\mathrm{Var}(R'_{n+1} + \cdots + R'_{n+s}) = \mathrm{Var}(R'_{n+1}) + \sum_{h=1}^{s} A_{n+h} - A_{n+s},$$

$\mathrm{Var}(R'_{n+1} + \cdots + R'_{n+s})/s$ converges to A as $s \to \infty$. Finally, $(R'_{n+1} + \cdots + R'_{n+s})/\sqrt{s}$ has the limiting distribution $\mathcal{N}(0, A)$ by Lemma 8. □

We note that for random operation sequences $\mathtt{O}_1\mathtt{O}_2\cdots$ (corresponding to random exponents with independent and unbiased bits), the convergence of (7) is not uniform with probability 1. However, for any given finite operation sequence $\mathtt{O}_{n+1}\cdots\mathtt{O}_{n+s}$ we may construct an infinite sequence $\mathtt{O}_1\mathtt{O}_2\cdots$ with subsequence $\mathtt{O}_{n+1}\cdots\mathtt{O}_{n+s}$ for which convergence of (7) is uniform and $\rho \approx \#_{\mathtt{M}}(\mathtt{O}_{n+1}\cdots\mathtt{O}_{n+s})/s$. Therefore, if s is sufficiently large, it is reasonable to assume that the normal approximation

$$\Pr\left(\frac{R_{n+1} + \cdots + R_{n+s} - \mathrm{E}(R_{n+1} + \cdots + R_{n+s})}{\sqrt{\mathrm{Var}(R_{n+1} + \cdots + R_{n+s})}} \le x\right) \approx \Phi(x) \tag{8}$$

is appropriate. We mention that in our experiments in Sect. 5.1 approximation (8) is applied and leads to successful attacks.

2.3 Summary of the relevant facts

In this section, we studied the random behaviour of the number of extra reductions when Barrett's modular multiplication algorithm is used within the square and multiply exponentiation algorithm. In Sect. 4.3, we generalize this approach to table-based exponentiation algorithms.

We defined a stochastic process $(S_n, R_n)_{n\in\mathbb{N}}$. The random variable S_n represents the (random) normalized intermediate value (= intermediate value divided by the modulus M) in Algorithm 2 after the n-th Barrett operation, and the random variable R_n represents the (random) number of extra reductions needed for the n-th Barrett operation.

Algorithm 1 needs 0, 1 or 2 extra reductions. The stochastic process $(S_n, R_n)_{n\in\mathbb{N}}$ is a non-homogeneous Markov chain on the state space $\mathcal{S} = [0, 1) \times \mathbb{Z}_3$. The projection onto the first component gives independent random variables $S_1, S_2, \ldots$, which are uniformly distributed on the unit interval $[0, 1)$. However, we are interested in the stochastic process $R_1, R_2, \ldots$ on $\mathbb{Z}_3$, which is more difficult to analyse. In particular, $\mathrm{E}(R_n)$ and $\mathrm{Var}(R_n)$ depend on the operation type $\mathtt{O}_n$ of the n-th Barrett operation (multiplication $\mathtt{M}$ or squaring $\mathtt{S}$), while the covariances $\mathrm{Cov}(R_n, R_{n+1})$ depend on the operation types of the n-th and the $(n+1)$-th Barrett operation ($\mathtt{SM}$, $\mathtt{MS}$ or $\mathtt{SS}$). The formulae (4), (5) and (6) provide explicit formulae for the expectations and the variances, while Lemma 7 (i), (ii) explains how to compute the covariances. Further, the stochastic process $(R_n)_{n\in\mathbb{N}_{\ge 1}}$ is 1-dependent. In particular, a version of the central limit theorem for dependent random variables can be applied to approximate the distribution of standardized finite sums (cf. (8)).

3 Montgomery multiplication versus Barrett multiplication

In Sect. 3.1, we briefly treat Montgomery's multiplication algorithm (MM) [21] and summarize relevant stochastic properties. This is because in Sect. 4 we consider the question whether the (known) pure and local timing attacks against Montgomery's multiplication algorithm can be transferred to implementations that apply Barrett's algorithm.

3.1 Montgomery's multiplication algorithm in a nutshell

Montgomery's multiplication algorithm is widely used to compute modular exponentiations because it transfers modulo operations and divisions to moduli and divisors, which are powers of 2.

For an odd modulus M (e.g. an RSA modulus or a prime), the integer $R := 2^t > M$ is called Montgomery's constant, and $R^{-1} \in \mathbb{Z}_M$ denotes its multiplicative inverse modulo M. Moreover, $M^* \in \mathbb{Z}_R$ satisfies the integer equation $RR^{-1} - MM^* = 1$. Montgomery's algorithm computes

$$(a, b) \mapsto \mathrm{MM}(a, b; M) := abR^{-1} \bmod M$$

with a version of Algorithm 3. Here, ws denotes the word size of the arithmetic operations (typically, depending on the platform $\mathsf{ws} \in \{8, 16, 32, 64\}$), which divides the exponent t. Further, $r = 2^{\mathsf{ws}}$, so that $R = r^v$ with $v = t/\mathsf{ws}$. In Algorithm 3, the operands x, y and s are expressed in the r-adic representation. That is, $x = (x_{v-1}, ..., x_0)_r$, $y = (y_{v-1}, ..., y_0)_r$ and $s = (s_{v-1}, ..., s_0)_r$ with $r = 2^{\mathsf{ws}}$. Finally, $m^* = M^* \bmod r$. After Step 3 the intermediate value $s \equiv abR^{-1} \pmod M$ and $s \in [0, 2M)$. The instruction $s := s - M$ in Step 4, called 'extra reduction' (ER), is carried out iff $s \in [M, 2M)$. This conditional integer subtraction is responsible for timing differences, and thus is the source of side channel attacks.

Algorithm 3 Montgomery modular multiplication.

Input: An odd modulus $M \in \mathbb{N}_{\geq 2}$, Montgomery constant $R = r^v > M$ with $r = 2^{\mathsf{ws}}$, $m^* = M^* \bmod r$, and integers $x = (x_{v-1}, \ldots, x_0)_r, y = (y_{v-1}, \ldots, y_0)_r \in \mathbb{Z}_M$.
Output: $xyR^{-1} \bmod M$.

1: $s \leftarrow 0$
2: **for** $i \leftarrow 0, \ldots, v-1$ **do**
3: $\quad u \leftarrow (s + x_i y_0) m^* \bmod r$
4: $\quad s \leftarrow (s + x_i y + uM)/r$
5: **end for**
6: **if** $s \geq M$ **then**
7: $\quad s \leftarrow s - M$ ▷ Extra reduction
8: **end if**
9: **return** s ▷ $s = \mathrm{MM}(x, y; M)$

Algorithm 4 Montgomery left-to-right binary exponentiation.

Input: An odd modulus $M \in \mathbb{N}_{\geq 2}$, Montgomery constant $R > M$, integers $x, y \in \mathbb{Z}_M$, and an exponent $d = (d_{\ell-1}, \ldots, d_0)_2 \in \mathbb{N}_{\geq 1}$ with $d_{\ell-1} = 1$.
Output: $y^d \bmod M$.

1: $\widetilde{y} \leftarrow \mathrm{MM}(y, R; M)$ ▷ $\widetilde{y} \leftarrow yR \bmod M$
2: $\widetilde{x} \leftarrow \widetilde{y}$
3: **for** $i \leftarrow \ell - 2, \ldots, 0$ **do**
4: $\quad \widetilde{x} \leftarrow \mathrm{MM}(\widetilde{x}, \widetilde{x}; M)$
5: $\quad$ **if** $d_i = 1$ **then**
6: $\quad\quad \widetilde{x} \leftarrow \mathrm{MM}(\widetilde{x}, \widetilde{y}; M)$
7: $\quad$ **end if**
8: **end for**
9: $x \leftarrow \mathrm{MM}(\widetilde{x}, 1; M)$ ▷ $x \leftarrow \widetilde{x}R^{-1} \bmod M$
10: **return** x

Assumption 1 (Montgomery modular multiplication) For fixed modulus M and fixed Montgomery constant R,

$$\mathrm{Time}\big(\mathrm{MM}(a, b; M)\big) \in \big\{c, c + c_{\mathrm{ER}}\big\} \quad \text{for all } a, b \in \mathbb{Z}_M, \quad (9)$$

which means that an MM operation costs time c if no ER is needed, and c_{ER} equals the time for an ER. (The constants c and c_{ER} depend on the concrete implementation.)

Justification of Assumption 1: (See [26], Remark 1, for a comprehensive analysis.) For known-input attacks (with more or less randomly chosen inputs), Assumption 1 should usually be valid. An exception is pure timing attacks on RSA with CRT implementations in old versions of OpenSSL [3,7], cf. Sect. 4.2. The reason is that OpenSSL applies different subroutines to compute the for-loop in Algorithm 3, depending on whether x and y have identical word size or not. The before-mentioned timing attacks on RSA with CRT are adaptive chosen input attacks, and during the attack certain MM-operands become smaller and smaller. This feature makes the attack to some degree more complicated but does not prevent it because new sources for timing differences occur. RSA implementations on smart cards and microcontrollers usually should not care about word lengths (and meet Assumption 1 in any case) because in normal use operands with different word size rarely occur so that an optimization of this case seems to be useless. □

In the following, we summarize some well-known fundamental stochastic properties of Montgomery's multiplication algorithm, or more precisely, on the distribution of random extra reductions within a modular exponentiation algorithm. Their knowledge is needed to develop (effective and efficient) pure or local timing attacks [3,7,22–28].

We interpret the normalized intermediate values of Algorithm 4 as realizations of random variables $S_0, S_1, \ldots$. With the same arguments as in Sect. 2.2 (for Barrett's multiplication), one concludes that for Algorithm 4 the random variables $S_1, S_2, \ldots$ are iid uniformly distributed on $[0, 1)$. We set $w_i = 1$ if the i-th Montgomery operation requires

an ER and $w_i = 0$ otherwise. We interpret the values $w_1, w_2, \ldots$ as realizations of $\{0, 1\}$-valued random variables $W_1, W_2, \ldots$.

Interestingly, it does not depend on the word size ws whether an ER is necessary but only on (a, b, M, R). This allows to consider the case $\mathsf{ws} = t$ (i.e. $v = 1$) when analysing the stochastic behaviour of the random variables W_i in modular exponentiation algorithms. In particular, the computation of $\mathrm{MM}(a, b; M)$ requires an ER iff

$$\frac{\mathrm{MM}(a, b, M)}{M} < \frac{a}{M}\frac{b}{M}\frac{M}{R}. \tag{10}$$

This observation allows to express the random variable W_i in terms of S_{i-1} and S_i. For Algorithm 4, this implies

$$W_i = \begin{cases} 1_{\{S_i < S_{i-1}\frac{\tilde{y}}{M}\frac{M}{R}\}} & \text{if the } i-\text{th op. is mult. by } \widetilde{y}, \\ 1_{\{S_i < S_{i-1}^2\frac{M}{R}\}} & \text{if the } i-\text{th op. is squaring.} \end{cases} \tag{11}$$

The random variables $W_1, W_2, \ldots$ have interesting properties which are similar to those of $R_1, R_2, \ldots$. In particular, they are neither stationary distributed nor independent but 1-dependent and under weak assumption they fulfil a version of the central limit theorem for dependent random variables. Relation (11) allows to represent joint probabilities $\Pr(W_i = w_i, \ldots, W_{i+k-1} = w_{i+k-1})$ as integrals over the $(k+1)$-dimensional unit cube. We just note that

$$\Pr(W_i = 1) = \begin{cases} \frac{\tilde{y}}{2R} & \text{if the } i-\text{th op. is mult. by } \widetilde{y}, \\ \frac{M}{3R} & \text{if the } i-\text{th op. is squaring.} \end{cases} \tag{12}$$

3.2 A closer look at Barrett's multiplication algorithm

In this subsection, we develop and justify equivalents to Assumption 1 for two (different) Barrett multiplication algorithms (Algorithm 1 and Algorithm 5). Therefrom, we deduce stochastic representations, which describe the random timing behaviour of modular exponentiations $y \mapsto y^d \bmod M$.

3.2.1 Modular exponentiation with Algorithm 1

At first we formulate an equivalent to Assumption 1.

Assumption 2 (Barrett modular multiplication) For fixed modulus M,

$$\mathrm{Time}\big(\mathrm{BM}(a, b; M)\big) \in \{c, c + c_{\mathrm{ER}}, c + 2c_{\mathrm{ER}}\} \tag{13}$$

for all $a, b \in \mathbb{Z}_M$, which means that a Barrett multiplication (BM) costs time c if no ER is needed, and c_{ER} equals the time for one ER. (The constants c and c_{ER} depend on the concrete implementation.)

Justification of Assumption 2: The justification of Assumption 2 is rather similar to the justification of Assumption 1 in Sect. 3.1. In Line 1 of Algorithm 1, $x, y \in \mathbb{Z}_M$ are multiplied in $\mathbb{Z}$, and in Line 2 the rounding off brackets $\lfloor\cdot\rfloor$ are simple shift operations if b equals 2 or a suitable power of two (e.g. $b = 2^{\mathsf{ws}}$, where ws denotes the word size of the underlying arithmetic). For known-input attacks (with more or less randomly chosen inputs), and for smart cards and microcontrollers in general it is reasonable to assume that the Lines 1–3 cost identical time for all $x, y \in \mathbb{Z}_M$ and, consequently, that Assumption 1 is valid. Exceptions may exist for adaptive chosen input timing attacks on RSA implementations (cf. Sect. 4.2) on PCs, which use large general libraries. Even then it seems to be very likely that (as for Montgomery's multiplication algorithm) such optimizations allow timing attacks anyway. □

This leads to the following stochastic representation of the (random) timing of a modular exponentiation $y^d \bmod M$ if it is calculated with Algorithm 2.

Algorithm 2 & Algorithm 1:

$$\begin{aligned}\mathrm{Time}(y^d \bmod M) &= t_{\mathrm{set}} + (\ell + \mathrm{ham}(d) - 2)c \\ &\quad + (R_1 + \cdots + R_{\ell+\mathrm{ham}(d)-2})c_{\mathrm{ER}} + N.\end{aligned} \tag{14}$$

Here, the 'setup-time' t_{set} summarizes the time needed for all operations that are not part of Algorithm 1, e.g. the time needed for input and output and maybe for the computation of the constant μ (if not stored). The random variable N quantifies the 'timing noise'. This includes measurement errors and possible deviations from Assumption 2. We assume that $N \sim \mathcal{N}(0, \sigma^2)$. We allow $\sigma^2 = 0$, which means 'no noise' (i.e. $N \equiv 0$), while a nonzero expectation $\mathrm{E}(N)$ is 'moved' to t_{set}. The data-dependent timing differences are quantified by the stochastic process $R_1, R_2, \ldots, R_{\ell+\mathrm{ham}(d)-2}$, which is thoroughly analysed in Sect. 2. Recall that the distribution of this stochastic process depends on the secret exponent d and on the ratio $t = y/M$.

