Exclusive Exponent Blinding May Not Suffice to Prevent Timing Attacks on RSA

Werner Schindler

Bundesamt für Sicherheit in der Informationstechnik (BSI)
Godesberger Allee 185–189
53175 Bonn, Germany
Werner.Schindler@bsi.bund.de

Abstract. The references [9, 3, 1] treat timing attacks on RSA with CRT and Montgomery's multiplication algorithm in unprotected implementations. It has been widely believed that exponent blinding would prevent any timing attack on RSA. At cost of significantly more timing measurements this paper extends the before-mentioned attacks to RSA with CRT when Montgomery's multiplication algorithm and exponent blinding are applied. Simulation experiments are conducted, which confirm the theoretical results. Effective countermeasures exist. In particular, the attack efficiency is higher than in the previous version [12] while large parts of both papers coincide.

Keywords: Timing attack, RSA, CRT, exponent blinding, Montgomery's multiplication algorithm.

1 Introduction

⋆ In 1996 Paul Kocher introduced timing analysis [6]. In particular, [6] presents a timing attack on an unprotected RSA implementation, which does not apply the Chinese Remainder Theorem (CRT). Reference [9] introduced a new timing attack on RSA implementations, which apply CRT and Montgomery's multiplication algorithm [8]. This attack was extended to OpenSSL (RSA, CRT, sliding window exponentiation algorithm, Montgomery's multiplication algorithm) [3], and later optimized [1]. Also [5, 9–11] consider timing attacks on RSA implementations that apply Montgomery's multiplication algorithm. All these attacks target unprotected RSA implementations.

Besides presenting the first timing attack on RSA (without CRT) [6] proposes various countermeasures (Section 10), including exponent blinding where a random multiple of Euler's ϕ function of the modulus is added to the secret exponent. Since then (exclusive) exponent blinding has widely been assumed to be effective to prevent (any type of) timing attacks on RSA, at least no successful timing attacks against exponent blinding have been known. The present paper extends the timing attack from [9] to RSA implementations, which apply exponent blinding, proving that exclusive exponent blinding (without additional

countermeasures) does not always prevent timing attacks on RSA. However, the presence of exponent blinding increases the number of timing measurements enormously.

In Section 2 the targeted implementation is described (RSA with CRT, square & multiply, Montgomery's multiplication algorithm, exponent blinding), assumptions are formulated and justified. Section 3 contains the theoretical foundations of our attack while in Section 4 the attack is specified and experimental results are given. Moreover, the attack is adjusted to table-based exponentiation algorithms, and effective countermeasures are proposed.

In this paper the attack efficiency is higher than in [12]. For several proofs in Section 3 and an parameter estimation process we refer to [12]. Apart from that and from editorial improvements both papers essentially coincide in large parts.

2 Modular Exponentiation with Montgomery's Multiplication Algorithm

In this section we describe the targeted RSA implementation. More precisely, we begin with the modular arithmetic, and finally we specify the modular exponentiation algorithm. Moreover, two assumptions are formulated and analysed, which will be applied later.

Montgomery's multiplication algorithm (MM) [8] fits perfectly to the hardware architecture of a computer, smart card or microcontroller since modulo operations and divisions only have to be carried out for moduli and divisors, which are powers of 2.

Definition 1. *For a positive integer $M > 1$ we set $Z_M := \{0, 1, \ldots, M-1\}$. We write $a \equiv b \bmod M$ if $(a - b)$ is a multiple of M. The term $b(\bmod M)$ denotes the unique element in Z_M, which is congruent to b modulo M.*

For an odd modulus M the integer $R := 2^t > M$ is called Montgomery's constant, and $R^{-1} \in Z_M$ denotes its multiplicative inverse modulo M. Moreover, $M^* \in Z_R$ satisfies the integer equation $RR^{-1} - MM^* = 1$.

On input (a, b) Montgomery's algorithm returns $\mathrm{MM}(a, b; M) := abR^{-1}(\bmod M)$. This value is computed with a multiprecision version of Montgomery's multiplication algorithm, which is adjusted to the particular device. More precisely, let ws denote the word size for the arithmetic operations (typically, $ws = 8, 16, 32, 64$), which divides the exponent t of R. Further, $r = 2^{ws}$, so that in particular $R = r^v$ with $v = t/ws$ (numerical example: $(ws, t, v) = (16, 1024, 64)$). In Algorithm 1 a, b and s are expressed in the r-adic representation. That is, $a = (a_{v-1}, ..., a_0)_r$, $b = (b_{v-1}, ..., b_0)_r$ and $s = (s_{v-1}, ..., s_0)_r$. Finally, $m^* = M^* \pmod r$. In particular, $MM^* = RR^{-1} - 1 \equiv -1 \bmod R$ and thus $m^* \equiv -M^{-1} \bmod r$.

Algorithm 1. Montgomery's multiplication algorithm (MM), multiprecision variant

1. Input: $a, b \in \mathrm{Z_M}$

2. $s := 0$
3. For $i = 0$ to $v - 1$ do {
   $u := (s + a_i b_0) m^* (\mathrm{mod}\ r)$
   $s := (s + a_i b + uM)/r$
   }
4. If $(s \geq M)$ then $s := s - M$ [= extra reduction (ER)]
5. return s $(= abR^{-1} (\mathrm{mod}\ M) = \mathrm{MM}(a, b; M))$

After Step 3 $s \equiv abR^{-1} \bmod M$ 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. Whether an ER is necessary does not depend on the chosen multiprecision variant but only on the quadruple (a, b, M, R) [9], Remark 1. This allows to consider the case $ws = t$ (i.e. $v = 1$) when analyzing the stochastic behaviour of the ERs in modular exponentiations.

Algorithm 2 combines Montgomery's multiplication algorithm with the square & multiply exponentiation algorithm.

Algorithm 2. Square & multiply with Montgomery's algorithm (s&m, MM)

```
Computes  y ↦ y^d (mod M) for  d = (d_{w-1}, ..., 0)_2
  temp :=  y_R  := MM(y, R^2 (mod M); M)      (Pre-multiplication)
  for i=w-1 down to 0 do {
    temp := MM(temp, temp; M)
    if (d_i=1) then temp := MM(temp, y_R; M)
    }
  MM(temp, 1; M)                              (Post-multiplication)
  return temp  ( = y^d (mod M) )
```

As usual, $n = p_1 p_2$ and R denotes the Montgomery constant while $\mathrm{MM}(a, b; n) := abR^{-1} (\mathrm{mod}\ n)$ stands for the Montgomery multiplication of a and b. The computation of $v = y^d (\mathrm{mod}\ n)$ is performed in several steps:

Algorithm 3. RSA, CRT, s&m, MM, exponent blinding

1. (a) Set $y_1 := y (\mathrm{mod}\ p_1)$ and $d_1 := d (\mathrm{mod}\ (p_1 - 1))$
   (b) (Exponent blinding) Generate a random number $r_1 \in \{0, 1, \ldots, 2^{eb} - 1\}$ and compute the blinded exponent $d_{1,b} := d_1 + r_1 \phi(p_1) = d_1 + r_1 (p_1 - 1)$.
   (c) Compute $v_1 := y_1^{d_{1,b}} (\mathrm{mod}\ p_1)$ with Algorithm 2 $(M = p_1)$.
2. (a) Set $y_2 := y (\mathrm{mod}\ p_2)$ and $d_2 := d (\mathrm{mod}\ (p_2 - 1))$
   (b) (Exponent blinding) Generate a random number $r_2 \in \{0, 1, \ldots, 2^{eb} - 1\}$ and compute the blinded exponent $d_{2,b} := d_2 + r_2 \phi(p_2) = d_2 + r_2 (p_2 - 1)$.
   (c) Compute $v_2 := y_2^{d_{2,b}} (\mathrm{mod}\ p_2)$ with Algorithm 2 $(M = p_2)$.
3. (Recombination) Compute $v := y^d (\mathrm{mod}\ n)$ from (v_1, v_2), e.g. with Garner's algorithm: $v := v_1 + p_1 \left(p_1^{-1} (\mathrm{mod}\ p_2) \cdot (v_2 - v_1) (\mathrm{mod}\ p_2) \right) (\mathrm{mod}\ n)$

