A Timing Attack against RSA with the Chinese Remainder Theorem

Werner Schindler

Bundesamt für Sicherheit in der Informationstechnik (BSI)
Godesberger Allee 183, 53175 Bonn, Germany
Werner.Schindler@bsi.bund.de

Abstract. We introduce a new type of timing attack which enables the factorization of an RSA-modulus if the exponentiation with the secret exponent uses the Chinese Remainder Theorem and Montgomery's algorithm. Its standard variant assumes that both exponentiations are carried out with a simple square and multiply algorithm. However, although its efficiency decreases, our attack can also be adapted to more advanced exponentiation algorithms. The previously known timing attacks do not work if the Chinese Remainder Theorem is used.
Keywords: Timing attack, RSA, Chinese Remainder Theorem, Montgomery multiplication.

1 Introduction

The central idea of any timing attack is to determine a secret parameter from differences between running times needed for various input values. At Crypto 96 Kocher introduced a timing attack on modular exponentiations ([6]). In [3] a successful attack against an early version of a Cascade smart card is described. In [9] these attacks are optimized and, moreover, the assumptions on the attacker's abilities are weakened considerably.

The attacks mentioned above recover an unknown exponent (e.g. a secret RSA key) bit by bit. They yet do not work if the Chinese Remainder Theorem (CRT) is used as it is essential to know the exact input values of the respective arithmetical operations at any instant. (Concerning the CRT there is no more than a rough idea sketched in [6] (Sect. 7) how to exploit time differences caused by the initial reduction $y \mapsto y (\mathrm{mod}\, p_j)$. However, this should be of little practical significance since the variance of the remaining hundreds of arithmetic operations usually is gigantic compared with this effect.)

In this paper we introduce and investigate a completely new type of timing attack. It enables the factorization of an RSA modulus n if the exponentiation with the secret exponent uses the CRT while the multiplications and squarings modulo the prime factors p_1 and p_2 are carried out with Montgomery's algorithm ([8]). The standard variant of our attack assumes that both exponentiations use a simple square and multiply algorithm. Although its efficiency decreases, our attack can also be adapted to more advanced exponentiation algorithms. Our

Ç.K. Koç and C. Paar (Eds.): CHES 2000, LNCS 1965, pp. 109–124, 2000.

attack is very robust and, for comparison, needs notably fewer time measurements than the attacks introduced in [6] and [3]. If exponentiation uses square and multiply under optimal circumstances the standard variant of our attack requires fewer than 600 time measurements to factor a 1024 bit modulus n. Making use of a lattice-based algorithm introduced by Coppersmith in [2] again nearly halves this number of time measurements. Further, one can verify during the attack with high probability whether the decisions have been correct so far. The only weakness of our attack is that it is a chosen-input attack.

First, we briefly describe and investigate Montgomery's algorithm and exponentiation with CRT. In Sect. 3 general assumptions are formulated and discussed and the central idea of our attack is illustrated. In Sect. 4 and 5 the standard variant of our attack is worked out, error probabilities are computed, and mechanisms for error detection and error correction are discussed. Sect. 6 presents results of practical experiments and in Sect. 7 we extend our attack to implementations which use more advanced exponentiation schemes than square and multiply. The paper ends with remarks on fields of application, possible countermeasures and concluding remarks.

2 Montgomery's Algorithm and the CRT

Let $(d_w d_{w-1} \dots d_0)_2$ denote the binary representation of $d \in \mathbb{N}$ where $d_w = 1$ denotes the most significant bit. If d denotes a secret key the computation of $y^d (\bmod m)$ usually requires hundreds of modular squarings and multiplications. Hence many implementations use Montgomery's algorithm ([8]) which transfers time-consuming modular multiplications modulo m to a modulus $R > m$ with $gcd(R, m) = 1$ which fits to the device's hardware architecture. (Usually $R = 2^\omega$ where ω is a multiple of 32 or 64.)

Let's have a closer look at Montgomery's algorithm. As usually, for $a \in Z$ the term $a (\bmod m)$ denotes the smallest nonnegative integer which is congruent to a modulo n while $R^{-1} \in Z_m := \{0, 1, \dots, m-1\}$ denotes the multiplicative inverse of R in Z_m. The integer $m^* \in Z_R$ satisfies the equation $RR^{-1} - mm^* = 1$ in Z. To simplify notation we introduce the mappings $\Psi, \Psi_* \colon Z \to Z_m$ defined by $\Psi(x) := (xR) (\bmod m)$ and $\Psi_*(x) := (xR^{-1}) (\bmod m)$. As easily can be checked Ψ and Ψ_* are bijective on Z_m and, moreover, inverse mappings; i.e. $\Psi_*(\Psi(x)) = x$ for all $x \in Z_m$. For $a' := \Psi(a)$ and $b' := \Psi(b)$ Montgomery's algorithm returns $s := \Psi_*(\Psi(a)\Psi(b)) = \Psi(ab)$. The subtraction $s - m$ in line 4 is called *extra reduction*.