Remark 5 (i) Without blinding mechanisms $\mathrm{Time}(y^d \bmod M)$ is identical for repeated executions with the same basis y if we neglect possible measurement errors. At first sight, the stochastic representation (14) may be surprising but the stochastic process $R_1, R_2, \ldots$ describes the random timing behaviour for bases $y' \in \mathbb{Z}_M$ whose ratio y'/M is close to $t = y/M$ (cf. Stochastic Model 1).

(ii) A similar stochastic representation exists for table-based exponentiation algorithms, see Sect. 4.3.

3.2.2 Modular exponentiation with Algorithm 5

Algorithm 5 is a modification of Algorithm 1 containing an optimization which was already proposed by Barrett [4]. Its Lines 3-6 substitute Line 3 of Algorithm 1. We may assume

$b = 2^{\mathsf{ws}} > 2$, where ws is the word size of the integer arithmetic (typically, $\mathsf{ws} \in \{8, 16, 32, 64\}$). Line 3 of Algorithm 1 computes a multiplication $\widetilde{q} \cdot M$ of two integers, which are in the order of b^k, and a subtraction of two integers, which are in the order of b^{2k}. In contrast, Line 3 of Algorithm 5 only requires a modular multiplication $(\widetilde{q} \cdot M) \bmod 2^{\mathsf{ws}(k+1)}$, the subtraction of two integers in the order of b^{k+1} and Line 5 possibly one addition by b^{k+1}.

Algorithm 5 Barrett modular multiplication (optimized).

Input: A modulus $M \in \mathbb{N}_{\geq 2}$ of length $k = \lfloor \log_b M \rfloor + 1$ in base $b \in \mathbb{N}_{\geq 3}$, the integer reciprocal $\mu = \lfloor b^{2k}/M \rfloor$ of M, and integers $x, y \in \mathbb{Z}_M$.
Output: $(x \cdot y) \bmod M$.
1: $z \leftarrow x \cdot y$
2: $\widetilde{q} \leftarrow \lfloor \lfloor z/b^{k-1} \rfloor \mu / b^{k+1} \rfloor$
3: $r \leftarrow \left(z \bmod b^{k+1}\right) - \left((\widetilde{q} \cdot M) \bmod b^{k+1}\right)$
4: **if** $r < 0$ **then**
5: $\quad r \leftarrow r + b^{k+1}$ ▷ Extra addition
6: **end if**
7: **while** $r \geq M$ **do**
8: $\quad r \leftarrow r - M$ ▷ Extra reduction
9: **end while**
10: **return** r

After Line 6 of Algorithm 5, $r \equiv z - (\widetilde{q} \cdot M) \pmod{b^{k+1}}$, and further $0 \leq r < b^{k+1}$. Since $\lfloor \log_b M \rfloor \leq \log_b M < k = \lfloor \log_b M \rfloor + 1$, we have $b^{k-1} \leq M < b^k$, hence $3M < 3b^k \leq b^{k+1}$. By Lemma 1, the values r after Line 3 of Algorithm 1 and r after Line 6 of Algorithm 5 thus coincide, which means that the number of extra reductions is the same for both algorithms. If c_{add} denotes the time needed for an addition of b^{k+1} in Line 5 of Algorithm 5, this leads to an equivalent of Assumption 2.

Assumption 3 (**Barrett modular multiplication, optimized**) For fixed modulus M,

$$\begin{aligned}\text{Time}\big(\text{BM}(a, b; M)\big) \in \big\{&c, c + c_{\text{ER}}, c + 2c_{\text{ER}},\\ &c + c_{\text{add}}, c + c_{\text{ER}} + c_{\text{add}}, c + 2c_{\text{ER}} + c_{\text{add}}\big\}\end{aligned} \tag{15}$$

for all $a, b \in \mathbb{Z}_M$, which means that a Barrett multiplication (BM) with Algorithm 5 (without extra reductions or an extra addition by b^{k+1}) costs time c, while c_{ER} and c_{add} equal the time for one ER or for an addition by b^{k+1}, respectively. (The constants c, c_{ER} and c_{add} depend on the concrete implementation.) When implemented on the same platform, the constant c in (15) should be smaller than in (13).

Justification of Assumption 3: The justification is rather similar to that of Assumption 2. The relevant arguments have already been discussed at the beginning of this subsection. This also concerns the general expositions to the impact of possible optimizations of the integer multiplication algorithm. □

The if-condition in Line 4 of Algorithm 5 introduces an additional source of timing variability, which has to be analysed. We have already explained that this if-condition does not affect the number of extra reductions. Next, we determine the probability of an extra addition by b^{k+1}.

Let $x, y \in \mathbb{Z}_M$ and let $z = x \cdot y$. We denote by r_a the number of extra additions by b^{k+1} in the computation of $z \bmod M$. Obviously, $r_a = 1$ iff $z \bmod b^{k+1} < \widetilde{q}M \bmod b^{k+1}$ (and $r_a = 0$ otherwise), or equivalently, when dividing both terms by b^{k+1},

$$r_a = 1 \quad \text{iff} \quad \left\{\frac{z}{b^{k+1}}\right\} < \left\{\frac{\widetilde{q}M}{b^{k+1}}\right\}. \tag{16}$$

Next, we derive a more convenient characterization of (16). Let $\gamma := M/b^{k+1} \in (0, b^{-1})$. We have

$$\begin{aligned}\frac{\widetilde{q}M}{b^{k+1}} &= \left\lfloor \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \frac{\lfloor b^{2k}/M \rfloor}{b^{k+1}} \right\rfloor \frac{M}{b^{k+1}}\\
&= \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \frac{\lfloor b^{2k}/M \rfloor}{b^{k+1}} \frac{M}{b^{k+1}} - \left\{ \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \frac{\lfloor b^{2k}/M \rfloor}{b^{k+1}} \right\} \frac{M}{b^{k+1}}\\
&= \frac{z}{b^{k-1}} \frac{\lfloor b^{2k}/M \rfloor}{b^{k+1}} \frac{M}{b^{k+1}} - \left\{ \frac{z}{b^{k-1}} \right\} \frac{\lfloor b^{2k}/M \rfloor}{b^{k+1}} \frac{M}{b^{k+1}}\\
&\quad - \gamma \left\{ \beta \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \right\}\\
&= \frac{z}{b^{k-1}} \frac{b^{2k}/M - \{b^{2k}/M\}}{b^{k+1}} \frac{M}{b^{k+1}} - \beta\gamma \left\{ \frac{z}{b^{k-1}} \right\}\\
&\quad - \gamma \left\{ \beta \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \right\}\\
&= \frac{z}{b^{k+1}} - \frac{z}{M^2} \frac{M^2}{b^{2k}} \left\{ \frac{b^{2k}}{M} \right\} \frac{M}{b^{k+1}} - \beta\gamma \left\{ \frac{z}{b^{k-1}} \right\}\\
&\quad - \gamma \left\{ \beta \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \right\}\\
&= \frac{z}{b^{k+1}} - \alpha\gamma \frac{z}{M^2} - \beta\gamma \left\{ \frac{z}{b^{k-1}} \right\} - \gamma \left\{ \beta \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \right\}.\end{aligned}$$

Since $0 \leq \alpha\gamma(z/M^2) + \beta\gamma\{z/b^{k-1}\} + \gamma\{\beta\lfloor z/b^{k-1} \rfloor\} \leq 3\gamma < 3/b \leq 1$, we can rewrite characterization (16) as

$$\begin{aligned}&r_a = 1 \quad \text{iff}\\ &\left\{\frac{z}{b^{k+1}}\right\} < \alpha\gamma \frac{z}{M^2} + \beta\gamma \left\{ \frac{z}{b^{k-1}} \right\} + \gamma \left\{ \beta \left\lfloor \frac{z}{b^{k-1}} \right\rfloor \right\}.\end{aligned} \tag{17}$$

Our aim is to develop a stochastic model for the extra addition. We start with a closer inspection of the right-hand side of (17). Let $z = (z_{2k-1}, \ldots, z_0)_b$ be the b-ary representation

of z, where leading zero digits are permitted. Then,

$$\begin{aligned} u' &:= \{z/b^{k+1}\} = (0.z_k, \ldots, z_0)_b, \\ v_1 &:= z/M^2 = (z_{2k-1}, \ldots, z_0)_b/M^2, \\ v_2 &:= \{z/b^{k-1}\} = (0.z_{k-2}, \ldots, z_0)_b, \quad \text{and} \\ v_3 &:= \{\beta \lfloor z/b^{k-1} \rfloor\} \\ &= \{\lfloor b^{2k}/M \rfloor (0.z_{2k-1}, \ldots, z_{k-1})_b\}. \end{aligned} \tag{18}$$

We now assume that Algorithm 5 is applied in the modular exponentiation algorithm Algorithm 2. In analogy to the extra reductions, we interpret the number of extra additions by b^{k+1} in the $(n+1)$-th Barrett operation, denoted by $r_{a;n+1}$, as a realization of a $\{0, 1\}$-valued random variable $R_{a;n+1}$. In particular, $(x \cdot y) \bmod M$ either represents a squaring ($x = y$) or a multiplication of the intermediate value x by the basis y, respectively.

As in Stochastic Model 2, we model $v_1 = (xy)/M^2$ as a realization of S_n^2 (squaring) or $S_n t$ (multiplication by the basis y) with $t = y/M$, respectively, and v_2 as a realization of the random variable U_{n+1} which is uniformly distributed on $[0, 1)$. With the same argumentation as for v_2, we model $u'_{n+1} := u' = \{xy/b^{k+1}\}$ as a realization of a random variable U'_{n+1} that is uniformly distributed on $[0, 1)$.

It remains to analyse v_3. Let for the moment $(a_k, \ldots, a_0)_b$ be the b-ary representation of $\lfloor b^{2k}/M \rfloor$ (note that we have $b^k \le \lfloor b^{2k}/M \rfloor < b^{k+1}$ since we excluded the corner case $M = b^{k-1}$) and $a_{-1} = a_{-2} = 0$. Then,

$$\begin{aligned} v_3 &= \left\{ \sum_{j=0}^{k} a_j b^j \sum_{i=1}^{k+1} z_{2k-i} b^{-i} \right\} \\ &= \left\{ \sum_{h=-k-1}^{k-1} b^h \sum_{j-i=h} a_j z_{2k-i} \right\} \\ &\approx \left\{ \sum_{h=-3}^{-1} b^h \sum_{j-i=h} a_j z_{2k-i} \right\} \\ &= \frac{\sum_{i=1}^{k+1} \left(\sum_{h=-3}^{-1} a_{i+h} b^{3+h} \right) z_{2k-i} \bmod b^3}{b^3}. \end{aligned} \tag{19}$$

We model $v_{n+1} := v_3$ as a realization of a random variable V_{n+1} that is uniformly distributed on $[0, 1)$.

By (18) and (19), the values u', v_1, v_2 and v_3 essentially depend on z_k (resp., on (z_k, z_{k-1}) if ws is small), on the most significant digits of z in the b-ary representation, on z_{k-2} (resp. on (z_{k-2}, z_{k-3}) if ws is small) and on the weighted sum of the b-ary digits $z_{2k-1}, \ldots, z_{k-1}$, respectively. This justifies the assumption that the random variables U'_{n+1}, S_n, U_{n+1}, V_{n+1} (essentially) behave as if they were independent.

We could extend the non-homogeneous Markov chain $(S_n, R_n)_{n\in\mathbb{N}}$ on the state space $[0, 1) \times \mathbb{Z}_3$ from Sect. 2 to a non-homogeneous Markov chain $(S_n, R_n, R_{a;n})_{n\in\mathbb{N}}$ on the state space $[0, 1) \times \mathbb{Z}_3 \times \mathbb{Z}_2$. Its analysis is analogous to that in Sect. 2 but more complicated in detail. Since for typical word sizes w the impact of the extra additions on the execution time is by an order of magnitude smaller than the impact of the extra reductions (due to the factor γ, cf. (20) and (21)), we do not elaborate on this issue. We only mention that

$$\begin{aligned} &\Pr(R_{a;n+1} = 1) \\ &= \Pr(U'_{n+1} < \alpha\gamma S_n^2 + \beta\gamma U_{n+1} + \gamma V_{n+1}) \\ &= \int_{[0,1)^3} \Pr(U'_{n+1} < \alpha\gamma s_n^2 + \beta\gamma u_{n+1} + \gamma v_{n+1}) \\ &\quad \mathrm{d}v_{n+1}\mathrm{d}u_{n+1}\mathrm{d}s_n \\ &= \int_{[0,1)^3} (\alpha\gamma s_n^2 + \beta\gamma u_{n+1} + \gamma v_{n+1}) \mathrm{d}v_{n+1}\mathrm{d}u_{n+1}\mathrm{d}s_n \\ &= \alpha\gamma/3 + \beta\gamma/2 + \gamma/2 =: \mu_{a,\mathtt{S}}, \quad \text{if } \mathtt{O}_{n+1} = \mathtt{S}, \end{aligned} \tag{20}$$

and analogously

$$\begin{aligned} &\Pr(R_{a;n+1} = 1) \\ &= \Pr(U'_{n+1} < \alpha\gamma S_n t + \beta\gamma U_{n+1} + \gamma V_{n+1}) \\ &= \alpha\gamma t/2 + \beta\gamma/2 + \gamma/2 =: \mu_{a,\mathtt{M}}, \quad \text{if } \mathtt{O}_{n+1} = \mathtt{M}. \end{aligned} \tag{21}$$

For the reason mentioned above, we treat the extra additions as noise. Equation (22) is the equivalent to (14) for the modified version of Barrett's multiplication algorithm. We use the same notation as in (14).

Algorithm 2 & Algorithm 5:

$$\begin{aligned} \mathrm{Time}(y^d \bmod M) &= t^*_{\mathrm{set}} + (\ell + \mathrm{ham}(d) - 2)c \\ &\quad + (R_1 + \cdots + R_{\ell+\mathrm{ham}(d)-2})c_{\mathrm{ER}} + N^*, \end{aligned} \tag{22}$$

with $t^*_{\mathrm{set}} = t_{\mathrm{set}} + (\mu_{a,\mathtt{S}}(\ell - 1) + \mu_{a,\mathtt{M}}(\mathrm{ham}(d) - 1))c_{\mathrm{add}}$ and $N^* \sim \mathcal{N}(0, (\mu_{a,\mathtt{S}}(1 - \mu_{a,\mathtt{S}})(\ell - 1) + \mu_{a,\mathtt{M}}(1 - \mu_{a,\mathtt{M}})(\mathrm{ham}(d) - 1))c^2_{\mathrm{add}} + \sigma^2)$. The expected time for all extra additions has been moved to the setup time, and the variance became part of N^*. While the formula for the expectation is exact, we used a coarse approximation for the variance which neglects any dependencies. We just mention that R_{n+1} and $R_{a;n+1}$ are positively correlated. This follows from the fact that apart from the factor γ the terms '$\alpha\gamma S_n^2$', '$\alpha\gamma S_n t$', and '$\beta\gamma U_{n+1}$' in (20) and (21) coincide with terms in (3) and since $R_{a;n+1}$ and R_{n+1} are both 'large' if the corresponding terms in (20), (21) and (3) are 'large'.