Assumption 1. For fixed modulus M and fixed Montgomery constant R

$$\text{Time}\left(\text{MM}(a,b;M)\right) \in \{c, c+c_{\text{ER}}\} \quad \text{for all } a,b \in Z_M\,, \tag{1}$$

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

Remark 1. [Justification of Assumption 1]
(i) Since the divisions and the modular reductions in Step 3 of Algorithm 1 can be realized by shifts and masking operations the calculations within the for-loop are essentially integer additions and integer multiplications or parts thereof, respectively. For fixed M and R the time per iteration of the for-loop should be constant. Since usually $\log_2(M) \approx \log_2(R)$ for known input attacks the leading words a_{v-1} and b_{v-1} are de facto always non-zero, at least if $ws \geq 16$, and thus may expect that (1) is fulfilled.
(ii) Our timing attack is an adapted chosen input attack, for which in the course of the attack in many Montgomery multiplications one factor has one or more leading zero words. For smart cards and microcontrollers one might assume that this feature may not violate Assumption 1 since optimizations of rare events (within the normal use of the device) seem to be unlikely.
(iii) On a PC cryptographic software might process small operands (i.e., those with leading zero-words) in Step 3 of Algorithm 1 differently, e.g. because different integer multiplication algorithm is applied (e.g., OpenSSL: normal multiplication vs. Karatsuba multiplication [3, 1]). Such effects, however, may complicate our attack but should not prevent it [3, 1].

Assumption 2. Assumption 1 holds for the modular multiplications $(\bmod p_1)$ and $(\bmod p_2)$ with identical time constants c and c_{ER}. The attacker knows the values c and c_{ER}.

Remark 2. (i) Usually, the r-adic representations of p_1 and p_2 comprise the same number of words, i.e. $\lceil \log_2(p_1)/ws \rceil = \lceil \log_2(p_2)/ws \rceil$, and R is identical in both cases. With regard to Remark 1 this justifies the first claim of Assumption 2. (If the number of words should be different we may expect unequal triplets $(R_1, c_1, c_{\text{ER},1})$ and $(R_2, c_2, c_{\text{ER},2})$, which would complicate the attack.)
(ii) In the best case (from the attacker's point of view) the attacker either knows c and c_{ER} or is able to determine them precisely with a simulation tool. Otherwise, he may estimate both values, see [12], Subsect. 4.4.

3 Theoretical Background of our Attack

This section contains the theoretical foundations of our attack. The main results are the mean value and the variance of the execution time of Algorithm 3 (Subsection 3.1 and Subsection 3.2) and the distinguisher, which allows to decide whether a given interval contains / does not contain a multiple of p_1 or p_2. (Subsection 3.3).

3.1 Exponentiation ($\mathbf{mod}\ p_i$)

In Subsection 3.1 we consider the stochastic timing behaviour of the exponentiations modulo p_1 and modulo p_2. More precisely, we focus on the for-loop in Algorithm 2 when applied by Step i(c) of Algorithm 3 with $M = p_i$ for $i = 1, 2$. By Step i(b) of Algorithm 3 the blinding factor r_i is a randomly selected eb-bit number, i.e. $r_i \in \{0, \ldots, 2^{eb} - 1\}$ for $i = 1, 2$. We interpret the measured execution times as realizations of random variables.

Definition 2. *Random variables are denoted by capital letters, and realizations (i.e., values taken on) of these random variables are denoted with the corresponding small letter. The abbreviation 'iid' stands for 'independent and identically distributed'. For a random variable Y the terms $E(Y)$, $E(Y^2)$ and $\mathrm{Var}(Y)$ denote its expectation (mean), its second moment and its variance, respectively. The term $Y \sim N(\mu, \sigma_2)$ means that the random variable N is normally distributed with mean μ and variance σ^2. The cumulative distribution of the standard normal distribution $N(0, 1)$ is given by $\Phi(x) := (2\pi)^{-1/2} \int_{-\infty}^{x} e^{-t^2/2}\, dt$.*

The distinguisher and the attack in Section 4 consider input values of the form $y = uR^{-1} (\mathrm{mod}\, n)$. A simple calculation shows that the pre-multiplication step in Algorithm 2 transforms the input value y into $y_{R,i} := u(\mathrm{mod}\ p_i)$ ([9], Sect. 3, after formula (5)). Consequently, we interpret the execution time of the for-loop in Algorithm 2 as a realization of a random variable $Z_i(u)$. With this notation

$$Z_i(u) := (Q_i + M_i)c + X_i\, c_{\mathrm{ER}} \tag{2}$$

expresses the random computation time for the exponentiation (mod p_i) in terms of the random variables Q_i, M_i and X_i. The random variables Q_i and M_i denote the random number of squarings and multiplications within the for loop in Step i(c) while X_i quantifies the number of extra reductions (ERs) in these squarings and multiplications ($i = 1, 2$). Unfortunately, the random variables Q_i, M_i and X_i are not independent.

The main goal of this subsection is to calculate $E(Z_i(u))$ and $\mathrm{Var}(Z_i(u))$. By definition

$$E\left(Z_i^v(u)\right) = \sum_{q_j} \sum_{m_k} \sum_{x_r} P(Q_i = q_i, M_i = m_k, X_i = x_r)\left((q_i + m_k)c + x_r\, c_{\mathrm{ER}}\right) =$$

$$\sum_{q_j} P(Q_i = q_j) \sum_{m_k} P(M_i = m_k \mid Q_i = q_j) \sum_{x_r} P(X_i = x_r \mid Q_i = q_j, M_i = m_k) \quad \times$$

$$\times \quad \left((q_i + m_k)c + x_r\, c_{\mathrm{ER}}\right)^v . \tag{3}$$

Clearly, $x_r \in \{0, \ldots, q_j + m_k\}$, $m_k \in \{0, \ldots, q_j\}$ and $q_j \in \{k-1, \ldots, k+eb-1\}$. Lemma 1 collects several facts, which will be needed in the following. Recall that $p_i < R$.

Lemma 1. *As in Section 2 the term y_i stands for $y(\mathrm{mod}\, p_i)$.*
(i) For $y := uR^{-1}(\mathrm{mod} n)$ the MM-transformed basis for the exponentiation $(\mathrm{mod}\, p_i)$ equals $u_i' := u(\mathrm{mod}\, p_i)$.

(ii) If $d_{i,b}$ is a k_i'-bit integer the computation of $y_i^{d_{i,b}} (\mathrm{mod}\, p_i)$ needs $q_i := k_i' - 1$ squarings and $m_i := \mathrm{ham}(d_{i,b}) - 1$ multiplications where $\mathrm{ham}(\cdot)$ denotes the Hamming weight of its argument.
(iii) The (conditional) random variable $(X_i \mid Q_i = q_i, M_i = m_i)\, c_{\mathrm{ER}}$ quantifies the overall random execution time for all extra reductions if $Q_i = q_i$ and $M_i = m_i$. Let

$$p_{i*} := \frac{p_i}{3R}\,, \quad p_{i(u')} := \frac{u_i'}{2p_i}\,, \quad \mathrm{cov}_{i,\mathrm{MS}(\mathrm{u_i'})} := 2p_{i(u')}^3 p_{i*} - p_{i(u')} p_{i*}\,, \tag{4}$$

$$\mathrm{cov}_{i,\mathrm{SM}(\mathrm{u_i'})} := \frac{9}{5} p_{i(u')} p_{i*}^2 - p_{i(u')} p_{i*}\,, \quad \mathrm{cov}_{i,\mathrm{SS}} := \frac{27}{7} p_{i*}^4 - p_{i*}^2\,. \tag{5}$$

The random variable $(X_i \mid Q_i = q_i, M_i = m_i)$ is normally distributed with expectation

$$E(X_i \mid Q_i = q_i, M_i = m_i) = q_i p_{i*} + m_i p_{i(u')} \qquad \text{and variance} \tag{6}$$

$$\begin{aligned}\mathrm{Var}(X_i \mid Q_i = q_i, M_i = m_i) = {} & q_i p_{i*}(1 - p_{i*}) + m_i p_{i(u')}(1 - p_{i(u')}) + \\ & 2m_i \mathrm{cov}_{i,\mathrm{SM}(\mathrm{u_i'})} + 2(m_i - 1)\mathrm{cov}_{i,\mathrm{MS}(\mathrm{u_i'})} + 2(q_i - m_i)\mathrm{cov}_{i,\mathrm{SS}}\end{aligned} \tag{7}$$

(iv) The random variable $(M_i \mid Q_i = q_i)$ quantifies the random number of multiplications if $Q_i = q_i$. It is approximately $N(q_i/2, q_i/4)$-distributed. In particular, $E(M_i^2 \mid Q_i = q_i) = \frac{1}{4}(q_i + q_i^2)$.