Montgomery's algorithm

```
z:=a'b'
r:=(z(mod R)m*) (mod R)
s:=(z+rm)/R
if s≥m then s:=s-m
return s        (= Ψ*(a'b') = a'b'R^-1 (mod m))
```

Remark 1. Many implementations use a more efficient multiprecision variant of Montgomery's algorithm than listed above (see e.g. [7], Algorithm 14.36, or

[10]). The factors a' and b' then are internally represented with respect to a basis h which fits perfectly to the hardware multipliers (typically, $h = 2^{32}$), that is, $a' := \sum_{j=0}^{t-1} a'_j h^j$ and $b' := \sum_{j=0}^{t-1} b'_j h^j$. Instead of the product $a'b'$ and the reduction modulo $R := h^t$ it suffices to calculate $a'_0 b', a'_1 b', \ldots, a'_{t-1} b'$ and to perform t reductions modulo h. However, whether an extra reduction is necessary does not depend on the chosen basis h but on $R := h^t$. (This can be verified with a simple induction proof.) Thus, as will become clear later, the concrete realization of Montgomery's algorithm is indeed of no significance for our attack.

Combined with Montgomery's algorithm the square and multiply exponentiation algorithm reads as follows:

Exponentiation algorithm 1
(Square and multiply using Montgomery's algorithm)

```
temp := Ψ(y)
for i=w-1 down to 0 do {
  temp := Ψ*(temp²)
  if (dᵢ=1) then temp := Ψ*(temp*Ψ(y))
  }
return Ψ*(temp)
```

Suggestively, we call operations $\Psi_*(\text{temp}^2)$ and $\Psi_*(\text{temp} * \Psi(y))$ *Montgomery multiplications* in the following. The number of extra reductions in Exponentiation algorithm 1 depends on the particular base y, or more directly, on $\Psi(y)$.

Lemma 1. *(i) Montgomery's algorithm requires an extra reduction step iff*

$$\frac{a'b'}{Rm} + \frac{a'b'm^* (\mathrm{mod}\ R)}{R} \geq 1. \tag{1}$$

(ii) Let the random variable B be equidistributed on Z_m. Unless the ratio $R/gcd(R, \Psi(a))$ is extremely small

$$\text{Prob}\,(\textit{extra reduction in } \Psi_*(\Psi(a)B)) = \frac{\Psi(a)}{2R} \qquad \textit{for } a \in Z_m \tag{2}$$

holds and similarly

$$\text{Prob}\left(\textit{extra reduction in } \Psi_*(B^2)\right) = \frac{m}{3R}\ . \tag{3}$$

Proof. For the proof of (1) to (3) we refer the interested reader to [9]. We merely sketch the central idea to verify (2). Its left-hand side is equivalent to $\text{Prob}(Ac + (Acmm^* (\mathrm{mod}\ 1)) \geq 1)$ where we temporarily use the abbreviations $A := B/m$ and $c := \Psi(a)/R$. Further, for fixed $v \in \mathbb{N}$ (e.g., $v = 32$) define the intervals $I_j := [j2^{-v}, (j+1)2^{-v})$ for $j < 2^v$. For realistic modulus size m under mild conditions $\text{Prob}(Acmm^* (\mathrm{mod}\ 1) \mid A \in I_i) \approx 2^{-v}$ should be an excellent approximation for $i, j < 2^v$. Then the left-hand side of (2) approximately equals $\text{Prob}(Uc + V \geq 1)$ where U and V denote independent random variables being equidistributed on $[0, 1)$. The latter probability equals the right-hand side of (2).

Remark 2. We point out that the proof of (2) and (3) is not exact in a mathematical sense as it uses (though plausible!) heuristic arguments at two steps. However, results from practical experiments (thousands of pseudorandom factors and various moduli) match perfectly with (2) and (3). For none of these factors and moduli neither (2) nor (3) turned out to be wrong. Hence it seems to be resonable to use "=" instead of "≈".