Remark 6 (i) The stochastic representations (14) and (22) are essentially identical although (22) has slightly larger noise.

(ii) The ratio $c_{\mathrm{add}}/c_{\mathrm{ER}}$ depends on the implementation.

3.2.3 Special values for α and β

The stochastic behaviour of the Barrett multiplication algorithm depends on α and β. In particular, α has significant impact on most of the attacks discussed in Sect. 4. In this subsection, we briefly analyse the extreme cases $\alpha \approx 0$ and $\beta \approx 0$.

The condition $k = \lfloor \log_b M \rfloor + 1$ (cf. Algorithm 1 and Algorithm 5) implies $b^{k-1} \leq M < b^k$. Now assume that $b^k/2 < M < b^k$, which is typically fulfilled if the modulus M has 'maximum length' (e.g. 1024 or 2048 bits). Then,

$$0 \leq \beta = \frac{\lfloor b^{2k}/M \rfloor}{b^{k+1}} \leq \frac{2b^k}{b^{k+1}} = \frac{2}{b}. \tag{23}$$

If $b = 2^{\mathsf{ws}}$ with $\mathsf{ws} \gg 1$ (e.g. $\mathsf{ws} \geq 16$), then $\beta \approx 0$. In this case, one may neglect the term 'βU' in (2), accepting a slight inaccuracy of the stochastic model.

Going the next step, cancelling 'βU_{n+1}' in (3) simplifies Stochastic Model 2 as (3) can be rewritten as $R_{n+1} = 1_{\{S_{n+1} < \alpha S_n^2\}}$ and $R_{n+1} = 1_{\{S_{n+1} < \alpha S_n t\}}$, respectively. This representation is equivalent to the Montgomery case (11), simplifying the computation of the variances, covariances and the probabilities in Lemma 7 (i) considerably (as for the Montgomery multiplication). The Barrett-specific features and difficulties yet remain.

Now assume that $b^{k-1} \leq M < 2b^{k-1}$. Then,

$$0 \leq \alpha = \frac{M^2}{b^{2k}} \left\{ \frac{b^{2k}}{M} \right\} \leq \frac{M^2}{b^{2k}} < \frac{4}{b^2}. \tag{24}$$

If again $b = 2^{\mathsf{ws}}$ with $\mathsf{ws} \gg 1$, e.g. $\mathsf{ws} \geq 8$, then $\alpha \approx 0$. As above we may neglect the terms '$\alpha s t$' in (2) and analogously 'αS_n^2', resp. '$\alpha S_n t$', in (3), which yields the representation $R_{n+1} = 1_{\{S_{n+1} < \beta U_{n+1}\}}$. Consequently, $R_1, R_2, \ldots$ are iid $\{0, 1\}$-valued random variables with $\Pr(R_j = 1) = \beta/2$, while $\beta \approx 0$ is quite likely the case $\alpha \approx 0$ should occur rarely because the bit length of the modulus would be slightly larger than a power of $b = 2^{\mathsf{ws}}$.

More generally, let us assume that $b^{k-1} \leq 2^{w'} b^{k-1} \leq M < 2^{w'+1} b^{k-1} \leq b^k$ for some $w' \in \{0, \ldots, \mathsf{ws} - 1\}$. With the same strategy as in (23) and (24), we conclude

$$0 \leq \alpha < \frac{(2^{w'+1} b^{k-1})^2}{b^{2k}} = \left(\frac{2^{w'+1}}{b} \right)^2 \quad \text{and}$$

$$0 \leq \beta \leq \frac{b^k/(2^{w'} b^{k-1})}{b^{k+1}} = \frac{1}{2^{w'}}.$$

The impact of $\alpha \approx 0$ and $\beta \approx 0$ on the attacks in Sect. 4 will be discussed in Sect. 4.4.

3.3 A short summary

The stochastic process $R_1, R_2, \ldots$ is the equivalent to $W_1, W_2, \ldots$ (Montgomery multiplication). Both stochastic processes are 1-dependent. Hence, it is reasonable to assume that attacks on Montgomery's multiplication algorithm can be transferred to implementations which use Barrett's multiplication algorithm. In Sect. 4, we will see that this is indeed the case.

However, for Barrett's multiplication algorithm additional problems arise. In particular, there is no equivalent to the characterization (10), which allows to directly analyse the stochastic process $W_1, W_2, \ldots$ For Barrett's algorithm, a 'detour' to the two-dimensional Markov process $(S_i, R_i)_{i \in \mathbb{N}}$ is necessary. Moreover, for Montgomery's multiplication algorithm, the respective integrals can be computed much easier than for Barrett's algorithm since simple closed formulae exist. If $\beta \approx 0$, the evaluation of the integrals becomes easier (as for Montgomery's algorithm), and if $\alpha \approx 0$, the computations become absolutely simple. For CRT implementations, the parameter estimation is more difficult for Barrett's multiplication algorithm than for Montgomery's algorithm. We return to these issues in Sect. 4.

4 Timing attacks against Barrett's modular multiplication

The conditional extra reduction in Montgomery's multiplication algorithm is the source of many timing attacks and local timing attacks [2,3,7,13,14,22–24,26–29]. Some of them even work in the presence of particular blinding mechanisms. When applied to modular exponentiation, the stochastic representations of the execution times are similar for Montgomery's and Barrett's multiplication algorithms. The analysis of Barrett's algorithm, however, is mathematically more challenging as explained in Sects. 2 and 3. In Sects. 4.1 to 4.3, we transfer attacks on Montgomery's multiplication algorithm to attacks on Barrett's multiplication algorithm, where we assume that the 'basic' Algorithm 1 is applied. We point out that our attacks can be adjusted to Algorithm 5 (cf. Sect. 3.2.2).

4.1 Timing attacks on RSA without CRT and on DH

In this subsection, we assume that M is an RSA modulus or the modulus of a DH-group (i.e. a subgroup of $\mathbb{F}_M^*$) and that $y^d \bmod M$ is computed with Algorithm 2, where $d = (d_{\ell-1}, \ldots, d_0)_2$ is a secret exponent. Blinding techniques are not applied. We transfer the attack from Sect. 6 in [24] to Barrett's multiplication algorithm and extend it by a look-ahead strategy.

The attacker (or evaluator) measures the execution times $t_j = \mathrm{Time}(y_j^d \bmod M)$ for $j = 1, \ldots, N$ for known bases y_j. The t_j may be noisy (cf. (14)). Moreover, we assume that the attacker knows (or has estimated) c and c_{ER}. (Sect. 6 in [29] explains a guessing procedure for Montgomery's multiplication algorithm.) In a pre-step, the sum $\ell + \mathrm{ham}(d)$ can be estimated in a straight-forward way. We may assume that the attacker knows ℓ and thus also $\mathrm{ham}(d)$. (If necessary the attack could be restarted with different candidates for ℓ. However, apart from its end the attack is robust against small deviations from the correct value ℓ.) At the beginning, the attacker subtracts the data-independent terms t_{set} and $(\ell + \mathrm{ham}(d) - 2)c$ from the timings t_j and divides the differences by c_{ER}, yielding the 'discretized' execution times $t_{\mathrm{d},1}, \ldots, t_{\mathrm{d},N}$.

The attack strategy is to guess subsequently the exponent bits $d_{\ell-1} = 1, d_{\ell-2}, \ldots$. For the moment we assume that the guesses $\widetilde{d}_{\ell-1} = 1, \widetilde{d}_{\ell-2}, \ldots, \widetilde{d}_{k+1}$ have been correct. Now we focus on the guessing procedure of the exponent bit d_k. Currently, Algorithm 2 'halts' before the if-statement (for $i = k$) so that k squarings and m (calculated from $\mathrm{ham}(d)$ and the guesses $\widetilde{d}_{\ell-1}, \ldots, \widetilde{d}_{k+1}$) multiplications still have to be carried out. On the basis of the previous guesses, the attacker computes the intermediate values x_j, and the number of extra reductions needed for the squarings and multiplications executed so far are subtracted from the $t_{\mathrm{d},j}$, yielding the discretized remaining execution times $t_{\mathrm{drem},j}$. In terms of random variables, this reads

$$T_{\mathrm{drem},j} = R_{\ell+\mathrm{ham}(d)-k-m-1,j} + \cdots + R_{\ell+\mathrm{ham}(d)-2,j} + N_{\mathrm{d},j}$$

for $1 \le j \le N$ with $N_{\mathrm{d},j} \sim \mathcal{N}(0, \sigma^2/c_{\mathrm{ER}}^2)$. Recall that the distribution of those R_i, which belong to multiplications, depends on the basis y_j; see, for example, the stochastic representation (3).

To optimize our guessing strategy, we apply statistical decision theory. We point the interested reader to [25], Sect. 2, where statistical decision theory is introduced in a nutshell and the presented results are tailored to side-channel analysis. In the following, $\Theta := \{0, 1\}$ denotes the parameter space, where $\theta \in \Theta$ corresponds to the hypothesis $d_k = \theta$.

We may assume that the probability that the exponent bit d_k equals θ is approximately 0.5. (In the case of RSA, $d_0 = 1$ and for indices k close to $\lceil \log_2(n) \rceil$ the exponent bits may be biased.) More formally, if we view $d_k, d_{k-1}, \ldots$ as realizations of iid uniformly $\{0, 1\}$-distributed random variables $Z_k, Z_{k-1}, \ldots$ we obtain the a priori distribution

$$\begin{aligned}\eta(\theta) &= \Pr(Z_k = \theta \mid \mathrm{ham}(d_k, \ldots, d_0) = m) \\ &= \left(\frac{k+1-m}{k+1}\right)^{1-\theta} \left(\frac{m}{k+1}\right)^{\theta}. \end{aligned} \tag{25}$$

To guess the next exponent bit d_k, we employ a look-ahead strategy. For look-ahead depth $\lambda \in \mathbb{N}_{\ge 1}$, the decision for exponent bit d_k is based on information obtained from the next λ exponent bits. As the attacker knows the intermediate value x_j (of sample j), he is able to determine the number of extra reductions needed to process the λ exponent bits $d_k, \ldots, d_{k-\lambda+1}$ for each of the 2^λ admissible values $\boldsymbol{\rho} = (\rho_0, \ldots, \rho_{\lambda-1}) \in \{0, 1\}^\lambda$. This yields the discretized time needed to process the left-over exponent bits $d_{k-\lambda}, \ldots, d_0$, which is fictional except for the correct vector $\boldsymbol{\rho}$.

In this subsection, $t_{\boldsymbol{\rho},j}$ denotes the number of extra reductions required to process the next λ exponent bits for the basis y_j if $(d_k, \ldots, d_{k-\lambda+1}) = \boldsymbol{\rho}$. For these computations, λ modular squarings and $\mathrm{ham}(\boldsymbol{\rho})$ modular multiplications by y_j are performed. Furthermore, $t_{\mathrm{drem},j} - t_{\boldsymbol{\rho},j}$ may be viewed as a realization of

$$R_{\ell+\mathrm{ham}(d)-k-m+\lambda+\mathrm{ham}(\boldsymbol{\rho})-1,j} + \cdots + R_{\ell+\mathrm{ham}(d)-2,j} + N_{\mathrm{d},j}.$$

By (8), Lemma 6 (ii), and Lemma 7 (ii), (iii), this random variable is approximately $\mathcal{N}(e_{\boldsymbol{\rho},j}, v_{\boldsymbol{\rho},j})$-distributed where

$$\begin{aligned} e_{\boldsymbol{\rho},j} &:= (k-\lambda)\mu_{\mathrm{S}} + (m - \mathrm{ham}(\boldsymbol{\rho}))\mu_{\mathrm{M},j} \quad \text{and} \\ v_{\boldsymbol{\rho},j} &:= (k-\lambda)\sigma_{\mathrm{S}}^2 + (m - \mathrm{ham}(\boldsymbol{\rho}))\sigma_{\mathrm{M},j}^2 \\ &\quad + 2(m - \mathrm{ham}(\boldsymbol{\rho}))(\mathrm{cov}_{\mathrm{MS},j} + \mathrm{cov}_{\mathrm{SM},j}) \\ &\quad + 2(k - \lambda - m + \mathrm{ham}(\boldsymbol{\rho}) - 1)\,\mathrm{cov}_{\mathrm{SS}} \\ &\quad + \mathrm{Var}(N_{\mathrm{d},j}). \end{aligned} \tag{26}$$

We define the observation space $\Omega := (\Omega')^N$ consisting of vectors $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_N)$ of timing observations

$$\omega_j = \left(t_{\mathrm{drem},j}, (t_{\boldsymbol{\rho},j})_{\boldsymbol{\rho}\in\{0,1\}^\lambda}\right) \in \Omega' := \mathbb{R} \times \mathbb{N}^{2^\lambda} \tag{27}$$

for $1 \le j \le N$. For the remainder of this subsection, we denote the Lebesgue density of $\mathcal{N}(e_{\boldsymbol{\rho},j}, v_{\boldsymbol{\rho},j})$ by $f_{\boldsymbol{\rho},j}(\cdot)$. The joint distribution of all N traces is given by the product density $f_{\boldsymbol{\rho},1}(\cdot) \cdots f_{\boldsymbol{\rho},N}(\cdot)$ with the arguments $t_{\mathrm{drem},1} - t_{\boldsymbol{\rho},1}, \ldots, t_{\mathrm{drem},N} - t_{\boldsymbol{\rho},N}$. For $\lambda = 1$ we have $\boldsymbol{\rho} = \rho_0 = \theta$, and for hypothesis $d_k = \theta$ the distribution of the discretized computation time needed for the left-over exponent bits $d_{k-\lambda}, d_{k-\lambda-1}, \ldots$ has the product density $\prod_{j=1}^N f_{\theta,j}(\cdot)$.

If $\lambda > 1$, the situation is more complicated. More precisely, for hypothesis $\theta \in \Theta$ the distribution of the left-over time is given by a convex combination of normal distributions with density $\overline{f}_\theta \colon \Omega \to \mathbb{R}$ given by

$$\overline{f}_\theta(\boldsymbol{\omega}) = \sum_{\substack{\boldsymbol{\rho}\in\{0,1\}^\lambda: \\ \rho_0=\theta}} \mu_{\boldsymbol{\rho}} \prod_{j=1}^N f_{\boldsymbol{\rho},j}(t_{\mathrm{drem},j} - t_{\boldsymbol{\rho},j}), \tag{28}$$

where the coefficients $\mu_{\boldsymbol{\rho}}$ are given by

$$\begin{aligned}\mu_{\boldsymbol{\rho}} &= \Pr\big(Z_{k-1}=\rho_1,\ldots,Z_{k-\lambda+1}=\rho_{\lambda-1}\mid \\ &\qquad \mathrm{ham}(d_{k-1},\ldots,d_0)=m-\rho_0\big)\\ &= \prod_{i=1}^{\lambda-1}\Pr\big(Z_{k-i}=\rho_i\mid \\ &\qquad \mathrm{ham}(d_{k-i},\ldots,d_0)=m-\mathrm{ham}(\rho_0,\ldots,\rho_{i-1})\big)\\ &= \prod_{i=1}^{\lambda-1}\left(\frac{k+1-i-(m-\mathrm{ham}(\rho_0,\ldots,\rho_{i-1}))}{k+1-i}\right)^{1-\rho_i}\\ &\qquad\cdot\left(\frac{m-\mathrm{ham}(\rho_0,\ldots,\rho_{i-1})}{k+1-i}\right)^{\rho_i}.\end{aligned} \tag{29}$$

Finally, we choose $A=\Theta$ as the set of alternatives and consider the loss function

$$s\colon \Theta\times A\to\mathbb{R}_{\geq 0},\quad s(\theta,a)=1_{\{\theta\neq a\}},$$

i.e. we penalize the wrong decisions ('0' instead of '1', '1' instead of '0') equally since all forthcoming guesses then are useless. We obtain the following optimal decision strategy for look-ahead depth λ.