Proof. see [12], proof of Lemma 1.

Theorem 1. *Combining the previous results we obtain*

$$E\left(Z_i(u)\right) = E(Q_i)\left(\frac{3}{2}c + \left(p_{i*} + \frac{1}{2}p_{i(u')}\right) c_{\mathrm{ER}}\right) \tag{8}$$

and

$$\begin{aligned}\mathrm{Var}\left(Z_i(u)\right) = {} & \mathrm{Var}(Q_i)\left(\frac{3}{2}c + (p_{i*} + \frac{1}{2}p_{i(u')})\, c_{\mathrm{ER}}\right)^2 \\ & + E(Q_i)\Big(\frac{1}{4}c^2 + \frac{1}{2}p_{i(u')} c\, c_{\mathrm{ER}} + \big(p_{i*}(1 - p_{i*}) + \frac{1}{2}p_{i(u')}(1 - p_{i(u')}) \\ & + 2p_{i(u')}^3 p_{i*} + \frac{9}{5}p_{i(u')}p_{i*}^2 - 2p_{i(u')}p_{i*} + \frac{27}{7}p_{i*}^4 - p_{i*}^2 + \frac{1}{4}p_{i(u')}){c_{\mathrm{ER}}}^2\Big) \\ & - 2(2p_{i(u')}^3 p_{i*} - 2p_{i(u')}p_{i*}){c_{\mathrm{ER}}}^2\,.\end{aligned} \tag{9}$$

Proof. see [12], proof of Theorem 1.

Lemma 2 provides explicit expressions for $E(Q_i)$ and $\mathrm{Var}(Q_i)$, which may be substituted into (8) and (9). Note that $p_i < 2^k \leq R$.

Lemma 2. *Let p_i be a k-bit number, and let $\gamma_i := p_i/2^k$.*
(i) Unless eb is artificially small approximately

$$E(Q_i) = (k - 1) + eb - \frac{1}{\gamma_i} \tag{10}$$

$$\mathrm{Var}(Q_i) = \frac{3}{\gamma_i} - \frac{1}{\gamma_i^2} \tag{11}$$

(ii) In particular, $E(Q_i)$ is monotonously increasing in γ_i and assumes values in $(k-1+eb-2, k-1+eb-1)$. The variance $\mathrm{Var}(Q_i)$ assumes values in $(2, 2.25]$. The maximum value 2.25 is taken on for $\gamma_i = 2/3$. If $2^k = R$ (typical case) then $\gamma_i = 3p_{i*}$.

Proof. see [12], proof of Lemma 2.

Remark 3. (i) Setting $\mathrm{Var}(Q_i) = 0$ and $E(Q_i) = k-1$ Theorem 1 provides the formulae for non-blinded implementations.
(ii) Numerical experiments verify that (11) approximates $\mathrm{Var}(Q_i)$ very well.

Table 1 evaluates the terms (8), (9), (10) and (11) for exemplary parameter sets.

				$\frac{u'}{p_i} = 0.0$		$\frac{u'}{p_i} = 0.5$		$\frac{u'}{p_i} = 1.0$	
$\log_2(R)$	eb	$\frac{p_i}{R}$	c_{ER}	$E(Z_i(u))$	$\mathrm{Var}(Z_i(u))$	$E(Z_i(u))$	$\mathrm{Var}(Z_i(u))$	$E(Z_i(u))$	$\mathrm{Var}(Z_i(u))$
512	64	0.75	$0.03\,c$	$864.8\,c$	$148.5\,c^2$	$866.4\,c$	$150.2\,c^2$	$868.0\,c$	$151.9\,c^2$
512	64	0.80	$0.03\,c$	$865.2\,c$	$148.5\,c^2$	$866.9\,c$	$150.3\,c^2$	$868.7\,c$	$152.0\,c^2$
512	64	0.85	$0.03\,c$	$865.6\,c$	$148.4\,c^2$	$867.4\,c$	$150.3\,c^2$	$869.3\,c$	$152.2\,c^2$
512	64	0.75	$0.05\,c$	$867.7\,c$	$148.7\,c^2$	$870.4\,c$	$151.5\,c^2$	$873.0\,c$	$154.3\,c^2$
512	64	0.80	$0.05\,c$	$868.3\,c$	$148.7\,c^2$	$871.1\,c$	$151.6\,c^2$	$874.0\,c$	$154.6\,c^2$
512	64	0.85	$0.05\,c$	$868.9\,c$	$148.6\,c^2$	$871.9\,c$	$151.7\,c^2$	$875.0\,c$	$154.9\,c^2$
1024	64	0.75	$0.03\,c$	$1636.6\,c$	$276.6\,c^2$	$1639.7\,c$	$279.7\,c^2$	$1642.8\,c$	$282.8\,c^2$
1024	64	0.80	$0.03\,c$	$1637.3\,c$	$276.6\,c^2$	$1640.6\,c$	$279.9\,c^2$	$1643.8\,c$	$283.2\,c^2$
1024	64	0.85	$0.03\,c$	$1638.0\,c$	$276.5\,c^2$	$1641.4\,c$	$280.0\,c^2$	$1644.9\,c$	$283.5\,c^2$
1024	64	0.75	$0.05\,c$	$1642.1\,c$	$276.9\,c^2$	$1647.2\,c$	$282.1\,c^2$	$1652.3\,c$	$287.4\,c^2$
1024	64	0.80	$0.05\,c$	$1643.1\,c$	$276.8\,c^2$	$1648.5\,c$	$282.4\,c^2$	$1654.0\,c$	$288.0\,c^2$
1024	64	0.85	$0.05\,c$	$1644.1\,c$	$276.8\,c^2$	$1649.9\,c$	$282.7\,c^2$	$1655.7\,c$	$288.6\,c^2$

Table 1. Expectation and variance for several sets of parameters $(R, eb, p_i/R, c_{\mathrm{ER}}/c)$

3.2 Further Arithmetic Operations and Noise

The random variables $Z_1(u)$ and $Z_2(u)$ quantify the random timing behaviour of the for-loop in Algorithm 2 when called in Step 1(c) and Step 2(c) of Algorithm 3, respectively. However, the computation of $(uR^{-1} (\mathrm{mod}\, n))^d (\mathrm{mod}\, n)$ requires several further steps: Step 1(a) and Step 2(a) (reduction modulo p_i), Step 1(b) and Step 2(b) (exponent blinding), Step 1(c) and Step 2(c) (here: pre-multiplication and post-multiplication of Algorithm 2), Step 3 (recombination), time for input and output etc. In analogy to Subsection 3.1 we view the required overall execution time for these before-mentioned steps as a realization of a random variable $Z_3(u)$.

It seems reasonable to assume that the time for input and output of data, for recombination and blinding as well as the reduction $(\mathrm{mod}\, p_i)$ in Step 1(a)

and Step 2(a) of Algorithm 3 do not (or at most weakly) depend on u. The postprocessing step in Algorithm 2 never needs an ER. (By [13], Theorem 1, in Algorithm 1, after Step 3 we have $s \leq M + \text{temp} \cdot r^{-v} < M + 1$, and thus $s \leq M$. If $s = M$ then temp $= 0$ after the extra reduction, which can only happen if u is a multiple of $M = p_i$ but then $y_R \equiv uR^{-1}R \equiv 0 \bmod p_i$, and Algorithm 2 does not need any extra reduction at all.) In the pre-multiplication in Algorithm 2 an ER may occur or not. Altogether, we may assume

$$E(Z_3(u)) \approx \bar{z}_3 \qquad \text{for all } u \in \mathrm{Z_n} \quad \text{and} \tag{12}$$

$$\mathrm{Var}(Z_3(u)) \ll \mathrm{Var}(Z_1(u)), \mathrm{Var}(Z_2(u)) \tag{13}$$

Assumption 3. In the following we assume $E(Z_3(u)) = \bar{z}_3$ for all u and interpret the centered random variable $Z_3(u) - \bar{z}_3$ as part of the noise, which is captured by the random variable N_e. If $\mathrm{Var}(N_e) = \sigma_N^2 > 0$ we assume $N_e \sim N(\mu_N, \sigma_N^2)$ while $\sigma_N^2 = 0$ means 'no noise' and $N_e = \bar{z}_3$ with probability 1.