Let $n = p_1 p_2$ denote an RSA-modulus with primes p_1 and p_2 while d denotes the secret exponent. Steps 1 to 3 below describe how to compute $y^d (\mathrm{mod}\, n)$ using the CRT and Montgomery's algorithm. The numbers d', d'', b_1 and b_2 and the parameters for the Montgomery multiplications $(\mathrm{mod}\, p_1)$ and $(\mathrm{mod}\, p_2)$ are precomputed in a setup Step carried out only once after loading (or generating within the device, resp.) p_1, p_2 and d. In particular, $d' := d(\mathrm{mod}\,(p_1 - 1))$ and $d'' := d(\mathrm{mod}\,(p_2 - 1))$ while $b_1 \equiv 1 \pmod{p_1}$ and $b_1 \equiv 0 \pmod{p_2}$, and similarly, $b_2 \equiv 0 \pmod{p_1}$ and $b_2 \equiv 1 \pmod{p_2}$.

CRT using Exponentiation algorithm 1

Step 1: a) $y_1 := y(\mathrm{mod}\, p_1)$
b) Compute $x_1 := (y_1)^{d'} (\mathrm{mod}\, p_1)$ with Exponentiation algorithm 1
Step 2: a) $y_2 := y(\mathrm{mod}\, p_2)$
b) Compute $x_2 := (y_2)^{d''} (\mathrm{mod}\, p_2)$ with Exponentiation algorithm 1
Step 3: Return $(b_1 x_1 + b_2 x_2)(\mathrm{mod}\, n)$

3 General Assumptions and the Central Idea

We assume that the attacker has access to a hardware device (smart card, PC etc.) which calculates the modular exponentiation with a secret RSA exponent using CRT with Montgomery multiplication. Below, we will formulate and discuss assumptions concerning the implementation. Then we derive the main theorem and explain the central idea of our attack.

Definition 1. *In analogy to Sect. 2 for $i = 1, 2$ we define the mappings $\Psi_i, \Psi_{i*}\colon Z \to Z_{p_i}$ by $\Psi_i(a) := (aR)(\mathrm{mod}\, p_i)$ and $\Psi_{i*}(a) := (aR^{-1})(\mathrm{mod}\, p_i)$. As usually, the greatest common divisor of integers a and b is denoted with $\gcd(a, b)$. The term $N(\mu, \sigma^2)$ denotes a normal distribution with mean (=expected value) μ and variance σ^2. A value taken on by a random variable X is called a* realization *of X.*

General Assumptions. a) The attacker is able to use the hardware device for chosen inputs and to measure the total time needed for an exponentiation.
b) A modular exponentiation $y^d (\mathrm{mod}\, n)$ is computed with the CRT using Exponentiation algorithm 1 stated at the end of Sect. 2.
c) The attacker knows the modulus n.
d) Both, Montgomery multiplications $(\mathrm{mod}\, p_1)$ and $(\mathrm{mod}\, p_2)$ use the same parameter value R.
e) Montgomery multiplications $(\mathrm{mod}\, p_1)$ and $(\mathrm{mod}\, p_2)$ require time c if no extra reduction is needed and $c + c_{\mathrm{ER}}$ otherwise.

f) Running times are reproducible, i.e. for fixed d and n the running time Time($y^d \pmod n$) does only depend on the base y but not on other (e.g. external) influences.
g) For a randomly chosen base y the cumulative time needed for all operations besides the Montgomery multiplications inside the loop of Exponentiation Algorithm 1 in Steps 1 and 2 of the CRT algorithm (in particular, the time needed for input and output operations, for the calculation of $y_i, \Psi_i(y_i), \Psi_{i*}(\text{temp})$ $(i = 1, 2)$ and for $b_1x_1 + b_2x_2 (\bmod n)$) may be viewed as realization of a $N(\mu_{\text{CRT}}, {\sigma_{\text{CRT}}}^2)$-distributed random variable.

Our attack is a chosen input attack which enables the factorization of the modulus n. It will turn out that it tolerates measurement errors and external influences and also works under less restrictive assumptions. In Sect. 7 it will be extended to CRT combined with more advanced exponentiation algorithms than Exponentiation algorithm 1. We will reference to the general assumptions using the abbreviation GA.