Decision Strategy 1 *(**Guessing** d_k) Let $\boldsymbol{\omega}=(\omega_1,\ldots,\omega_N)$ be a vector of N timing observations $\omega_j\in\Omega'$ as in (27). Then, the indicator function*

$$\tau(\boldsymbol{\omega}) := 1_{\{\overline{f}_0(\boldsymbol{\omega})/\overline{f}_1(\boldsymbol{\omega})\leq\eta(1)/\eta(0)\}} \tag{30}$$

is an optimal strategy (Bayes strategy) against the a priori distribution η.

Proof We interpret $\omega_1,\omega_2,\ldots\omega_N$ as realizations of independent random vectors $X_j := (T_{\mathrm{drem},j},(T_{\boldsymbol{\rho},j})_{\boldsymbol{\rho}\in\{0,1\}^\lambda})$ with values in Ω' for $1\leq j\leq N$, which has already been assumed when (28) was developed. We denote by μ the product of the Lebesgue measure on $\mathbb{R}$ and the counting measure on $\mathbb{N}^{2^\lambda}$. We equip Ω' with the product σ-algebra $\mathscr{B}(\mathbb{R})\otimes\mathscr{P}(\mathbb{N}^{2^\lambda})$. Then, μ is a σ-finite measure on Ω', and the N-fold product measure $\mu^N=\mu\otimes\cdots\otimes\mu$ is σ-finite on the observation space $\Omega=\Omega'\times\cdots\times\Omega'$. Note that $\Pr((T_{\boldsymbol{\rho},j})_{\boldsymbol{\rho}\in\{0,1\}^\lambda}=(t_{\boldsymbol{\rho},j})_{\boldsymbol{\rho}\in\{0,1\}^\lambda})$ is independent of $(d_k,\ldots,d_{k-\lambda+1})$ and thus in particular independent of d_k. Furthermore, this probability is >0 because all alternatives $\boldsymbol{\rho}$ are principally possible. Define $C_j := \mathbb{R}\times\prod_{\boldsymbol{\rho}\in\{0,1\}^\lambda}\{t_{\boldsymbol{\rho},j}\}$ and $C := \prod_{j=1}^N C_j\subseteq\Omega$. (Here, $\prod$ denotes the Cartesian product of sets.) If $d_k=\theta$, then the conditional probability distribution of $(X_1,\ldots,X_N)$ given C is $\overline{f}_\theta\cdot\mu^N$. Thus, all conditions of Theorem 1 (iii) in [25] are fulfilled, and this completes the proof. □

For look-ahead depth $\lambda=1$, Decision Strategy 1 is essentially equivalent to Theorem 6.5 (i) in [24].

Remark 7 All decisions after a wrong bit guess are useless because then the attacker computes wrong intermediate values $x'_1,\ldots,x'_N$ and therefore values $t'_{\boldsymbol{\rho},j}$ that are not correlated to the correct number of extra reductions $t_{\boldsymbol{\rho},j}$. However, the situation is not symmetric in 0 and 1 because for $d_k=0$ one uncorrelated term and for $d_k=1$ two uncorrelated terms are subtracted. In [28] (look-ahead depth $\lambda=1$) for Montgomery's multiplication algorithm, an efficient three-option error detection and correction strategy was developed, which allowed to reduce the number of attack traces by $\approx 40\%$. We do not develop an equivalent strategy for Barrett's multiplication algorithm but apply a dynamic look-ahead strategy. This is much more efficient as we will see in Sect. 5.1. To the best of our knowledge, this look-ahead strategy is new if we ignore the fact that the idea was very roughly sketched in [24], Remark 4.1.

4.2 Timing attacks on RSA with CRT

The references [3,7,22] introduce and analyse or improve timing attacks on RSA implementations which use the CRT and Montgomery's multiplication algorithm, including the square & multiply exponentiation algorithm and table-based exponentiation algorithms. Even more, these attacks can be extended to implementations which are protected by exponent blinding [26,27].

Unless stated otherwise, we assume in this subsection that RSA with CRT applies Algorithm 6 with Barrett's multiplication (Algorithm 1). Let $n=p_1p_2$ be an RSA modulus, let d be a secret exponent, and let $y\in\mathbb{Z}_n$ be a basis. We set $y_{(i)} := y \bmod p_i$ and $d_{(i)} := d \bmod (p_i-1)$ for $i=1,2$. For $y\in\mathbb{Z}_n$ let $T(y) := \mathrm{Time}(y^d \bmod n)$.

Algorithm 6 RSA with CRT, left-to right exponentiation, Barrett multiplication.

Input: Prime factors p_1 and p_2 with $n=p_1p_2$, exponents $d_{(i)}=d \bmod (p_i-1)$, and an integer $y\in\mathbb{Z}_n$.
Output: $y^d \bmod n$.
1: $y_{(1)} := y \bmod p_1$
2: Compute $x_{(1)}=y_{(1)}^{d_{(1)}} \bmod p_1$ with Alg. 2
3: $y_{(2)} := y \bmod p_2$
4: Compute $x_{(2)}=y_{(2)}^{d_{(2)}} \bmod p_2$ with Alg. 2
5: Compute $x=y^d \bmod n$ from $x_{(1)}$ and $x_{(2)}$ (recombination step)
6: **return** x

Let $\nu := \lfloor\log_2 n\rfloor+1$ be the bit-length of n. We may assume that p_1, p_2 have bit-length $\approx\nu/2$ and that $d_{(1)}$, $d_{(2)}$

have Hamming weight $\approx \nu/4$. From (4), we obtain

$$\mathrm{E}(T(y)) \approx t_{\mathrm{set}} + 2c\left(\frac{\nu}{2} + \frac{\nu}{4}\right) + c_{\mathrm{ER}}\frac{\nu}{2}\sum_{i=1,2}\left(\frac{\alpha_i}{3} + \frac{\beta_i}{2}\right) + c_{\mathrm{ER}}\frac{\nu}{4}\sum_{i=1,2}\left(\frac{\alpha_i t_i}{2} + \frac{\beta_i}{2}\right), \tag{31}$$

where $t_i := y_{(i)}/p_i$. Now assume that $0 < u_1 < u_2 < n$ with $u_2 - u_1 \ll p_1, p_2$. Three cases are possible:

Case A: $\{u_1 + 1, \ldots, u_2\}$ does not contain a multiple of p_1 or p_2.
Case B_i: $\{u_1 + 1, \ldots, u_2\}$ contains a multiple of p_i, but not of p_{3-i}.
Case C: $\{u_1 + 1, \ldots, u_2\}$ contains a multiple of both p_1 and p_2.

By (31), we conclude

$$\mathrm{E}(T(u_2) - T(u_1)) = \mathrm{E}(T(u_2)) - \mathrm{E}(T(u_1)) \approx \begin{cases} 0 & \text{in Case A,} \\ -\frac{1}{8}\nu\alpha_i c_{\mathrm{ER}} & \text{in Case } B_i, \\ -\frac{1}{8}\nu(\alpha_1 + \alpha_2)c_{\mathrm{ER}} & \text{in Case C,} \end{cases} \tag{32}$$

because if p_i is in $\{u_1 + 1, \ldots, u_2\}$, then

$$t_{i,2} := (u_2 \bmod p_i)/p_i \approx 0 \quad \text{and} \quad t_{i,1} := (u_1 \bmod p_i)/p_i \approx 1.$$

For RSA without CRT, the parameters α and β can easily be calculated, while for RSA with CRT, the parameters $\alpha_1, \alpha_2, \beta_1, \beta_2$ are unknown and thus need to be estimated.

We note that

$$\beta_1\beta_2 = \left\lfloor \frac{b^{2k}}{p_1} \right\rfloor \cdot \left\lfloor \frac{b^{2k}}{p_2} \right\rfloor b^{-(2k+2)} \approx \frac{b^{2k-2}}{n}. \tag{33}$$

The parameter β_i is not sensitive against small deviations of p_i and could be approximated by $\lfloor b^{2k}/p_i \rfloor / b^{k+1} \approx b^{k-1}/p_i \approx b^{k-1}/\sqrt{n} \in (0, 1)$. However, this estimate can be improved at the end of attack phase 1 below because then more precise information on p_1 and p_2 is available. We mention that in the context of this timing attack the knowledge of β_1 and β_2 is only relevant to estimate $\mathrm{Var}(T(y))$, which allows to determine an appropriate sample size for the attack steps. Unlike β_i, the second term $\{b^{2k}/p_i\}$ of $\alpha_i = (p_i^2/b^{2k})\{b^{2k}/p_i\}$ and thus α_i is very sensitive against deviations of p_i since $p_i \ll b^{2k}$.

In the remainder of this subsection, we assume $p_1 < p_2 < 2p_1$, i.e. that p_1, p_2 have bit-length $\ell \approx \nu/2$. It follows that the interval $I_1 := \left(\sqrt{n/2}, \sqrt{n}\right)$ contains p_1 but no multiple of p_2 and the interval $I_2 := \left(\sqrt{n}, \sqrt{2n}\right)$ contains p_2 but no multiple of p_1. (In the general case, we would have to guess $r \in \mathbb{N}_{\geq 2}$ such that $(r-1)p_1 < p_2 < rp_1$. Then, $I_1 := \left(\sqrt{n/r}, \sqrt{n/(r-1)}\right)$ contains p_1 but no multiple of p_2 and $I_2 := \left(\sqrt{(r-1)n}, \sqrt{rn}\right)$ contains p_2 but no multiple of p_1.) Let $u'_0 := \lceil\sqrt{n/2}\rceil < u'_1 < \ldots < u'_h := \lfloor\sqrt{n}\rfloor$ be approximately equidistant integers in I_1 and let $u''_0 := \lceil\sqrt{n}\rceil < u''_1 < \ldots < u''_h := \lfloor\sqrt{2n}\rfloor$ be approximately equidistant integers in I_2, where $h \in \mathbb{N}$ is a small constant (say $h = 4$). Further, define

$$\mathrm{MeanTime}(u, N) := \frac{1}{N}\sum_{j=1}^{N} T(u + j). \tag{34}$$

The goal of attack phase 1 is to identify j', j'' such that $p_1 \in [u'_{j'-1}, u'_{j'}]$ and $p_2 \in [u''_{j''-1}, u''_{j''}]$. The selection of j' and j'' follows from the quantitative interpretation of (32). If α_i is small but α_{3-i} is significantly larger, the decision (for j', resp., for j'') in attack phase 1 might be incorrect, but this is of minor importance since attack phase 2 searches p_{3-i} anyway. If both α_1 and α_2 are small, the efficiency of the attack is low anyway. To be on the safe side one then may repeat phase 1 with larger sample size N_1. Moreover, (33) allows to check the selection of j' and j''.

Attack Phase 1

(1) Select an appropriate integer N_1.
(2) For $j = 1, \ldots, h$, compute

$$\delta'_j \leftarrow \mathrm{MeanTime}(u'_{j-1}, N_1) - \mathrm{MeanTime}(u'_j, N_1) \quad \text{and}$$
$$\delta''_j \leftarrow \mathrm{MeanTime}(u''_{j-1}, N_1) - \mathrm{MeanTime}(u''_j, N_1).$$

(3) Set $j' \leftarrow \arg\max_{1\leq j\leq h}\{\delta'_j\}$ and $j'' \leftarrow \arg\max_{1\leq j\leq h}\{\delta''_j\}$. (The attacker believes that $p_1 \in [u'_{j'-1}, u'_{j'}]$ and $p_2 \in [u''_{j''-1}, u''_{j''}]$.)
(4) Set $\widetilde{\alpha}_1 \leftarrow 8\delta'_{j'}/(\nu c_{\mathrm{ER}})$ and $\widetilde{\alpha}_2 \leftarrow 8\delta''_{j''}/(\nu c_{\mathrm{ER}})$ (estimates for α_1 and α_2).
(5) Set $\widetilde{\beta}_1 \leftarrow 2b^{k-1}/(u'_{j'-1} + u'_{j'})$ and $\widetilde{\beta}_2 \leftarrow 2b^{k-1}/(u''_{j''-1} + u''_{j''})$ (estimates for β_1 and β_2).

Attack Phase 2

(1) If $\widetilde{\alpha}_1 > \widetilde{\alpha}_2$, then set $i \leftarrow 1$, $u_1 \leftarrow u'_{j'-1}$, and $u_2 \leftarrow u'_{j'}$; else set $i \leftarrow 2$, $u_1 \leftarrow u''_{j''-1}$, and $u_2 \leftarrow u''_{j''}$. (Attack phase 2 searches for p_i iff $\widetilde{\alpha}_i > \widetilde{\alpha}_{3-i}$. This prime is assumed to be contained in $[u_1, u_2]$.)
(2) Select N_2 (depending on $\widetilde{\alpha}_i$ and $\mathrm{Var}_{\widetilde{\alpha}_1,\widetilde{\alpha}_2,\widetilde{\beta}_1,\widetilde{\beta}_2}(T(u))$).

(3) While $\log_2(u_2 - u_1) > \ell/2 - 6$, do the following:

(a) Set $u_3 \leftarrow \lfloor (u_1 + u_2)/2 \rfloor$.
(b) If

$$\begin{aligned}&\text{MeanTime}(u_2, N_2) - \text{MeanTime}(u_3, N_2)\\ &> -\tfrac{1}{16}\nu\widetilde{\alpha}_i c_{\text{ER}},\end{aligned}$$

then set $u_2 \leftarrow u_3$ (the attacker believes that Case A is correct); else set $u_1 \leftarrow u_3$ (the attacker believes that Case B_i is correct).

The decision rule follows from (32) (Case A vs. Case B_i). After phase 2 more than half of the upper bits of u_1 and u_2 coincide, which yields more than half of the upper bits of p_i (more precisely, $\approx \ell/2 + 6$). This enables attack phase 3.

Attack Phase 3

(1) Compute p_i with Coppersmith's algorithm [11].