Remark 4. [Justification of Assumption 3]
The part of Assumption 3, which concerns $Z(u)$, follows from the preceding arguments. Measurement errors are usually assumed to be Gaussian distributed, and if the noise comprises of several contributions (of comparable size) the Central Limit Theorem may be used as an additional argument for the assumption of Gaussian noise. However, the core of our attack is a distinguisher, which separates two probability distributions with different mean values. As long as the noise is assumed to be data-independent the distinguisher should work for arbitrary noise distributions (maybe the number of timing measurements varies).

3.3 The Distinguisher

Now we derive a distinguisher, which will be the core of our attack (to be developed in Section 4). With regard to the preceding the overall random execution time for input u is described by the random variable

$$Z(u) = Z_1(u) + Z_2(u) + \bar{z}_3 + N_e \,. \tag{14}$$

In the following we assume

$$0 < u_1 < u_2 < n \qquad \text{and } u_2 - u_1 \ll p_1, p_2 \,. \tag{15}$$

Theorem 1 implies

$$\begin{aligned} E\left(Z(u_2) - Z(u_1)\right) &= E\left(Z_1(u_2) - Z_1(u_1)\right) + E\left(Z_2(u_2) - Z_2(u_1)\right) \qquad (16)\\ &= \frac{1}{2} \sum_{i=1}^{2} E(Q_i) \left(p_{i(u'_{(2)})} - p_{i(u'_{(1)})}\right) c_{\mathrm{ER}} \quad \text{with } u'_{(j)} = u_j (\bmod p_i) \end{aligned}$$

As in [9] we distinguish between three cases:
Case A: The interval $\{u_1 + 1, \ldots, u_2\}$ does not contain a multiple of p_1 or p_2.

Case B: The interval $\{u_1+1,\ldots,u_2\}$ contains a multiple of p_s but not of p_{3-s}.
Case C: The interval $\{u_1+1,\ldots,u_2\}$ contains a multiple of p_1 and p_2.

Let's have a closer look at (16). By (4)

$$p_{i(u'_{(2)})} - p_{i(u'_{(1)})} \begin{cases} = \frac{u_2-u_1}{2R}\, c_{\mathrm{ER}} \approx 0 & \text{Case A, Case B (for } i \neq s\text{)} \\ \approx -\frac{p_i}{2R}\, c_{\mathrm{ER}} & \text{Case B (for } i = s\text{), Case C} \end{cases} \tag{17}$$

Further,

$$\begin{aligned} E(Q_i) &= k_i + eb - 1 - \gamma_i^{-1} = 2^{\lceil \log_2(p_i)\rceil} + eb - 1 - \frac{2^{k_i}}{p_i} \\ &= \log_2(R) + (\lceil \log_2(p_i)\rceil - \log_2(R)) + eb - 1 - \frac{R}{p_i}\cdot\frac{2^{k_i}}{R} \end{aligned} \tag{18}$$

where $\lceil x\rceil$ denotes the smallest integer $\geq x$. At the beginning of our attack we have no concrete information on the size of the primes p_1 and p_2, and thus we use the rough approximation

$$p_1, p_2 \approx \sqrt{n} \quad \text{and set } \beta := \frac{\sqrt{n}}{R}\,. \tag{19}$$

With approximation (19) formula (18) simplifies to

$$E(Q_i) \approx \log_2(R) + eb - 1 - \beta^{-1} \qquad \text{if } \sqrt{0.5} < \beta < 1\,, \text{ and similarly} \tag{20}$$

$$\mathrm{Var}(Q_i) \approx 3\beta^{-1} - \beta^{-2} \qquad \text{if } \sqrt{0.5} < \beta < 1 \tag{21}$$

since $k_i = \lceil \log_2(p_i)\rceil = \log_2(R)$ then. Finally (17) and (20) imply

$$E\left(Z(u_2) - Z(u_1)\right) \approx \begin{cases} 0 & \text{in Case A} \\ -\frac{1}{4}\left((\log_2(R) + eb - 1)\beta - 1\right) c_{\mathrm{ER}} & \text{in Case B} \\ -\frac{1}{2}\left((\log_2(R) + eb - 1)\beta - 1\right) c_{\mathrm{ER}} & \text{in Case C} \end{cases} \tag{22}$$

In the following we focus on the case $\sqrt{0.5} < \beta < 1$, which is the most relevant case since then $0.5R^2 < n < R^2$, i.e. n is a $2\log_2(R)$ bit modulus and, consequently, p_1 and p_2 are $\log_2(R)$-bit numbers. We point out that the case $\beta < \sqrt{0.5}$ can be treated analogously. In (20) and (21) the parameter β_i^{-1} then should be better replaced by $\beta_i^{-1} 2^{\lceil \log_2(p_i)\rceil - \log_2(R)}$. However, the 'correction factor' may not be unambiguous, which might lead to some inaccuracy in the formulae, finally implying a slight loss of attack efficiency.

From (14) we obtain

$$\mathrm{Var}\left(Z(u_2) - Z(u_1)\right) = \sum_{j=1}^{2}\left(\sum_{i=1}^{2}\mathrm{Var}\left(Z_i(u_j)\right) + \mathrm{Var}(N_{e,j})\right) \tag{23}$$

For given $R, eb, c, c_{\mathrm{ER}}, u$ the variance $\mathrm{Var}(Z_i(u))$ is nearly independent of p_i/R and increases somewhat when the ratio u/p_i increases (see Table 1). Since the

true values p_1/R and p_2/R are unknown during the attack we approximate (23) by

$$\mathrm{Var}\left(Z(u_2) - Z(u_1)\right) \approx 4\mathrm{var}_{\beta;max} + 2\sigma_N^2 \tag{24}$$

Here '$\mathrm{var}_{\beta;max}$' suggestively stands for the term (9) with βR in place of p_i and u', i.e. we replace the probabilities p_{i*} and $p_{i(u')}$ by $\beta/3$ and $\beta/2$, respectively. We point out that variance (23) has no direct influence on the decision strategy of our attack but determines the required sample size. Usually, (24) should overestimate (23) somewhat. Moreover, decision errors can be detected and corrected (cf. Section 4, 'confirmed intervals'). So we should be on the safe side anyway. For fixed p_i the mean $E(Z_i(u))$ increases monotonically in u/p_i (see (8)). In fact, our attack exploits these differences.

On basis of execution times for input values (bases) $y = u_i R^{-1} (\mathrm{mod}\, n)$ $(i = 1, 2)$ the attacker has to decide for hundreds of intervals $\{u_1 + 1, \ldots, u_2\}$ whether they contain p_1 or p_2. By (22) the value

$$\mathrm{decbound} := -\frac{1}{8}((\log_2(\mathrm{R}) + \mathrm{eb} - 1)\beta - 1)\, \mathrm{c_{ER}} \tag{25}$$

is a natural decision boundary. In fact, for given $u_1 < u_2$ and $y_i := (u_i R^{-1} (\mathrm{mod}\, n)$ this suggests the following decision rule:

$$\begin{aligned} &\text{Decide for Case A iff } (\mathrm{Time}(y_2^d (\mathrm{mod}\, n)) - \mathrm{Time}(y_1^d (\mathrm{mod}\, n)) > \mathrm{decbound})\,, \\ &\text{and for (Case B or Case C) else.} \end{aligned} \tag{26}$$

(Note that we do not need to distinguish between Case B and Case C.) Here $\mathrm{Time}(y_i^d (\mathrm{mod}\, n))$ denotes the execution time for input value y_i, which of course depends on the blinding factors for the modular exponentiation $(\mathrm{mod}\, p_1)$ and $(\mathrm{mod}\, p_2)$. However, since the variance $\mathrm{Var}(Z(u_2) - Z(u_1))$ is too large for reliable decisions we consider N iid random variables $Z_{[1]}(u), \ldots, Z_{[N]}(u)$ in place of $Z(u)$, which are distributed as $Z(u)$ (corresponding to N exponentiations with input value $y = uR^{-1} (\mathrm{mod}\, n)$). Unlike for decision strategy (26) we evaluate the average timing difference from N pairs of timing measurements (see Sect. 4). For N_τ the inequality

$$\begin{aligned} &\sqrt{\mathrm{Var}\left(\frac{1}{N_\tau}\sum_{j=1}^{N_\tau}\left(Z_{[j]}(u_2) - Z_{[j]}(u_1)\right)\right)} \approx \sqrt{\frac{4\mathrm{var}_{\beta;max} + 2\sigma_N^2}{N_\tau}} \\ &\leq \frac{|\mathrm{decbound} - 0|}{\tau} \quad \text{implies} \\ &N_\tau \geq \frac{\tau^2(4\mathrm{var}_{\beta;max} + 2\sigma_N^2)}{|\mathrm{decbound}|^2} = \frac{64\tau^2(4\mathrm{var}_{\beta;max} + 2\sigma_N^2)}{((\log_2(R) + eb - 1)\beta - 1)^2\, {c_{\mathrm{ER}}}^2}\,. \end{aligned} \tag{27}$$

Applying the above decision strategy (26) to $N \geq N_\tau$ pairs of timing differences the Central Limit Theorem then implies

$$\mathrm{Prob}(\text{wrong decision}) \leq \Phi(-\tau)\,. \tag{28}$$

$\log_2(R)$	eb	c_{ER}	$\beta = \frac{\sqrt{n}}{R}$	$N_{2.5}$	$N_{2.7}$	$\beta = \frac{\sqrt{n}}{R}$	$N_{2.5}$	$N_{2.7}$
512	64	$0.03\,c$	0.75	1458	1701	0.85	1137	1326
512	64	$0.05\,c$	0.75	533	622	0.85	417	486
1024	64	$0.03\,c$	0.75	758	885	0.85	592	690
1024	64	$0.05\,c$	0.75	277	324	0.85	217	253

Table 2. Exemplary sample sizes N_τ for several parameter sets for $\sigma_N^2 = 0$ (no noise). Note that $\Phi(-2.5) = 0.0062$ and $\Phi(-2.7) = 0.0035$. Larger τ reduces the error probability for each decision but increases the sample size N_τ.

Table 2 evaluates (27) for several parameter sets with $\sigma_N^2 = 0$. If $\sigma_N^2 = \alpha \cdot (2\mathrm{var}_{\beta;max})$ the sample size N_τ increases by factor $(1+\alpha)$.

At the end of Phase 1 our attack algorithm from Section 4 has found u_1 and u_2 with $p_i \in \{u_1+1, \ldots, u_2\}$ for $i = 1$ or $i = 2$. Thus in Phase 2 we may replace β by the more precise estimate $\beta_{(2)} := (u_1+u_2)/(2R)$, which may be substituted into the formulae (20) to (27). In particular, we obtain a new decision boundary

$$decbound_{\mathrm{II}} := -\frac{1}{8}((\log_2(R) + eb - 1)\beta_{(2)} - 1)\, c_{\mathrm{ER}}\,, \tag{29}$$

which should be 'better' centered between the mean values $E(Z(u_2) - E(u_1))$ for Case A and for Case B than decbound.

4 The Attack

In this section we describe and analyse the attack algorithm. Two improvements increase its efficiency compared to [12]. We provide experimental results and adjust our attack to table-based exponentiation algorithms. Effective countermeasures are proposed. Amazingly, the attack algorithm and its underlying ideas are rather similar to the attack on unprotected implementations.

4.1 The Attack Algorithm

To simplify notation we introduce the abbreviation

$$\mathrm{MeanTime(u, N)} := \frac{1}{\mathrm{N}} \sum_{\mathrm{j=1}}^{\mathrm{N}} \mathrm{Time}(\mathrm{y_j^d (mod\ n)}) \quad \text{with } \mathrm{y_j := uR^{-1} (mod\ n)} \tag{30}$$

That is, MeanTime(u, N) denotes the average time of N modular exponentiations $y^d (\mathrm{mod}\, n)$ with basis $y = uR^{-1} (\mathrm{mod}\, n)$. The sample size N is selected with regard to the results from Subsection 3.3. In our simulation experiments we used $N_{2.5}$. The attack falls into three phases. The goal of Phase 1 is to find an interval $\{u_1 + 1, \ldots, u_2\}$, which contains p_1 or p_2. In Phase 2 this interval is successively bisected into two halves where that halve is maintained, which is assumed to contain p_i. Phase 2 ends when the attacker knows the upper half

plus few bits of the binary representation of p_i, and in Phase 3 the prime p_i is computed with Coppersmith's algorithm, which transfers the search for p_i into a lattice problem [4]. With regard to Phase 3 one should take care that in Phase 1 and Phase 2 indeed p_1 or p_2 are targeted and not just an integer multiple thereof. If the most relevant case where $p_i > 0.5R$ (definitely fulfilled if $\beta = \sqrt{n}/R > \sqrt{0.5}$) the interval $[\beta R, R]$ contains p_1 or p_2 but no multiple. The following attack may require a pre-step in which the timing parameters c and c_{ER} are estimated (see Remark 1).

The Attack.

Phase 1

Select an integer u somewhat smaller than βR, set (e.g.) $\Delta := 2^{-6}R$
$u_1 := u, u_2 := u_1 + \Delta$
while $(\mathrm{MeanTime}(\mathrm{u}_2, \mathrm{N}) - \mathrm{MeanTime}(\mathrm{u}_1, \mathrm{N}) > \mathrm{decbound})$ do*{
  $u_1 := u_2, u_2 := u_2 + \Delta$
}

Phase 2

while $(\log_2(u_2 - u_1) > 0.5 \log_2(R) - 10)$ do {
  $u_3 := \lfloor (u_1 + u_2)/2 \rfloor$
  if$(\mathrm{MeanTime}(\mathrm{u}_2, \mathrm{N}) - \mathrm{MeanTime}(\mathrm{u}_3, \mathrm{N}) > \mathrm{decbound}_{\mathrm{II}})$ then $u_2 := u_3^*$
else $u_1 := u_3^{**}$}

Phase 3

Apply Coppersmiths algorithm to determine p_i

* The attacker believes that Case A is correct
** The attacker believes that Case B or Case C is correct

After Phase 2 the upper $\approx 0.5 \log_2(p_i) + 10$ bits of u_1 and u_2 coincide, which yields $\approx 0.5 \log_2(p_i) + 10$ bits of p_i. That is, $p_i = \widetilde{p}_i + x_0$ with known $\widetilde{p}_i$ and unknown x_0 and $\log_2(x_0) \approx 0.5 \log_2(R) - 10$. The division $n/\widetilde{p}_i$ yields an analogous decomposition $p_{3-i} = \widetilde{p}_{3-i} + y_0$. Altogether, we obtain a bivariate polynomial equation

$$f(x, y) := (\widetilde{p}_i + x)(\widetilde{p}_{3-i} + y) - n = p_1 p_2 - n = 0\,, \tag{31}$$

for which (x_0, y_0) is a 'small' solution. Reference [4] transfers the problem into a shortest vector problem, which can be solved with the LLL algorithm. This requires that $\log_2(x_0), \log_2(y_0) < 0.25 \log_2(n)$. In Phase 2 we determine ≈ 10 bits more than the upper halve of the bits of p_i to speed up the execution time of the LLL algorithm. We did not solve the lattice problem in our experiments. We counted an attack successful if after Phase 2 p_1 or p_2 was contained in the final interval $\{u_1 + 1, \ldots, u_2\}$.