Of course, all decisions in attack phase 2 (including the initial choice of u_1 and u_2) need to be correct. However, it is very easy to verify from time to time whether all decisions in attack phase 2 have been correct so far, or equivalently, whether the current interval (u_1, u_2) indeed contains p_i. If

$$\begin{aligned}&\text{MeanTime}(u_2 + N_2, N_2) - \text{MeanTime}(u_3 + N_2, N_2)\\ &< -\tfrac{1}{16}\nu\widetilde{\alpha} c_{\text{ER}},\end{aligned}$$

this confirms the assumption that (u_1, u_2) contains p_i, and (u_1, u_2) is called a 'confirmed interval', but if not, one computes $\text{MeanTime}(u_2+2N_2, N_2) - \text{MeanTime}(u_3+2N_2, N_2)$. If this difference is $< -\frac{1}{16}\nu\widetilde{\alpha} c_{\text{ER}}$, then (u_1, u_2) becomes a confirmed interval. Otherwise, the attack goes back to the preceding confirmed interval $(u_{1;c}, u_{2;c})$ and restarts with values in the neighbourhood of $u_{1;c}$ and $u_{2;c}$, which have not been used before when the attack already was at this stage.

Remark 8 (i) *Similarities to Montgomery's multiplication algorithm.* By (4), the expected number of extra reductions needed for a multiplication by $y_i := y \bmod p_i$ is an affine function in $t_i = y_i/p_i$. (For Montgomery's multiplication algorithm, it is a linear function in $(yR \bmod p_i)/p_i$, cf. (12).) As for Montgomery's multiplication algorithm, (31) allows to decide whether an interval contains a prime p_1 or p_2 and finally to factorize the RSA modulus n.

(ii) *Differences to Montgomery's multiplication algorithm.* If $y_1 < p_i < y_2$, the expectation $\text{E}(T(y_1) - T(y_2))$ is linear in α_i, which is very sensitive to variations in p_i. Consequently, the attack efficiency may be very different whether the attacker targets p_1 or p_2. This is unlike to Montgomery's multiplication algorithm where the corresponding expectation is linear in $p_i/R \approx \sqrt{n}/R$. As a consequence, attack phase 1 is very different in both cases, depending on whether the targeted implementation applies Barrett's or Montgomery's multiplication algorithm.

It should be noted that this timing attack against Barrett's multiplication algorithm can be adapted to fixed window exponentiation and sliding window exponentiation and also works against exponent blinding. For table-based methods, the timing difference in (32) gets smaller, while exponent blinding causes large algorithmic noise. In both cases, the parameters N_1 and N_2 must be selected considerably larger, which of course lowers the efficiency of the timing attack. This is rather similar to timing attacks on Montgomery's multiplication algorithm [26, 27].

4.3 Local timing attacks

Unlike for the 'pure' timing attacks discussed in Sects. 4.1 and 4.2, we assume that a potential attacker is not only able to measure the overall execution time but also the timing for each squaring and multiplication, which means that he knows the number of extra reductions. This may be achieved by power measurements. In [2], an instruction cache attack was applied against Montgomery's multiplication algorithm. The task of a spy process was to realize when a particular routine from the BIGNUM library is applied, which is only used to calculate the extra reduction. This approach may not be applicable against Barrett's multiplication algorithm because here more than one extra reduction is possible.

In this subsection, we assume that fixed window exponentiation is applied where basis blinding (introduced in [19], Sect. 10, a.k.a. message blinding) is a applied as a security measure. Algorithm 7 updates the blinding values to prevent an attacker from calibrating an attack to fixed blinding values. Our attack works against RSA without CRT and against RSA with CRT as well. The papers [1,2,14,23] consider several modular exponentiation algorithms with Montgomery's multiplication algorithm.

Algorithm 7 Fixed Window Exponentiation, basis blinding, Barrett modular multiplication.

Input: A modulus $M \in \mathbb{N}_{\geq 2}$ of length $k = \lfloor \log_b M \rfloor + 1$ in base b, width of the window $w \geq 2$, a secret exponent $d = (d_{\ell-1}, \ldots, d_0)_{2^w}$ in 2^w-ary representation, blinding values $A, B = A^{-d} \bmod M \in \mathbb{Z}_M$, the integer reciprocal $\mu = \lfloor b^{2k}/M \rfloor$ of M, and a basis $y \in \mathbb{Z}_M$.
Output: $y^d \bmod M$ and updated blinding values $A^2 \bmod M$, $B^2 \bmod M$.

1: $u_0 \leftarrow 1$
2: $u_1 \leftarrow A \cdot y \bmod M$
3: **for** $j \leftarrow 2, \ldots, 2^w - 1$ **do**
4: $\quad u_j \leftarrow u_{j-1} \cdot u_1 \bmod M$
5: **end for**
6: $x \leftarrow u_0$
7: **for** $j \leftarrow \ell - 1, \ldots, 0$ **do**
8: $\quad$ **for** $i \leftarrow 1, \ldots, w$ **do**
9: $\quad\quad x \leftarrow x \cdot x \bmod M$
10: $\quad$ **end for**
11: $\quad x \leftarrow x \cdot u_{d_j} \bmod M$
12: **end for**
13: $x \leftarrow B \cdot x \bmod M$
14: $A \leftarrow A^2 \bmod M$, $B \leftarrow B^2 \bmod M$
15: **return** x and A, B

Algorithm 8 RSA with CRT, modular exponentiation with Alg. 7.

Input: Prime factors p_1 and p_2 with $n = p_1 p_2$, exponents $d_{(i)} = d \bmod (p_i - 1)$, and a basis $y \in \mathbb{Z}_n$.
Output: $y^d \bmod n$.

1: $y_{(1)} := y \bmod p_1$
2: Compute $x_{(1)} = y_{(1)}^{d_{(1)}} \bmod p_1$ with Alg. 7
3: $y_{(2)} := y \bmod p_2$
4: Compute $x_{(2)} = y_{(2)}^{d_{(2)}} \bmod p_2$ with Alg. 7
5: Compute $x = y^d \bmod n$ from $x_{(1)}$ and $x_{(2)}$ (recombination step)
6: **return** x

4.3.1 RSA without CRT and DH

In this subsection, we assume that $y^d \bmod M$ is computed with Algorithm 7. The exponentiation phase starts in Step 6. Analogously to Algorithm 2 (left-to-right exponentiation), the exponentiation phase of Algorithm 7 can be described by a sequence of operations $\mathtt{O}_1, \mathtt{O}_2, \ldots$ with $\mathtt{O}_j \in \{\mathtt{S}, \mathtt{M}_0, \ldots, \mathtt{M}_{2^w-1}\}$, where $\mathtt{M}_\theta$ stands for a multiplication by the table entry u_θ. Since one multiplication by some table entry u_j follows each w squarings, it remains to guess the operations $\mathtt{O}_{j(w+1)} \in \{\mathtt{M}_0, \ldots, \mathtt{M}_{2^w-1}\}$ for $j = 1, 2, \ldots$.

As for Algorithm 2, the exponentiation phase can be described by a stochastic process $(S_n, R_n)_{n \in \mathbb{N}}$ on the state space $[0, 1) \times \mathbb{Z}_3$, where S_0 models Step 6 in Algorithm 7, i.e. $S_0 = 0$. Again, this is a Markov process. However, there are some peculiarities: First of all, $S_0 = \ldots = S_w$ are identical and correspond to the initialization $x := 1$ and to the first w squarings. We may assume that $S_{w+1}, S_{w+2}, \ldots$ are uniformly distributed on $[0, 1)$. However, if $\mathtt{O}_{j(w+1)} = \mathtt{M}_0$ (multiplication by $u_0 = 1$), then $S_{j(w+1)-1} = S_{j(w+1)}$. The sequence $S_{w+1}, S_{w+2}, \ldots$ with those duplicate variables removed is iid uniformly distributed on $[0, 1)$. In particular, Remark 2 (ii) allows to identify the j's for which $\mathtt{O}_{j(w+1)} = \mathtt{M}_0$ (multiplication by 1). Consequently, we define $R_{w+1} := 0$ (multiplication by $x = 1$) and $R_{j(w+1)} := 0$ for all $j \geq 2$ with $\mathtt{O}_{j(w+1)} = \mathtt{M}_0$ (multiplication by $u_0 = 1$).

Since $R_{w+1} = 0$, the attacker has to guess the operation $\mathtt{O}_{w+1} \in \{\mathtt{M}_0, \ldots, \mathtt{M}_{2^w-1}\}$ by exhaustive search, which costs at most 2^w trials. From now on we focus on the operations $\mathtt{O}_{2(w+1)}, \mathtt{O}_{3(w+1)}, \ldots$

Lemma 10 *Let $j \geq 2$. For $\theta \in \{1, \ldots, 2^w - 1\}$, we have*

$$\Pr_\theta(R_{j(w+1)} = r_{j(w+1)}) = \int_0^1 \left(\int_0^1 h_{[\theta]}(s_{j(w+1)}, r_{j(w+1)} \mid s_{j(w+1)-1}) \mathrm{d}s_{j(w+1)} \right) \mathrm{d}s_{j(w+1)-1}. \tag{35}$$

The conditional density $h_{[\theta]}(s_{n+1}, r_{n+1} \mid s_n)$ is defined like h_{n+1} in Lemma 5 (iii) with $\mathtt{O}_{n+1} = \mathtt{M}$ and (s_{n+1}, s_n, t) replaced by $(s_{j(w+1)}, s_{j(w+1)-1}, t_{[\theta]} = s'_\theta)$. Here, $\Pr_\theta$ denotes the probability under the hypothesis θ, i.e. under the assumption $\mathtt{O}_{j(w+1)} = \mathtt{M}_\theta$. Further, we have

$$\Pr_0(R_{j(w+1)} = 0) = 1. \tag{36}$$

Proof Equation (35) follows from Lemma 6 (i) with $\mathtt{M} = \mathtt{M}_\theta$ and equation (36) holds by definition. □

However, the attacker does not know the ratios $t_1, \ldots, t_{2^w-1}$ and in Sect. 4.3.2, additionally, not even the moduli p_1 and p_2. Anyway, the table initialization phase is the source of our attack since the attacker knows the type of operations. In analogy to the exponentiation phase, we formulate a stochastic process $(S'_j, R'_j)_{1 \leq j \leq 2^w-1}$ on the state space $[0, 1) \times \mathbb{Z}_3$, where S'_j corresponds to the normalized table entry $t_j := u_j/M$. This again defines a two-dimensional Markov process, and $S'_1, \ldots, S'_{2^w-1}$ are iid uniformly distributed on $[0, 1)$. This leads to Lemma 11.

Lemma 11 *Let $j \geq 2$.*

(i) *For $\theta \in \{1, \ldots, 2^w - 1\}$, we have*

$$\begin{aligned}\Pr_\theta(R'_2 = r'_2, \ldots, R'_{2^w-1} = r'_{2^w-1}, R_{j(w+1)} = r_{j(w+1)}) = \int_0^1 \int_0^1 h^*(s'_2, r'_2 \mid s'_1) \cdots \int_0^1 h^*(s'_{2^w-1}, r'_{2^w-1} \mid s'_{2^w-2}) \\ \int_0^1 \int_0^1 h_{[\theta]}(s_{j(w+1)}, r_{j(w+1)} \mid s_{j(w+1)-1}) \\ \mathrm{d}s_{j(w+1)} \mathrm{d}s_{j(w+1)-1} \mathrm{d}s'_{2^w-1} \mathrm{d}s'_{2^w-2} \cdots \mathrm{d}s'_2 \mathrm{d}s'_1.\end{aligned} \tag{37}$$

For $j = 1, \ldots, 2^w - 2$, the function $h^*(s_{j+1}, r_{j+1} \mid s_j)$ is defined like h_{n+1} for $\mathtt{O}_{n+1} = \mathtt{M}$ in Lemma 5 (iii) with $(s'_{j+1}, s'_j, t_1 = s'_1)$ in place of (s_{n+1}, s_n, t).

(ii) *For $\theta \in \{0, \ldots, 2^w - 1\}$, we have*

$$\begin{aligned}&\Pr_\theta(R'_2 = r'_2, \ldots, R'_{2^w-1} = r'_{2^w-1}, R_{j(w+1)} = r_{j(w+1)})\\ &= \int_0^1 \int_0^1 h^*(s'_2, r'_2 \mid s'_1) \cdots \int_0^1 h^*(s'_{2^w-1}, r'_{2^w-1} \mid s'_{2^w-2})\\ &\quad \Pr_\theta(R_{j(w+1)} = r_{j(w+1)}) \mathrm{d}s'_{2^w-1} \mathrm{d}s'_{2^w-2} \ldots \mathrm{d}s'_2 \mathrm{d}s'_1.\end{aligned} \tag{38}$$

Proof The random variables $S'_1, \ldots, S'_{2^w-1}, S_{j(w+1)-1}, S_{j(w+1)}$ are iid uniformly distributed on $[0, 1)$ for $\theta > 0$. If $\theta = 0$, then $S_{j(w+1)-1} = S_{j(w+1)}$, but this property holds for $S'_1, \ldots, S'_{2^w-1}, S_{j(w+1)-1}$. For $j = 1 \ldots, 2^w - 2$ and for given r'_{j+1}, the conditional densities $h^*(s'_{j+1}, r'_{j+1} \mid s'_j)$ quantify the dependency on the 'history' (remember that $S'_1, \ldots, S'_{2^w-1}$ is a Markov process). Similarly, $h_{[\theta]}(s_{j(w+1)}, r_{j(w+1)} \mid s_{j(w+1)-1})$ quantifies the dependency on the 'history'. As in the proof of Lemma 7 (i), equation (37) follows from the Ionescu–Tulcea theorem. Evaluating the integrals with respect to $s_{j(w+1)-1}$ and $s_{j(w+1)}$ proves (38) for $\theta > 0$. For $\theta = 0$, it is obvious because $R'_{j(w+1)}$ does not depend on $S'_1, \ldots, S'_{j(w+1)-1}$. □

Decision Strategy 2 *Let $j \geq 2$ and assume that the attacker has observed N samples of extra reduction vectors*

$$(r'_{2,k}, \ldots, r'_{2^w-1,k}, r_{j(w+1),k}) \quad \text{for } 1 \leq k \leq N.$$

Then, the optimal decision strategy is to decide for $\mathtt{O}_{j(w+1)} = \mathtt{M}_{\theta^*}$, where

$$\begin{aligned}\theta^* := arg\ max_{\theta \in \{0,\ldots,2^w-1\}} \prod_{k=1}^{N} \Pr_\theta\big(&R'_{2,k} = r'_{2,k}, \ldots,\\ &R'_{2^w-1,k} = r'_{2^w-1,k}, R_{j(w+1),k} = r_{j(w+1),k}\big).\end{aligned} \tag{39}$$

Proof All $\theta \in \{0, \ldots, 2^w - 1\}$ are equally likely, and each false decision is equally harmful. Hence, the optimal decision strategy is given by the maximum likelihood estimator, which follows, for example, from Theorem 1 (i) in [25]. The extra reduction vectors

$$\begin{aligned}&(R'_{2,1}, \ldots, R'_{2^w-1,1}, R_{j(w+1),1}), \ \ldots\\ &\ldots, \ (R'_{2,N}, \ldots, R'_{2^w-1,N}, R_{j(w+1),N})\end{aligned}$$

may be considered to be independent, which yields (39). Formula (39) combines both cases $\theta = 0$ and $\theta \neq 0$. □

Remark 9 (i) *Similarities to Montgomery's multiplication algorithm.* For both Barrett's and Montgomery's multiplication algorithm, the joint probabilities can be expressed by integrals over high-dimensional unit cubes. In both cases, the type of the first multiplication (operation $w + 1$) has to be guessed exhaustively.