Of course, if after the end of Phase 2 $\{u_1 + 1, \ldots, u_2\}$ does not contain p_1 or p_2 in Phase 3 the modulus n cannot be factored and thus the attack fails. This means that all decisions until the end of Phase 2 must be correct. For 1024 bit primes, for instance, the algorithm requires about 550 individual decisions. Fortunately, it is very easy to check whether an intermediate interval $\{u_1 + 1, \ldots, u_2\}$ indeed contains a prime (cf. [9], Sect. 5).

Confirmed intervals (i) Assume that after Phase 1 or during Phase 2 the attack algorithm has determined an interval $\{u_1+1,\ldots,u_2\}$. To check whether this interval indeed contains p_1 or p_2 one may perform $2N$ new timing measurements, compute $\mathrm{MeanTime}(\mathrm{u}_2,\mathrm{N})-\mathrm{MeanTime}(\mathrm{u}_1,\mathrm{N})$ and apply the above decision rule. If the time difference is $< decbound_{\mathrm{II}}$ we are convinced that $\{u_1+1,\ldots,u_2\}$ contains p_1 or p_2, and we call $\{u_1+1,\ldots,u_2\}$ a 'confimed interval'. If not, we repeat the test with $2N$ new timing measurements: in case of '$< decbound_{\mathrm{II}}$' we believe that the first test result has been false, and $\{u_1+1,\ldots,u_2\}$ is the new confirmed interval. If again '$> decbound_{\mathrm{II}}$' we believe that an earlier decision was wrong and restart the attack at the preceding confirmed interval. Confirmed intervals should be established after *con* decisions. The value *con* should be selected with regard to the probability for a wrong individual decision. The first confirmed interval should be established at the end of Phase 1.
(ii) Of course, an erroneously confirmed interval will let the attack fail. This probability can be reduced e.g. by applying a 'majority of three' decision rule where the 'original' interval $\{u_1+1,\ldots,u_2\}$ (determined by our attack algorithm) unlike in (i) does not count. Alternatively, the algorithm might jump back to the last but one confirmed interval if the preceding confirmed interval turns out to be wrong with high probability.

Improvements compared to [12] Compared to [12] the attack algorithm features two improvements: First of all, it aims at the larger prime, which increases the difference $E(Z(u_2)) - E_(Z(u_1)$ for Case B and Case C, and in Phase 2 it applies the readjusted decision boundary (29) in place of (25). A comparision between the simulation results in Table 3 with those in Table 3 in [12] shows that these improvements reduce the average number of timing measurements significantly. Additional options to further increase the attack efficiency might be to optimize the selection of *con* in dependence of τ and to apply sequential analysis as in [1].

Remark 5. [Scaling] We assume $eb \ll \log_2(R)$ (typical case).
(i) By (25), (29) and (9) doubling the length of the prime factors p_1 and p_2 roughly doubles decbound, $\mathrm{decbound}_{\mathrm{II}}$ and $\mathrm{var}_{\beta;max}$. If $\sigma_N^2 \approx 0$ by (27) N_τ decreases to approximately 50%. On the other hand, the attack needs about twice as many individual decisions. This points to the surprising fact that the overall number of timing measurements per attack is to a large extent independent of the modulus length if $\sigma_N^2 \approx 0$.
(ii) Similarly, halving c_{ER} halves decbound and $\mathrm{decbound}_{\mathrm{II}}$ but leaves $\mathrm{var}_{\beta;max}$ nearly unchanged. If $\sigma_N^2 \approx 0$ by (27) the attack then requires about 4 times as many timing measurements. The decision boundaries depend linearly on c_{ER} (25). For realistic ratios c_{ER}/c in (9) the $\mathrm{E}(Q_i)(\ldots)$-term, and within the bracket the first summand dominates. Consequently, (27) implies that the number of timing measurements increases roughly like $(c_{\mathrm{ER}}/c)^{-2}$.

Remark 6. As its predecessors in [9, 3, 1] our attack and its variants for table-based exponentiation algorithms (Subsection 4.3) are adaptive chosen input attacks. We point out that our attack would also work for input values $(u + x)R^{-1} (\mathrm{mod}\, n)$ with $|x| \ll n^{1/4}$ in place of the input values $uR^{-1} (\mathrm{mod}\, n)$. This property allows to meet possible minor restrictions on the input values (e.g. some set bits), which might be demanded by the targeted RSA application.

4.2 Experimental Results

In this subsection we present experimental results. As already mentioned in Section 2 it only depends on the quadruple (a, b, M, R) but not on any features of the implementation whether $\mathrm{MM}(a, b; M)$ requires an extra reduction. This property allows to simulate the modular exponentiations $y^d (\mathrm{mod}\, n)$ and to count the number of extra reductions, which finally corresponds to an attack under perfect timing measurements and with $\mathrm{E}(Z_3(u)) = \bar{z}_3$, $\mathrm{Var}(Z_3(u)) = 0$, i.e. $Z_3(u) \equiv z_3$ for all $0 < u < n$, which is an idealization of (12) and (13). Consequently, also in the absence of noise in real-life experiments the number of timing measurements thus should be somewhat larger than for our simulation experiments. The impact of noise was quantified in Subsection 3.3.

In our experiments we selected the primes p_1 and p_2 pseudorandomly. The table entry $p_i/R = 0.75$, for instance, means that p_i has been selected pseudorandomly in the interval $[0.75 - 0.025, 0.75 + 0.025]R$. The secret exponent d was computed according to the public exponent $e = 2^{16} + 1$. Table 3 provides experimental results for several sets of parameters. In our experiments we assumed $\sigma_N^2 = 0$. We calculated N_τ with formula (27) (in Phase 2 with $\mathrm{decbound}_{\mathrm{II}}$), which also allows to extrapolate the number of timing measurements for any noise level. Table 3 confirms the considerations from Remark 5. Several experiments with $p_1/R \approx p_2/R$ were conducted, which verify that the attack becomes the more efficient the larger these ratios are. The reason is that $|\mathrm{decbound}|$ and $|\mathrm{decbound}_{\mathrm{II}}|$ depend almost linearly on β while $\mathrm{var}_{\beta;max}$ remains essentially unchanged. To save computation time many experiments were conducted for 512-bit primes and ratio $c_{\mathrm{ER}}/c \approx 0.05$, which may seem to be relatively large for real-world applications. Remark 5 allows the extrapolation of the simulation results to smaller ratios c_{ER}/c and to other modulus lengths.

The number of timing measurements, which are required for a successful attack, has non-negligible variance. The reason is that if an error has been detected the algorithm steps back to the preceding confirmed interval. We established confirmed intervals after the end of Phase 1, after the end of Phase 2 and regularly after $con = 40$ decisions. For fixed value con a larger τ increases the success rate of the attack but also the number of timing measurements per individual decision.

$\log_2(R)$	eb	c_{ER}	$\frac{p_1}{R}$	$\frac{p_2}{R}$	τ	success rate	$av.\#$exponentiations
512	64	$0.02\,c$	0.75	0.85	2.5	24/25	830,000
512	64	$0.025\,c$	0.75	0.85	2.5	24/25	541,000
512	64	$0.03\,c$	0.75	0.85	2.5	24/25	395,000
512	64	$0.05\,c$	0.75	0.85	2.5	25/25	140,000
512	64	$0.05\,c$	0.70	0.70	2.5	24/25	203,000
512	64	$0.05\,c$	0.80	0.80	2.5	24/25	141,000
512	64	$0.05\,c$	0.85	0.85	2.5	25/25	140,000
512	64	$0.05\,c$	0.90	0.90	2.5	23/25	127,000
768	64	$0.03\,c$	0.75	0.85	2.5	23/25	382,000
768	64	$0.05\,c$	0.75	0.85	2.5	23/25	139,000
1024	64	$0.03\,c$	0.75	0.85	2.5	24/25	410,000
1024	64	$0.05\,c$	0.75	0.85	2.5	24/25	152,000

Table 3. Simulated attack: experimental results. The average numbers of exponentiations (rounded to thousands) refer to the successful attacks. As explained above the primes have been selected pseudorandomly within small intervals around the values in the fourth and fifth column.

4.3 Table-Based Exponentiation Algorithms

The timing attack against unprotected implementations can be adjusted to table-based exponentiation algorithms [9, 3, 1]. This is also possible in case of exponent blinding.

We first consider the fixed-window exponentiation ([7], 14.82), which is combined with Montgomery's exponentiation algorithm. The window size is $b > 1$. In Step i(c) of Algorithm 3 (exponentiation modulo p_i) for basis $y = uR^{-1}(\bmod\ p_i)$ the following precomputations are carried out:

$$\begin{aligned} &y_{0,i} = R(\bmod p_i), y_{1,i} = \mathrm{MM}(y, R^2(\bmod p_i), p_i) = u(\bmod p_i), \quad \text{and} \\ &y_{j,i} := \mathrm{MM}(y_{j-1,i}, y_{1,i}, p_i) \quad \text{for } j = 2, \ldots, 2^b - 1\,. \end{aligned} \tag{32}$$

The exponentiation modulo p_i requires $(2^b - 3) + (\log_2(R) + ebr)/(b2^b)$ Montgomery multiplications by $y_{1,i}$ in average (table initialization + exponentiation phase; the computation of $y_{2,i}$ is actually a squaring operation). The attack tries to exploit these Montgomery multiplications modulo p_1 or p_2, respectively. Compared to the s&m exponentiation algorithm the attack efficiency decreases significantly since the percentage of 'useful' operations (here: the multiplications by y_1) shrinks tremendously. The Montgomery multiplications by $y_{1,i}$ are responsible for the mean timing difference between Case A and (Case B or Case C) . In analogy to (25) for $\sqrt{0.5} < \beta < 1$ we conclude

$$\begin{aligned} decbound_b &= -\frac{1}{2}\left(\frac{E(Q)}{b2^b} + 2^b - 3\right)\frac{\sqrt{n}}{2R}\,c_{\mathrm{ER}} \\ &= -\frac{1}{4}\left(\frac{(\log_2(R) + eb - 1)\beta - 1}{b2^b} + (2^b - 3)\beta\right)c_{\mathrm{ER}}\ . \end{aligned} \tag{33}$$

The computation of $\mathrm{Var}_b(Z_i(u))$ may be organized as in the s&m case. We do not carry out these lengthy calculations in the following but derive an approximation (34), which suffices for our purposes. We yet give some advice how to organize an exact calculation. First of all, the table initialisation modulo p_i costs an additional squaring. In average, there are $E(Q)/b2^b+2^b-3$ multiplications by $y_{i,1}$ (responsible for exploitable timing differences), $E(Q)/b2^b$ multiplications by $y_{i,0}$ (do not need extra reductions) and altogether $(2^b-2)E(Q)/b2^b$ multiplications by some $y_{i,j}$ with $j>1$. When computing the second moment additionally to the s&m case the covarianc $\mathrm{cov}_{i,\mathrm{MM}(\mathrm{u}_\mathrm{i}')}$ (2^b-4 times, table initialization) occur. The term $\mathrm{cov}_{i,\mathrm{MM}(\mathrm{u}_\mathrm{i}')}$ is defined and calculated analogously to $\mathrm{cov}_{i,\mathrm{SM}(\mathrm{u}_\mathrm{i}')}, \mathrm{cov}_{i,\mathrm{MS}(\mathrm{u}_\mathrm{i}')}$ and $\mathrm{cov}_{i,\mathrm{SS}}$.

To assess the efficiency of our timing attack on b-bit fixed window exponentiation we estimate the ratio of the variances $\mathrm{Var}_b(Z_i(u))$ and $\mathrm{Var}(Z_i(u))$ (s&m case). Therefore, we simply count the number of Montgomery operations in both cases (neglecting the different ratios between squarings and multiplications). This gives the rough estimate

$$\frac{\mathrm{Var}_b(Z_i(u))}{\mathrm{Var}(Z_i(u))} \approx \frac{E(Q)+E(Q)/b+2^b}{E(Q)+0.5E(Q)} = \frac{2(b+1)}{3b} + \frac{2^{b+1}}{3E(Q)} =: f_1(b). \tag{34}$$

Finally, we obtain a pendant to (27)

$$N_{\tau,b} \geq \frac{\tau^2(4\mathrm{var}_{\beta;max_b}+2\sigma_N^2)}{|\mathrm{decbound_b}|^2} \approx \frac{\tau^2(4\mathrm{var}_{\beta;max}f_1(b)+2\sigma_N^2)}{|\mathrm{decbound}|^2\mathrm{f}_2^2(\mathrm{b})} \tag{35}$$

with $f_2(b) := |\mathrm{decbound_b}/\mathrm{decbound}|$. In particular, if $\sigma_N^2 \approx 0$ then

$$N_{\tau,b} \approx N_\tau \frac{f_1(b)}{f_2^2(b)}. \tag{36}$$

Remark 7. In analogy to (29) after Phase 1 $\mathrm{decbound_b}$ may be adjusted to $\mathrm{decbound_{b,II}}$. Replacing $\mathrm{decbound_b}$ and decbound by $\mathrm{decbound_{b,II}}$ and $\mathrm{decbound_{II}}$ should not significantly change $f_2(b)$ and $N_{\tau,b}$. The sliding window exponentiation below allows analogous considerations.

For b-bit sliding window exponentiation the table initialization the comprises the following operations:

$$\begin{aligned} &y_{1,i} = \mathrm{MM}(y, R^2 (\mathrm{mod}\, p_i), p_i), y_{2,i} := \mathrm{MM}(y_{1,i}, y_{1,i}, p_i) \quad \text{and} \\ &y_{2j+1,i} := \mathrm{MM}(y_{2j-1,i}, y_{2,i}, p_i) \quad \text{for } j = 1, \ldots, 2^{b-1}-1\,. \end{aligned} \tag{37}$$

In the exponentiation phase the exponent bits are scanned from the left to the right. In the following we derive an estimate for the number of multiplications by the table entries within an exponentiation $(\mathrm{mod}\, p_i)$. Assume that the last window either 'ended' at exponent bit $d_{i,b;j}$ or already at $d_{i,b;j+t'}$, followed by exponent bits $d_{i,b;j+t'-1} = \ldots = d_{i,b;j} = 0$. Let $d_{i,b;j-t}$ denote the next bit that equals 1. We may assume that t is geometrically distributed with parameter 1/2. The next window 'ends' with exponent bit $d_{i,b;j-t}$ iff

$d_{i,b;j-t-1} = \cdots = d_{i,b;j-t-(b-1)} = 0$. In this case table entry $y_{1,i}$ is applied, and this multiplication is followed by $(b-1)$ squarings that correspond to the exponent bits $d_{i,b;j-t-1}, \ldots, d_{i,b;j-t-(b-1)}$. Alternatively, the next window might end with exponent bit $d_{i,b;j-t-2}$ (resp. with exponent bit $d_{i,b;j-t-3}$,...,$d_{i,b;j-t-(b-1)}$) iff $d_{i,b;j-t-2} = 1$, $d_{i,b;j-t-3} = \cdots = d_{i,b;j-t-(b-1)} = 0$ (resp. iff $d_{i,b;j-t-3} = 1$, $d_{i,b;j-t-4} = \cdots = d_{i,b;j-t-(b-1)} = 0$,..., iff $d_{i,b;j-t-(b-1)} = 1$). Of course, if the window ends before exponent bit $d_{i,b;j-t-(b-1)}$ it is followed by some squarings. Altogether, the exponent bits $d_{i,b;j-1}, \ldots, d_{i,b;j-t-(b-1)}$ need one multiplication by some table entry. Neglecting boundary effects one concludes that sliding window exponentiation requires one multiplication by a table entry per

$$\sum_{s=1}^{\infty} s2^{-s} + (b-1) = 2 + b - 1 = b + 1 \tag{38}$$

exponent bits in average. This gives the pendant to (34):

$$\begin{aligned} \frac{\mathrm{Var}_{b,sw}(Z_i(u))}{\mathrm{Var}(Z_i(u))} &\approx \frac{\left(1 + \frac{1}{b+1}\right) E(Q) + 2^{b-1} + 1}{E(Q) + 0.5E(Q)} \\ &= \frac{2(b+2)}{3(b+1)} + \frac{2^b + 2}{3E(Q)} =: f_{1,sw}(b). \end{aligned} \tag{39}$$