(ii) *Differences to Montgomery's multiplication algorithm.* For Barrett's multiplication algorithm, these integrals are $(2^w + 1)$-dimensional, while for Montgomery's algorithm, they are $(2^w + 2)$-dimensional. This is due to the pre-operation (cf. Algorithm 4, Step 1).

4.3.2 RSA with CRT

In this subsection, we assume that $y^d \bmod n$ is computed with Algorithm 8. Algorithm 8 calls Algorithm 7 twice, which applies basis blinding. (Alternatively, the basis blinding could also be moved to a pre-step 0 and to Step 6 in Algorithm 8, but this would not affect the attack.) Here, the situation for the attacker is even less favourable than in Sect. 4.3.1 because not only the blinded basis and thus the table entries are unknown but also the moduli p_1 and p_2. However, apart from additional technical difficulties our attack still applies.

We first note that it suffices to recover $d_{(1)}$ or $d_{(2)}$. In fact, if $x = y^d \bmod n$, then $\gcd(x - y^{d(i)} \bmod n, n) = p_i$ for $i = 1, 2$ (cf. [2], eq. (1)). If necessary, the attacker may construct such a pair, e.g. by setting $y = x'^e \bmod n$ and $x = x'$ for some $x' \in \mathbb{Z}_n$.

The plan is to apply the attack method from the preceding subsection to the modular exponentiation $\bmod p_1$ or $\bmod p_2$. First, similarly as in Sect. 4.2, we estimate $\alpha_1, \beta_1, \alpha_2, \beta_2$. To estimate α_i and β_i we consider all $k'_\mathtt{S}$ squarings in the modular exponentiation $\bmod p_i$, beginning with operation $w + 2$. We neglect all multiplications by 1, which can be identified by the fact that no extra reductions can occur, so that $k'_\mathtt{M}$ relevant multiplications remain. Counting the extra reductions in the relevant squarings and multiplications over all N exponentiations give numbers $n_\mathtt{S}$ and $n_\mathtt{M}$, respectively. We may assume that for $\theta > 0$ the normalized table entries behave like realizations of iid random variables, which are uniformly distributed on $[0, 1)$. Thus, we may assume that the average of all normalized table entries over all N samples is ≈ 0.5. From (4), we thus conclude

$$n_\mathtt{S} \approx N k'_\mathtt{S} \left(\frac{\alpha_i}{3} + \frac{\beta_i}{2}\right) \quad \text{and} \quad n_\mathtt{M} \approx N k'_\mathtt{M} \left(\frac{\alpha_i}{4} + \frac{\beta_i}{2}\right). \tag{40}$$

Replacing $\approx$ by $=$ and solving the linear equations provides estimates $\widetilde{\alpha}_i$ and $\widetilde{\beta}_i$ for α_i and β_i. The attacker focuses on the exponentiation $\bmod p_1$ iff $\widetilde{\alpha}_1 > \widetilde{\alpha}_2$.

Remark 10 (i) *Similarities to Montgomery's multiplication algorithm.* As for Montgomery's multiplication algorithm, the attack remains feasible if the CRT is applied.

(ii) *Differences to Montgomery's multiplication algorithm.* Unlike for Montgomery's multiplication algorithm, the choice, whether the modular exponentiation mod p_1 or mod p_2 is attacked, may have significant impact on the attack's efficiency. This is similar to the situation in Sect. 4.2.

4.4 $\alpha \approx 0, \beta \approx 0$, Algorithm 5: impact on the attack efficiency

In Sect. 3.2.3, we mentioned that $\alpha \approx 0$ and $\beta \approx 0$ may occur if $b = 2^{\mathsf{ws}} \gg 2$, e.g. for $\mathsf{ws} = 8, 16, 32, 64$. While $\beta \approx 0$ is quite likely, $\alpha \approx 0$ is certainly possible but very unusual. For both special cases, the necessary computations become considerably easier.

For $\beta \approx 0$, all before-mentioned attacks remain feasible. The attacks in Sect. 4.2 and Sect. 4.3 exploit differences between multiplications with different factors, which are caused by different α-values. As $\beta \approx 0$, this reduces the part of the variances that does not depend on the particular factor. Hence, $\beta \approx 0$ there leads to even more efficient attacks, while this has little relevance for the attack in Sect. 4.1. If $\alpha \approx 0$ the $\{0, 1\}$-valued random variables $R_1, R_2, \ldots$ are iid distributed with $\Pr(R_j = 1) = \beta/2$, and thus, only the timing attack in Sect. 4.1 remains feasible.

In Sect. 3.2.2, we have shown that in Algorithm 5 extra additions are rather rare and thus can be treated like noise. For the timing attacks in Sect. 4.1 and Sect. 4.2, this effect slightly increases the variance and thus also the sample size. If $c_{\text{add}} \not\approx c_{\text{ER}}$ in the local timing attacks in Sect. 4.3, the extra additions can be detected yielding to the same situation as for Algorithm 1. If $c_{\text{add}} \approx c_{\text{ER}}$, we occasionally get a wrong number of extra reductions, which may slightly increase the sample size. (As already pointed out in Sect. 3.2.3 for $\beta \approx 0$, the situation is rather similar to Montgomery's multiplication algorithm.)

5 Experiments

In this section, we report experimental results for the attacks presented in Sect. 4. In each experiment, we used simulations of the exponentiation algorithms returning the exact number of extra reductions needed by the multiplications and squarings, either cumulative (pure timing attacks) or per operation (local timing attacks). This represents the ideal (noise-free) case (accurate timing measurement, no time noise by other operations). Of course, non-ideal scenarios would require larger sample sizes.

We performed timing attacks on RSA without CRT and on Diffie–Hellman exemplarily on 512-bit RSA moduli and on a 3072-bit DH group, and for RSA with CRT, we considered 1024-bit moduli and 3072-bit moduli. Finally, we carried out local timing attacks on 512-bit RSA moduli and on a 3072-bit DH group. Our experiments confirmed the theoretical results. Of course, 512-bit RSA moduli have not been secure for many years, and also 1024-bit RSA is no longer state-of-the-art. These choices yet allow direct comparisons to results from previous papers on Montgomery's multiplication algorithm which considered these modulus sizes.

We primarily report on experiments using Barrett's multiplication algorithm with base $b = 2$, but in the respective subsections, we also explain how the results change when a more common base $b = 2^{\mathsf{ws}}$ for $\mathsf{ws} \geq 8$ is used. The case $b = 2$ is usually less favourable for the attacker in terms of efficiency and represents a worst-case analysis in this sense. Mathematically, it is also the most challenging case as it requires the most general stochastic model.

The experiments were implemented using the Julia programming language [5] with the Nemo algebra package [15].

5.1 Timing attacks on RSA without CRT and on DH

We implemented the timing attack with the look-ahead strategy as presented in Sect. 4.1.

Since our simulations are noise-free, we set $\mathrm{Var}(N_{\mathrm{d},j}) := 0$ in (26). For simplicity, in (26) we approximated the covariances $\mathrm{cov}_{\mathtt{MS},j}$, $\mathrm{cov}_{\mathtt{SM},j}$ and $\mathrm{cov}_{\mathtt{SS}}$ by the empirical covariances of suitable samples generated from Stochastic Model 2. For instance, $\mathrm{cov}_{\mathtt{MS},j}$ can be approximated using samples from

$$R_n = \lceil \alpha S_{n-1} t_j + \beta U_n - S_n \rceil,$$
$$R_{n+1} = \lceil \alpha S_n^2 + \beta U_{n+1} - S_{n+1} \rceil,$$

where $t_j := y_j/M$ is the normalized basis of the j-th timing sample. (Recall that the random variables S_{n-1}, S_n, U_n and U_{n+1} are iid uniformly distributed on $[0, 1)$.) Furthermore, the n-th and the $(n+1)$-th Barrett operations correspond to a multiplication by the basis and a squaring, respectively. We point out that Lemma 7 (i) would theoretically allow an exact computation of these covariances since $\mathrm{Cov}(R_n, R_{n+1}) = \mathrm{E}(R_n R_{n+1}) - \mathrm{E}(R_n)\,\mathrm{E}(R_{n+1})$ and

$$\mathrm{E}(R_n R_{n+1}) = \sum_{1 \leq i,j \leq 2} ij \Pr(R_n = i,\, R_{n+1} = j).$$

In order to make the attack more efficient, we chose the look-ahead depth λ dynamically during the attack. Our strategy is based on the observation that when Decision Strategy 1 fails for the first time, the decisions usually are close, i.e.

using the notation of Sect. 4.1,

$$|\eta(0)\overline{f}_0(\boldsymbol{\omega}) - \eta(1)\overline{f}_1(\boldsymbol{\omega})|$$

is small. Close decisions happen mostly at the beginning of the attack when the number of remaining exponent bits k is large (if $\mathrm{Var}(N_{\mathrm{d},j}) \approx 0$, then the variance $v_{\boldsymbol{\rho},j}$ in (26) decreases (essentially) linearly in k) or if the number N of timing samples is insufficient. For each decision, we gradually incremented the look-ahead depth λ starting from 1 until either the heuristic condition

$$\left|\eta(0)\overline{f}_0(\boldsymbol{\omega}) - \eta(1)\overline{f}_1(\boldsymbol{\omega})\right| \geq 3 \cdot \log_2\left(1 + N/k\right)$$

was fulfilled or we reached some maximum look-ahead depth $\lambda_{\max}$.

For simplicity, we assumed in our experiments that we know the bit-length and Hamming weight of the secret exponent (cf. Sect. 4.1).

For the first experiment, we used Diffie–Hellman group `dhe3072` defined in Appendix A.2 of RFC 7919 [16]. This reference recommends to use this group with secret exponents of bit-length at least 275. The 3072-bit modulus of this group has Barrett parameters $\alpha \approx 0.63$ and $\beta \approx 0.5$ for the base $b = 2$ we used. The results of the experiment are reported in Table 2. For each sample size N and maximum look-ahead depth $\lambda_{\max}$ given in the table, we conducted 100 trials of the timing attack. For each trial, a 275-bit secret exponent and N input bases were chosen independently at random.

Table 2 shows that in terms of the sample size the applied (dynamic) look-ahead strategy is much more efficient than the constant look-ahead strategy $\lambda = 1$.

For the second experiment, we considered RSA with 512-bit moduli and again used Barrett's multiplication with base $b = 2$. Of course, factoring 512-bit RSA moduli has been an easy task for many years, but this choice allows a direct comparison with the results on Montgomery's multiplication algorithm in [28]. The results of these experiments are reported in Table 3. For each sample size N and maximum look-ahead depth $\lambda_{\max}$ given in the table, we conducted 100 trials of the timing attack. For each trial, a 512-bit RSA-modulus and N input bases were chosen independently at random. Since we chose $e := 65537$ as public exponents, the secret exponents were ensured to be of bit-length near 512 as well.

Reference [28] treats 512-bit RSA moduli, the square and multiply algorithm and Montgomery's multiplication algorithm. In our terminology, [28] applies the optimal decision strategy for the constant look-ahead strategy $\lambda = 1$. For the sample size $N = 5000$ (for $N = 7000$, for $N = 9000$), simulations yielded success probabilities of 12% (55%, 95%). The results in Table 3 underline that the efficiency of the attacks on Barrett's and on Montgomery's multiplication algorithm is rather similar for $\lambda = 1$. Moreover, in [28] also so-called real-life attacks were conducted where the timings were gained from emulations of the Cascade chip (see [28], Remark 4). For the above-mentioned sample sizes, the real-life attack was successful in 15%, 40% or 72% of the trials. In [28], further experiments were conducted where the optimal decision strategy was combined with an error detection and correction strategy. There already $N = 5000$ led to success rates of 85% (simulation) and 74% (real-world attack). This improved the efficiency of the original attack on the Cascade chip in [13] by a factor ≈ 50. We refer the interested reader to [28], Table 1 and Table 2, for further experimental results.

This error detection and correction strategy can be adjusted to Barrett's multiplication algorithm, and for $\lambda = 1$ this should also increase the attack efficiency (in terms of N) considerably. Of course, one might combine this error detection and correction strategy with our look-ahead strategy. We do not expect a significant improvement because the look-ahead strategy treats suspicious ('close') decisions with particular prudence (by enlarging λ). However, the other way round the look-ahead strategy can be applied to Montgomery's multiplication algorithm as well and should yield similar improvements.

Decision Strategy 1 considers the remaining execution time and the following λ exponent bits. Since the variance of the remaining execution time $v_{\boldsymbol{\rho},j}$ (26) is approximately linear in the number of exponent bits, wrong decisions should essentially occur in the beginning of the attack. This observation suggests a coarse rule of thumb to extrapolate the necessary sample size to different exponent lengths: When the exponent length increases from ℓ_1 to ℓ_2, the sample size N should increase by factor $\approx \ell_2/\ell_1$. The experimental results in Table 2 and Table 3 are in line with this formula, cf. the results for $N = 1000$ (Table 2) and $N = 2000$ (Table 3), for instance.

Our experiments showed the interesting feature that unlike for the attacks in Sects. 5.2 and 5.3, the parameters α and β only play a small role for the efficiency, at least under optimal conditions when no additional noise occurs. Qualitatively, this feature can be explained as follows: Decision Strategy 1 essentially exploits the fact that the variances of the 2^λ (hypothetical) remaining execution times should be the smaller the more of the left-hand bits of the λ-bit window are correct. Large (resp., small) α, β imply large (resp., small) variances and differences of variances. In the presence of additional noise large α, β should favour the attack efficiency because then the relative impact of the additional noise on the variance is smaller.

We repeated some of the experiments in Tables 2 and 3 for base $b = 2^8$ and obtained comparable success rates.

Table 2 Timing attack on Diffie–Hellman group `dhe3072` with 275-bit secret exponents (base $b = 2$)

Sample size	Success rate *(in %)*									
N	$\lambda_{\max}$:	1	2	3	4	5	6	7	8	9	10
400		0	0	0	0	0	1	2	1	1	4
600		0	0	1	2	12	13	21	41	39	50
800		0	0	5	23	37	54	74	73	75	83
1000		0	0	31	42	65	82	83	88	93	91
1200		0	11	37	72	88	89	94	97	98	100
1400		3	32	69	89	95	94	100	99	99	99

Table 3 Timing attack on 512-bit RSA without CRT (base $b = 2$)

Sample size	Success rate *(in %)*									
N	$\lambda_{\max}$:	1	2	3	4	5	6	7	8	9	10
1000		0	0	0	0	0	0	2	8	8	12
1500		0	0	0	8	18	32	50	63	71	77
2000		0	0	15	37	62	80	84	93	97	92
2500		0	7	40	74	91	89	97	98	100	97
3000		0	25	66	87	96	98	100	100	98	99
3500		0	49	80	98	100	99	99	98	100	99
5000		17	89	100							
7000		66	99								
9000		89	100								

5.2 Timing attacks on RSA with CRT

We implemented the timing attack presented in Sect. 4.2 (RSA with CRT, square & multiply algorithm).

The efficiency of the attack depends on the Barrett parameters $\alpha_1, \alpha_2, \beta_1, \beta_2$ of the RSA-primes p_1, p_2. The value $\alpha_i := \max\{\alpha_1, \alpha_2\}$ (which is estimated in attack phase 1) determines the 'size' of useful information ('signal'), whereas the values $\alpha_{3-i}, \beta_1, \beta_2$ determine the size of unwanted 'algorithmic noise' for the attack. We use $\alpha_{\max} := \max\{\alpha_1, \alpha_2\}$ and $\overline{\beta} := (\beta_1 + \beta_2)/2$ as simple measures for the amount of signal and algorithmic noise, respectively.

The results of our experiments are reported in Table 4 and Table 5 for RSA moduli of bit-size $\nu := 1024$ (allowing a comparison with [22]) and $\nu := 3072$, respectively. We used Barrett's multiplication algorithm with base $b = 2$ and RSA primes of bit-size $\nu/2$. Each row of the table represents an experiment consisting of 100 trials. For each trial, we used rejection sampling to generate an RSA modulus such that $\alpha_{\max}$ and $\overline{\beta}$ are contained in the given intervals.

In attack phase 1, we divided the initial intervals containing p_1 and p_2 into $h := 4$ subintervals each and chose a sample size of $N_1 := 32$. For attack phase 2, we estimated the sample size N_2 as follows. For an interval $[u_3, u_2]$, we consider $\Delta := T(u_2) - T(u_3)$ as random variable. Then, we have $\mathrm{Var}(\Delta) \approx (v_{i,3} + v_{i,2} + v_{3-i,3} + v_{3-i,2})c_{\mathrm{ER}}^2$, where the second indices equal the index of the corresponding value u_j. More precisely,

$$v_{i,j} := \frac{\nu}{2}\,\mathrm{var}_{\mathrm{S},\alpha_i,\beta_i} + \frac{\nu}{4}\,\mathrm{var}_{\mathrm{M},t_{i,j},\alpha_i,\beta_i} + 2\left(\frac{\nu}{4}\,\mathrm{cov}_{\mathrm{SS},\alpha_i,\beta_i} + \frac{\nu}{4}\,\mathrm{cov}_{\mathrm{MS},t_{i,j},\alpha_i,\beta_i} + \frac{\nu}{4}\,\mathrm{cov}_{\mathrm{SM},t_{i,j},\alpha_i,\beta_i}\right)$$

and $t_{i,j} := (u_j \bmod p_i)/p_i$. Plugging in the estimates

$$\widetilde{\alpha}_1, \widetilde{\alpha}_2, \widetilde{\beta}_1, \widetilde{\beta}_2$$

from attack phase 1 as well as $\widetilde{t}_{i,3} \approx 1$, $\widetilde{t}_{i,2} = 0$, and $\widetilde{t}_{3-i,3} = \widetilde{t}_{3-i,2} \approx 1$ (worst case), we obtain an approximate upper bound $\widetilde{\sigma}_{\Delta,i}^2$ for $\mathrm{Var}(\Delta)$ (holding for both Case A and Case B_i). Let γ denote the probability that

$$\mathrm{MeanTime}(u_2, N_2) - \mathrm{MeanTime}(u_3, N_2) > -\frac{1}{16}\nu\widetilde{\alpha}_i c_{\mathrm{ER}},$$

although $p_i \in [u_3, u_2]$. Each difference $T(u_2 + j) - T(u_1 + j)$ may be viewed as a realization of the difference of two normally distributed random variables, and thus, the error probability γ is approximately

$$\Phi\left(-\frac{1}{16}\nu\widetilde{\alpha}_i c_{\mathrm{ER}}/(\widetilde{\sigma}_{\Delta,i}/\sqrt{N_2})\right).$$

Table 4 Timing attack on 1024-bit RSA with CRT (base $b = 2$, $\gamma = 2^{-8}$ in (41))

Signal	Alg. noise	Sample sizes				Errors	Success
$\alpha_{\max}$	$\overline{\beta}$	$\widetilde{N}_2$	$\overline{N}_2$	$\widetilde{N}_{\text{total}}$	$\overline{N}_{\text{total}}$	$\overline{E}_{\text{total}}$	*(in %)*
[0.0,0.2]	[0.65, 0.80]	72.5	124.71	24226.5	46566.76	1.26	62
	[0.50, 0.65]	59	101.14	18851.5	33270.86	0.78	64
[0.2,0.4]	[0.65, 0.80]	15	18.19	4878	5882.48	0.53	97
	[0.50, 0.65]	15	17.80	4624	5630.72	0.41	86
[0.4,0.6]	[0.65, 0.80]	6	6.40	2171	2168.31	0.39	98
	[0.50, 0.65]	6	6.20	1973	2149.02	0.44	98
[0.6,0.8]	[0.65, 0.80]	3	3.37	1200	1241.42	0.21	100
	[0.50, 0.65]	3	3.31	1101	1240.08	0.25	99
[0.8,1.0]	[0.65, 0.80]	2	2.06	830	874.54	0.20	94
	[0.50, 0.65]	2	2.06	830	864.67	0.13	100
[0.0,1.0]	[0.50, 0.80]	6	25.28	1973	9244.66	0.51	93

Table 5 Timing attack on 3072-bit RSA with CRT (base $b = 2$, $\gamma = 2^{-9}$ in (41))

Signal	Alg. noise	Sample sizes				Errors	Success
$\alpha_{\max}$	$\overline{\beta}$	$\widetilde{N}_2$	$\overline{N}_2$	$\widetilde{N}_{\text{total}}$	$\overline{N}_{\text{total}}$	$\overline{E}_{\text{total}}$	*(in %)*
[0.0,0.2]	[0.65, 0.80]	29	57.52	27103	49030.68	0.80	69
	[0.50, 0.65]	26.5	40.58	23444.5	34818.14	0.74	76
[0.2,0.4]	[0.65, 0.80]	7	7.34	5877.5	6431.51	0.60	92
	[0.50, 0.65]	6	6.87	5082	6008.79	0.50	94
[0.4,0.6]	[0.65, 0.80]	3	2.79	2685	2582.59	0.33	100
	[0.50, 0.65]	3	2.73	2685	2499.22	0.21	99
[0.6,0.8]	[0.65, 0.80]	2	1.53	1886	1525.63	0.22	100
	[0.50, 0.65]	2	1.54	1884	1530.20	0.18	100
[0.8,1.0]	[0.65, 0.80]	1	1.00	1087	1095.58	0.13	100
	[0.50, 0.65]	1	1.00	1087	1089.64	0.04	100
[0.0,1.0]	[0.50, 0.80]	3	8.17	2682	7188.71	0.24	95

Therefore, a desired maximum error probability γ for a single decision can approximately be achieved by setting

$$N_2 := \left\lceil \left(\frac{16 \cdot \widetilde{\sigma}_{\Delta,i} \cdot \Phi^{-1}(\gamma)}{\nu \cdot \widetilde{\alpha}_i \cdot c_{\text{ER}}} \right)^2 \right\rceil. \tag{41}$$

We point out that (41) does not depend on c_{ER} because $\widetilde{\sigma}_{\Delta,i}$ is a multiple of c_{ER}. In the simulation, we thus may assume $c_{\text{ER}} = 1$. Of course, in a noisy setting the relation between the noise and c_{ER} is relevant. For the experiments in Tables 4 and 5, we chose $\gamma := 2^{-8}$ and $\gamma := 2^{-9}$, respectively. The median and mean of the values chosen for N_2 in successful attacks are denoted by $\widetilde{N}_2$ and $\overline{N}_2$; the median and mean of the total number of timing samples required for an successful attack are denoted by $\widetilde{N}_{\text{total}}$ and $\overline{N}_{\text{total}}$.

The experiments were implemented using the error detection and correction strategy as outlined in Sect. 4.2. We 'confirmed' every 64-th interval using additional N_2 timing samples and aborted attacks as soon as 10 errors had been detected. The average number of errors that were corrected in successful attacks is denoted by $\overline{E}_{\text{total}}$ in Tables 4 and 5.

Our experiments confirm the theory developed in Sect. 4.2. In particular, the efficiency of the attack increases with $\alpha_{\max}$, which is the reason why in Step 2 of the attack we focus on the prime p_i with larger α_i.

It may be surprising that the average sample sizes for 1024-bit moduli and for 3072-bit moduli are comparable although the latter takes almost three times as many individual decisions. The reason is that the average gap $\mathrm{E}(T(u_2) - T(u_1))$ (32) increases linearly in the exponent size, while the standard deviation of $\mathrm{MeanTime}(u, N)$ (34) (in the absence of additional noise, for fixed N) only grows as its square root.

For $b = 2^{\mathsf{ws}}$ with $\mathsf{ws} \geq 8$, we have $\beta_1, \beta_2 \approx 0$ and the algorithmic noise is considerably reduced, whereas the gap remains constant. In this case, the required sample sizes are smaller on average compared to those reported in Tables 4 and 5 (except for ranges where $\overline{N}_2$ is already 1).

5.3 Local timing attacks

We implemented the local timing attacks presented in Sect. 4.3.

For window width w, the computation of the decision rule (39) requires the evaluation of $(2^w + 1)$-dimensional integrals in (38). In contrast to the corresponding attack against Montgomery's algorithm, those integrals cannot easily be determined analytically, which is why we have to resort to numerical methods. However, due to the so-called curse of dimensionality, generic numerical integration algorithms break down completely for $w \geq 3$ in terms of either efficiency or accuracy. We therefore take a step back and consider the stochastic model for the table initialization phase,

$$\begin{aligned} R'_2 &= \lceil \alpha S'_1 S'_1 + \beta U'_2 - S'_2 \rceil, \\ R'_3 &= \lceil \alpha S'_2 S'_1 + \beta U'_3 - S'_3 \rceil, \\ &\vdots \\ R'_{2^w-1} &= \lceil \alpha S'_{2^w-2} S'_1 + \beta U'_{2^w-1} - S'_{2^w-1} \rceil, \end{aligned} \tag{42}$$

and for operation $\mathtt{O}_{j(w+1)} = \mathtt{M}_\theta$ $(j \geq 2, \theta \in \{1, \ldots, 2^w - 1\})$ of the exponentiation phase,

$$R_{j(w+1)} = \lceil \alpha S_{j(w+1)-1} S'_\theta + \beta U_{j(w+1)} - S_{j(w+1)} \rceil.$$

Let $(r'_2, \ldots, r'_{2^w-1}, r_{j(w+1)})$ be a corresponding realization of extra reductions. Let us assume for the moment that we are able to draw samples $(s'_{1,i}, \ldots, s'_{2^w-1,i})$ and $(u'_{2,i} \ldots, u'_{2^w-1,i})$ from (42) for $1 \leq i \leq N'$ which give rise to the given extra reduction vector $(r'_2, \ldots, r'_{2^w-1})$. For N' sufficiently large (we used $N' := 10{,}000$ in our experiments), we obtain the approximation

$$\begin{aligned} &\Pr_\theta\big(R_{j(w+1)} = r_{j(w+1)} \mid R'_2 = r'_2, \ldots, R'_{2^w-1} = r'_{2^w-1}\big) \\ &\quad \approx \frac{1}{N'} \sum_{i=1}^{N'} \Pr\big(R(S_{j(w+1)-1}, s'_{\theta,i}) = r_{j(w+1)}\big) \end{aligned} \tag{43}$$

for all $\theta \in \{1, \ldots, 2^w - 1\}$, where $R(\cdot, \cdot)$ is defined as in (2). The probabilities on the right-hand side of (43) can be computed explicitly using Lemma 3 (i). Since

$$\Pr(R'_2 = r'_2, \ldots, R'_{2^w-1} = r'_{2^w-1})$$

is independent of θ (and > 0), the joint probabilities $\Pr_\theta(R'_2 = r'_2, \ldots, R'_{2^w-1} = r'_{2^w-1}, R_{j(w+1)} = r_{j(w+1)})$ in decision rule (39) can be replaced by the conditional probabilities in (43) without affecting the decision.

A required sample $(s'_1, \ldots, s'_{2^w-1})$ and $(u'_2 \ldots, u'_{2^w-1})$ from (42) giving rise to an extra reduction vector $(r'_2, \ldots, r'_{2^w-1})$ can in principle be generated using rejection sampling. For $w \geq 3$, however, this is way too inefficient for aforementioned reasons. We therefore use an approach akin to Gibbs sampling. First, instead of generating the components of

$$(s'_1, \ldots, s'_{2^w-1}) \quad \text{and} \quad (u'_2 \ldots, u'_{2^w-1})$$

independently and uniformly from $[0, 1)$, we sample them adaptively in the order $s'_1, s'_2, u'_2, \ldots, s'_{2^w-1}, u'_{2^w-1}$, with each choice conditioned on the previous choices (when we reach a dead end, we start over). At this moment, the samples $(s'_1, \ldots, s'_{2^w-1})$ and $(u'_2 \ldots, u'_{2^w-1})$ give rise to $(r'_2, \ldots, r'_{2^w-1})$, but are biased and require some correction. Therefore, we re-sample the elements $s'_1, s'_2, u'_2, \ldots, s'_{2^w-1}, u'_{2^w-1}$ cyclically in this order, with each choice conditioned on the current values of the other variables. In our experiments, we used just 10 such rounds. The final values of $(s'_1, \ldots, s'_{2^w-1})$ are taken as the desired sample. For the next sample, we restart the whole process from the beginning. (Our experiments have shown that continuing the process from the previous sample may lead to a biased distribution.) Although this method is somewhat experimental, our experiments below demonstrate that it is sufficient for our attacks to work. Note that the time complexity of this approach is linear in 2^w, while the complexity of rejection sampling is in general exponential in 2^w.

For the first experiment, we used Diffie–Hellman group `dhe3072` defined in Appendix A.2 of RFC 7919 [16] with 275-bit exponents (cf. Sect. 5.1). The 3072-bit modulus of this group has Barrett parameters $\alpha \approx 0.63$ and $\beta \approx 0.5$ for the base $b = 2$ we used. The results of the experiment are reported in Table 6. For each window size w and sample size N given in the table, we conducted 100 trials of the timing attack. For each trial, a 275-bit secret exponent and N (unknown) input bases were chosen independently at random. We counted attacks with at most 2 errors as successful since one or two errors can easily be corrected by exhaustive search, beginning with the most plausible alternatives. The mean number of errors is denoted by $\overline{E}$.

It can be observed that the attack is exceptionally efficient for window size $w = 1$. The reason is that in this case only the operations $\mathtt{M}_0$ (multiplication by 1) and $\mathtt{M}_1$ (multiplication by the unknown input basis) have to be distinguished, which is easy because for $\mathtt{M}_0$ extra reductions never occur.