where the subscript 'sw' stands for 'sliding window'. Since there is a bijection between the table entries and the $(b-1)$ exponent bits $(d_{i,b;j-t-1} \ldots, d_{i,b;j-t-b+1})$ all table entries are equally likely. For many parameter sets $(\log_2(R) + eb, b)$ the table entry $y_{1,i}$ occurs less often than $y_{2,i}$, which is carried out $2^{b-1} - 1$ times within the table initialization. (Numerical example: For $(\log_2(R)+eb, b) = (1024+64, 5)$ in average $E(Q)/(16(5+1)) \approx 11.3 < 15$ multiplications with table entry $y_{1,i}$ occur.) Then, as in [1] our attack then focuses on the Montgomery multiplications by $y_{2,i}$. In particular, we then obtain the decision boundary

$$decbound_{b,sw} = -\frac{1}{2}\left(2^{b-1} - 1\right) \frac{\sqrt{n}}{2R}\, c_{\mathrm{ER}} = -\frac{1}{4}\left(2^{b-1} - 1\right) \beta\, c_{\mathrm{ER}} \tag{40}$$

(Of course, if $2^{-(b-1)}E(Q_i)/(b+1) > 2^{b-1} - 1$ then in (40) the term $(2^{b-1} - 1)$ should be replaced by $2^{-(b-1)}E(Q_i)/(b+1)$, and the attack should focus on the multiplications by table value $y_{i,1}$.) Setting

$$f_{2,sw}(b) := |\mathrm{decbound_{b,sw}}|/\mathrm{decbound}| \tag{41}$$

we obtain an analogous formula to (36):

$$N_{\tau,b,sw} \approx N_\tau \frac{f_{1,sw}(b)}{f_{2,sw}^2(b)}\,. \tag{42}$$

Example 1. $\log_2(R) = 1024$ (i.e., 2048-bit RSA), $eb = 64$, $\beta = 0.8$, and $\sigma_N^2 \approx 0$. (i) [$b = 6$] For fixed window exponentiation $N_{\tau,b} \approx 59N_\tau$, i.e. the overall attack costs ≈ 59 times the number of timing measurements for s&m. For sliding

window exponentiation we obtain $N_{\tau,b,sw} \approx 240N_\tau$. We applied formula (21) to estimate $E(Q_i)$.
(ii) $[b = 5]$ $N_{\tau,b} \approx 189N_\tau$, $N_{\tau,b,sw} \approx 1032N_\tau$.
(iii) $[b = 4]$ $N_{\tau,b} \approx 277N_\tau$, $N_{\tau,b,sw} \approx 322N_\tau$.
(iv) $[b = 3]$ $N_{\tau,b} \approx 104N_\tau$, $N_{\tau,b,sw} \approx 54N_\tau$.
(v) $[b = 2]$ $N_{\tau,b} \approx 16N_\tau$, $N_{\tau,b,sw} \approx 8N_\tau$.
Note: For $b = 2, 3, 4$ the timing attack on sliding window exponentiation aims at the multiplications by $y_{i,1}$, for $b = 5, 6$ on the multiplications by $y_{2,i}$ during the table initialization. For $b = 4, 5, 6$ the attack on fixed window exponentiation is more efficient than the attack on sliding window exponentiation while for $b = 2, 3$ the converse is true.

It is hardly possible to define a clear-cut lower bound for the number of timing measurements from which on the attack should be viewed impractical. The maximum number of available timing measurements clearly depends on the concrete attack scenario. Cryptographic software on PCs and servers usually applies a large table size b, and the timing measurements are often to some degree noisy. Example 1 shows that for large window size b and realistic ratios c_{ER}/c the attack requires a gigantic number of timing measurements, all the more in the presence of non-negligible noise. Example 1 provides these numbers relative to the square & multiply case. The absolute numbers of timing measurements depend on the ratios c_{ER}/c and p_i/R and on the level of noise (cf. Remark 5(ii), Subsection 4.2 and Subsection 3.3).

4.4 Countermeasures

The most solid countermeasure is to avoid extra reductions entirely. In fact, one may resign on the extra reductions within modular exponentiation if $R > 4p_i$ ([13], Theorem 3 and Theorem 6). This solution (resigning on extra reductions) was selected for OpenSSL as response on the instruction cache attack described in [2]. We point out that the present attack could also be prevented by combining exponent blinding with base blinding ([6], Sect. 10), for example, which in particular would also prevent the attack from [2]. However, the first option is clearly preferable as it prevents any type of timing attack.

5 Conclusion

It has widely been assumed that exclusive exponent blinding would prevent timing attacks. This paper shows that this assumption is not generally true (although exponent blinding reduces the efficiency of our timing attack significantly). In the presence of little or moderate noise our attack is a practical threat against square & multiply exponentiation and should be considered (see also Remark 6). Our attack can also be applied to fixed window exponentiation and to sliding window exponentiation. However, for large window size b the attack requires a very large number of timing measurements. The attack may be practically infeasible then, in particular for small ratios c_{ER}/c or in the presence of non-negligible noise. Fortunately, effective countermeasures exist.

References

1. O. Acıiçmez, W. Schindler, Ç.K. Koç, Improving Brumley and Boneh Timing Attack on Unprotected SSL Implementations. In: C. Meadows, P. Syverson (eds.): 12^{th} ACM Conference on Computer and Communications Security — CCS 2005, ACM Press, New York 2005, 139–146.
2. O. Acıiçmez, W. Schindler: A Vulnerability in RSA Implementations due to Instruction Cache Analysis and Its Demonstration on OpenSSL. In: T. Malkin (Hrsg.): Topics in Cryptology — CT-RSA 2008, Springer, Lecture Notes in Computer Science 4964, Berlin 2008, 256–273.
3. D. Brumley, D. Boneh: Remote Timing Attacks are Practical. In: Proceedings of the 12th Usenix Security Symposium, 2003.
4. D. Coppersmith: Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. J. Cryptology 10 (1997), 233–260.
5. J.-F. Dhem, F. Koeune, P.-A. Leroux, P.-A. Mestré, J.-J. Quisquater, J.-L. Willems: A Practical Implementation of the Timing Attack. In: J.-J. Quisquater and B. Schneier (eds.): Smart Card – Research and Applications, Springer, Lecture Notes in Computer Science 1820, Berlin 2000, 175–191.
6. P. Kocher: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In: N. Koblitz (ed.): Crypto 1996, Springer, Lecture Notes in Computer Science 1109, Heidelberg 1996, 104–113.
7. A.J. Menezes, P.C. van Oorschot, S.C. Vanstone: Handbook of Applied Cryptography, Boca Raton, CRC Press 1997.
8. P.L. Montgomery: Modular Multiplication without Trial Division. Math. Comp. 44 (1985), 519–521.
9. W. Schindler: A Timing Attack against RSA with the Chinese Remainder Theorem. In: Ç.K. Koç, C. Paar (eds.): Cryptographic Hardware and Embedded Systems — CHES 2000, Springer, Lecture Notes in Computer Science 1965, Berlin 2000, 110–125.
10. W. Schindler, F. Koeune, J.-J. Quisquater: Improving Divide and Conquer Attacks Against Cryptosystems by Better Error Detection / Correction Strategies. In: B. Honary (ed.): Cryptography and Coding — IMA 2001, Springer, Lecture Notes in Computer Science 2260, Berlin 2001, 245–267.
11. W. Schindler: Optimized Timing Attacks against Public Key Cryptosystems. Statist. Decisions 20 (2002), 191–210.
12. W. Schindler: Exponent Blinding May Not Prevent Timing Attacks on RSA. Cryptology ePrint Archive, Report 2014/869, https://eprint.iacr.org/2014/869, Version 20141022:205703
13. C. D. Walter: Precise Bounds for Montgomery Modular Multiplication and Some Potentially Insecure RSA Moduli. In. B. Preneel (ed.): Topics in Cryptology — CT-RSA 2002, Springer, Lecture Notes in Computer Science 2271, Berlin 2002, 30–39.