For the second experiment, we considered RSA without CRT with 512-bit moduli. We used Barrett's multiplication algorithm with base $b = 2$ and $b = 2^8$, and RSA primes of bit-size 256. In this experiment, we limited ourselves to window size $w = 4$. The results of the experiment for $b = 2$ are reported in Table 7 and for $b = 2^8$ in Table 8. Since the attack is sensitive to the value α of the modulus ('signal'), we conducted trials for several ranges of α. For each row of the table, we conducted 100 trials. For each trial, we used rejection sampling to generate an RSA modulus with α in the given interval and we chose N (unknown) input bases

Table 6 Local timing attack on Diffie–Hellman group `dhe3072` with 275-bit secret exponents (base $b = 2$)

Window width w	Sample size N	Errors $\overline{E}$	Success rate *(in %)*
1	6	6.03	9
	8	2.33	64
	10	0.62	97
	12	0.23	100
2	400	5.76	7
	500	3.25	45
	600	2.19	59
	700	1.43	79
	800	0.92	94
	900	0.57	96
	1000	0.31	100
3	500	6.14	6
	600	3.78	26
	700	2.24	62
	800	1.74	74
	900	1.09	91
	1000	0.92	92
	1100	0.47	100
4	600	5.26	7
	800	2.35	55
	1000	0.93	89
	1200	0.50	98
	1400	0.24	100
5	600	6.89	5
	800	2.93	44
	1000	1.33	88
	1200	0.71	96
	1400	0.23	100

Table 7 Local timing attack on 512-bit RSA with window width $w = 4$ (base $b = 2$)

Signal α	Sample size N	Errors $\overline{E}$	Success rate *(in %)*
[0.4,0.6]	1600	6.10	33
	2000	3.75	46
	2400	2.78	54
	2800	1.93	67
	3200	1.23	82
	3600	1.14	84
	4000	0.74	90
	4400	0.49	95
	4800	0.39	96
[0.6,0.8]	800	4.11	36
	1000	1.67	76
	1200	0.84	90
	1400	0.27	100
[0.8,1.0]	500	3.80	29
	600	1.84	77
	700	0.79	95
	800	0.35	99

Table 8 Local timing attack on 512-bit RSA with window width $w = 4$ (base $b = 2^8$)

Signal α	Sample size N	Errors $\overline{E}$	Success rate *(in %)*
[0.4,0.6]	800	3.25	44
	1200	0.62	93
[0.6,0.8]	600	1.47	84
	800	0.23	100
[0.8,1.0]	400	2.42	56
	500	0.88	95

independently at random. Again, we counted attacks with at most 2 errors as successful.

As in Sect. 5.1, the choice of 512-bit RSA moduli allows a comparison with the results in [23] and [25], Section 6. There Montgomery's multiplication algorithm was applied together with a slightly modified exponentiation algorithm, which resigned on the multiplication by 1 (resp. by Montgomery constant R) when a zero block of the exponent bits was processed. Even for large α the attack on Barrett's multiplication algorithm with $b = 2$ is to some extent less efficient than the attack against Montgomery's algorithm although there only one guessing error was allowed. In contrast, for $\mathsf{ws} = 8$ (and large α), the success rates are similar to those for Montgomery's multiplication algorithm. The results for $\mathsf{ws} > 8$ should be alike because $\beta \approx 0$ in all cases.

For the local timing attack against RSA with CRT, the Barrett parameters $\alpha_1, \beta_1, \alpha_2, \beta_2$ of the unknown primes p_1, p_2 have to be estimated in a pre-step as outlined in Sect. 4.3.2. We successfully verified this procedure in experiments. Since the remaining part of the attack is equivalent to the local timing attack against RSA without CRT, we dispense with reporting additional results on the full attack against RSA with CRT.

A single decision by Decision Strategy 2 depends on the whole table initialization phase but only on one Barrett operation within the exponentiation phase. For given parameters α, β, w, b and N its error probability thus does not depend on the length of the exponent. Since the number of individual decisions increases linearly in the length of the exponent, for longer exponents the sample size N has to be raised to

some extent to keep the expected number of guessing errors roughly constant. Tables 6 to 8 underline that local timing attacks scale well when the exponent length increases. Consider $w = 4$ in Table 6, for instance: increasing N from 1000 to 1400 reduces the error probability of single decisions to $\approx 25\%$, which in turn implies that $N = 1400$ should lead to a similar success probability for exponent length 1024 as $N = 1000$ for exponent length 275.

6 Countermeasures

In Sect. 4, we discussed several attack scenarios against Barrett's multiplication algorithm. It has been pointed out that these attacks are rather similar to those when Montgomery's multiplication algorithm is applied. Consequently, the same countermeasures apply.

The most rigorous countermeasure definitely is when all modular squarings and multiplications within a modular exponentiation need identical execution times. Obviously, then timing attacks and (passive) local timing attacks cannot work. Identical execution times could be achieved by inserting up to two dummy operations per Barrett multiplication if necessary. However, this should be done with care. A potential disadvantage of a dummy operation approach is that dummy operations might be identified by a power attack, and for software implementations on PCs the compiler might unnoticeably cancel the dummy operations if they are not properly implemented. An additional difficulty for software implementations is that secret-dependent branches and memory accesses must be avoided in order to thwart microarchitectural attacks.

It should be noted that for Montgomery's multiplication algorithm, a smarter approach exists: one may completely resign on extra reductions if not only $R > M$ but even $R > 4M$ for modulus M [30]. For Barrett's multiplication algorithm, a similar approach exists, see the variant of this algorithm presented in [12].

Basis blinding and exponent blinding work differently. While basis blinding shall prevent an attacker from learning and controlling the input, exponent blinding shall prevent an attacker from combining information on particular exponent bits from several exponentiations because the exponent is constantly changing.

State-of-the-art (pure) timing attacks on RSA without CRT (or on DH) neither work against basis blinding nor against exponent blinding. While basis blinding suffices to protect RSA with CRT implementations against pure timing attacks, exponent blinding does not suffice; the attack from [26,27] can easily be transferred to Barrett's multiplication algorithm. On the other hand, sole basis blinding does not prevent local timing attacks as shown in Sects. 4.3 and 5.3, whereas exponent blinding counteracts these attacks.

Principally, state-of-the-art knowledge might be used to determine minimal countermeasures, but we recommend to stay on the safe side by applying combinations of blinding techniques (at least basis blinding and exponent blinding). Since blinding also counteracts other types of attacks, we recommend to use blinding techniques even if the implementation does not have timing differences.

7 Conclusion

We have thoroughly analysed the stochastic behaviour of Barrett's multiplication algorithm when applied in modular exponentiation algorithms. Unlike Montgomery's multiplication algorithm, Barrett's multiplication algorithm does not only allow one but even two extra reductions, a feature, which increases the mathematical difficulties considerably. All known timing attacks and local timing attacks against Montgomery's multiplication algorithm were adapted to Barrett's multiplication algorithm, but specific features require additional attack substeps when RSA with CRT is attacked. Moreover, for timing attacks against RSA without CRT and against DH, we developed an efficient look-ahead strategy. Extensive experiments confirmed our theoretical results. Fortunately, effective countermeasures exist.

Acknowledgements We thank Friederike Laus for her careful reading of an earlier iteration of this document and the anonymous reviewer for his thoughtful comments on the submitted version. Both helped us to improve the presentation of the paper.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Aciiçmez, O., Schindler, W.: A major vulnerability in RSA implementations due to microarchitectural analysis threat. IACR Cryptology ePrint Archive **2007**, 336 (2007). http://eprint.iacr.org/2007/336
2. Aciiçmez, O., Schindler, W.: A vulnerability in RSA implementations due to instruction cache analysis and its demonstration on openssl. In: T. Malkin (ed.) Topics in Cryptology-CT-RSA 2008,

The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8–11, 2008. Proceedings, Lecture Notes in Computer Science, vol. 4964, pp. 256–273. Springer (2008). https://doi.org/10.1007/978-3-540-79263-5_16
3. Aciiçmez, O., Schindler, W., Koç, Ç.K.: Improving Brumley and Boneh timing attack on unprotected SSL implementations. In: V. Atluri, C.A. Meadows, A. Juels (eds.) Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7–11, 2005, pp. 139–146. ACM (2005). http://doi.acm.org/10.1145/1102120.1102140
4. Barrett, P.: Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In: A.M. Odlyzko (ed.) Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings, Lecture Notes in Computer Science, vol. 263, pp. 311–323. Springer (1986). https://doi.org/10.1007/3-540-47721-7_24
5. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: a fresh approach to numerical computing. SIAM Rev. **59**(1), 65–98 (2017). https://doi.org/10.1137/141000671
6. Bosselaers, A., Govaerts, R., Vandewalle, J.: Comparison of three modular reduction functions. In: D.R. Stinson (ed.) Advances in Cryptology—CRYPTO'93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22–26, 1993, Proceedings, Lecture Notes in Computer Science, vol. 773, pp. 175–186. Springer (1993). https://doi.org/10.1007/3-540-48329-2_16
7. Brumley, D., Boneh, D.: Remote timing attacks are practical. In: Proceedings of the 12th USENIX Security Symposium, Washington, D.C., USA, August 4–8, 2003. USENIX Association (2003). https://www.usenix.org/conference/12th-usenix-security-symposium/remote-timing-attacks-are-practical
8. Brumley, D., Boneh, D.: Remote timing attacks are practical. Comput. Netw. **48**(5), 701–716 (2005). https://doi.org/10.1016/j.comnet.2005.01.010
9. Buonocore, A., Pirozzi, E., Caputo, L.: A note on the sum of uniform random variables. Stat. Probab. Lett. **79**(19), 2092–2097 (2009). https://doi.org/10.1016/j.spl.2009.06.020
10. Cohen, H., Frey, G., Avanzi, R., Doche, C., Lange, T., Nguyen, K., Vercauteren, F. (eds.): Handbook of Elliptic and Hyperelliptic Curve Cryptography. Chapman and Hall/CRC, Boca Raton (2005). https://doi.org/10.1201/9781420034981
11. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptol. **10**(4), 233–260 (1997). https://doi.org/10.1007/s001459900030
12. Dhem, J.: Design of an efficient public-key cryptographic library for RISC-based smart cards. Ph.D. thesis, Université Catholique de Louvain (1998)
13. Dhem, J., Koeune, F., Leroux, P., Mestré, P., Quisquater, J., Willems, J.: A practical implementation of the timing attack. In: J. Quisquater, B. Schneier (eds.) Smart Card Research and Applications, This International Conference, CARDIS'98, Louvain-la-Neuve, Belgium, September 14–16, 1998, Proceedings, Lecture Notes in Computer Science, vol. 1820, pp. 167–182. Springer (1998). https://doi.org/10.1007/10721064_15
14. Dugardin, M., Guilley, S., Danger, J., Najm, Z., Rioul, O.: Correlated extra-reductions defeat blinded regular exponentiation. In: B. Gierlichs, A.Y. Poschmann (eds.) Cryptographic Hardware and Embedded Systems—CHES 2016—18th International Conference, Santa Barbara, CA, USA, August 17–19, 2016, Proceedings, Lecture Notes in Computer Science, vol. 9813, pp. 3–22. Springer (2016). https://doi.org/10.1007/978-3-662-53140-2_1
15. Fieker, C., Hart, W., Hofmann, T., Johansson, F.: Nemo/hecke: Computer algebra and number theory packages for the julia programming language. In: Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC'17, pp. 157–164. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3087604.3087611
16. Gillmor, D.K.: Negotiated finite field Diffie-Hellman ephemeral parameters for transport layer security (TLS). RFC (2016). https://doi.org/10.17487/RFC7919
17. Hoeffding, W., Robbins, H.: The central limit theorem for dependent random variables. Duke Math. J. **15**(3), 773–780 (1948). https://doi.org/10.1215/S0012-7094-48-01568-3
18. Knezevic, M., Vercauteren, F., Verbauwhede, I.: Faster interleaved modular multiplication based on Barrett and Montgomery reduction methods. IEEE Trans. Comput. **59**(12), 1715–1721 (2010). https://doi.org/10.1109/TC.2010.93
19. Kocher, P.C.: Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems. In: N. Koblitz (ed.) Advances in Cryptology—CRYPTO'96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18–22, 1996, Proceedings, Lecture Notes in Computer Science, vol. 1109, pp. 104–113. Springer (1996). https://doi.org/10.1007/3-540-68697-5_9
20. Menezes, A., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
21. Montgomery, P.L.: Modular multiplication without trial division. Math. Comput. **44**, 510–521 (1985)
22. Schindler, W.: A timing attack against RSA with the Chinese remainder theorem. In: Ç.K. Koç, C. Paar (eds.) Cryptographic Hardware and Embedded Systems—CHES 2000, Second International Workshop, Worcester, MA, USA, August 17–18, 2000, Proceedings, Lecture Notes in Computer Science, vol. 1965, pp. 109–124. Springer (2000). https://doi.org/10.1007/3-540-44499-8_8
23. Schindler, W.: A combined timing and power attack. In: D. Naccache, P. Paillier (eds.) Public Key Cryptography, 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002, Paris, France, February 12–14, 2002, Proceedings, Lecture Notes in Computer Science, vol. 2274, pp. 263–279. Springer (2002). https://doi.org/10.1007/3-540-45664-3_19
24. Schindler, W.: Optimized timing attacks against public key cryptosystems. Stat- Risk Model. **20**(1–4), 191–210 (2002). https://doi.org/10.1524/strm.2002.20.14.191
25. Schindler, W.: On the optimization of side-channel attacks by advanced stochastic methods. In: S. Vaudenay (ed.) Public Key Cryptography—PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23–26, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3386, pp. 85–103. Springer (2005). https://doi.org/10.1007/978-3-540-30580-4_7
26. Schindler, W.: Exclusive exponent blinding may not suffice to prevent timing attacks on RSA. In: T. Güneysu, H. Handschuh (eds.) Cryptographic Hardware and Embedded Systems - CHES 2015—17th International Workshop, Saint-Malo, France, September 13–16, 2015, Proceedings, Lecture Notes in Computer Science, vol. 9293, pp. 229–247. Springer (2015). https://doi.org/10.1007/978-3-662-48324-4_12
27. Schindler, W.: Exclusive exponent blinding is not enough to prevent any timing attack on RSA. J. Cryptogr. Eng. **6**(2), 101–119 (2016). https://doi.org/10.1007/s13389-016-0124-7
28. Schindler, W., Koeune, F., Quisquater, J.: Improving divide and conquer attacks against cryptosystems by better error detection/correction strategies. In: B. Honary (ed.) Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17–19, 2001, Proceedings, Lecture Notes in Computer Science, vol. 2260, pp. 245–267. Springer (2001). https://doi.org/10.1007/3-540-45325-3_22
29. Schindler, W., Koeune, F., Quisquater, J.: Unleashing the full power of timing attack. Tech. Rep. CG 2001/3, Université Catholique de Louvain, Crypto Group (2001)

30. Walter, C.D.: Precise bounds for montgomery modular multiplication and some potentially insecure RSA moduli. In: B. Preneel (ed.) Topics in Cryptology-CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18–22, 2002, Proceedings, Lecture Notes in Computer Science, vol. 2271, pp. 30–39. Springer (2002). https://doi.org/10.1007/3-540-45760-7_3