Remark 3. (i) Re GA d): The assumption that both, exponentiation $(\bmod p_1)$ and $(\bmod p_2)$ use the same parameter value R is usually fulfilled as both prime factors have the same number of bits.
(ii)Re GA e): Reductions modulo a power of 2 and divisions by a power of 2 can simply be realized by neglecting the high-value bits or as a shift operation, resp. Due to GA d) assumption GA e) usually should be fulfilled (see also [3]). However, slight deviations from constant running times could be interpreted as a part of the measurement error (see Remark 7).
(iii) Re GA f): Concerning external influences assumption GA f) should be fulfilled for smart cards (but not for multi-user-systems). Instead, there might be randomly chosen dummy operations masking the "true" running time (see Remark 7).

To simplify further notation we will use the following abbreviations and definitions

$$R^{-1} := R^{-1} (\bmod n), \qquad \beta := \sqrt{n/R^2} \tag{4}$$

$$T(u) := \text{Time}((uR^{-1} (\bmod n))^d (\bmod n)) \ . \tag{5}$$

The term $T(u)$ covers all the time needed to compute $(uR^{-1}(\bmod n))^d (\bmod n)$. The CRT delivers $(uR^{-1} (\bmod n))R \equiv u \pmod{p_i}$ so that Theorem 1 is an immediate corollary of Lemma 1. We recall that the right-hand sides of (6) and (7) are at least excellent approximations. Theorem 1 is crucial for our attack.

Theorem 1. *Let B_i denote a random variable being uniformly distributed on Z_{p_i}. Then for $i = 1, 2$*

$$\text{pr}_{i*} := \mathit{Prob}\left(\mathit{extra\ reduction\ in}\ \Psi_{i*}(B_i^2)\right) = \frac{p_i}{3R} \tag{6}$$

and, unless the ratio $R/gcd(R, u (\bmod p_i))$ is extremely small, also

$$\begin{aligned} \text{pr}_i(u) :=& \text{Prob}\left(\mathit{extra\ reduction\ in}\ \Psi_{i*}(\Psi_i(uR^{-1} (\bmod n))B_i) = \Psi_{i*}(uB_i)\right) \\ &= \frac{u (\bmod p_i)}{2R} \qquad \mathit{for}\ u \in Z_n \ . \end{aligned} \tag{7}$$

Figures 1 and 2 below illustrate the central idea of our attack. For base $uR^{-1} (\mathrm{mod}\, n)$ hundreds of Montgomery multiplications have to carried out with factors $u (\mathrm{mod}\, p_1)$ and $u (\mathrm{mod}\, p_2)$, respectively. ¿From (7) we conclude that the probability for an extra reduction within any of these multiplication is linear in the respective factor. Differences between running times required for modular exponentiations result from different numbers of extra reductions within the respective modular multiplications and squarings. Figure 2 plots the expected number of extra reductions (=E(#er)) as a function of u in a neighborhood of kp_i where k is an integer and $i \in \{1, 2\}$. Hence for $u_1 < u_2$ with $u_2 - u_1 \ll p_i$ the time difference $T(u_2) - T(u_1)$ should reveal whether the interval $\{u_1 + 1, \ldots, u_2\}$ contains an integer multiple of at least one prime factor p_i or not. In the first case $T(u_1)$ should be significantly larger than $T(u_2)$ while in the second case both running times should approximately be equal. In the following sections we will make this intuitive idea precise.

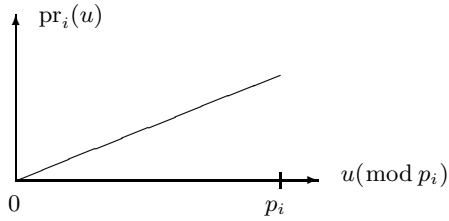

Fig. 1. Probability for an extra reduction with a random cofactor

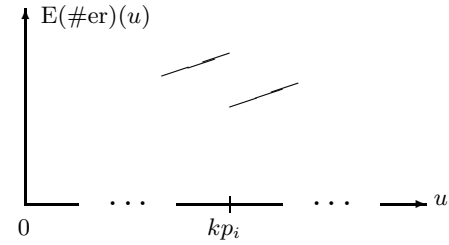

Fig. 2. The expected number of extra reductions is discontinuous at each integer multiple of p_i.

Our attacks falls in three phases: In Phase 1 an "interval set" $\{u_1 + 1, \ldots, u_2\}$ has to found which contains an integer multiple of p_1 or p_2. Starting from this set in Phase 2 a sequence of decreasing interval subsets has to be determined, each of which containing an integer multiple of p_1 or p_2. The decisions in Phase 1 and 2 are based on the respective time differences $T(u_2) - T(u_1)$. As soon as the actual subset is small enough Phase 3 begins where $\gcd(u, n)$ is calculated for all u contained in this subset. If all decisions within Phase 1 and 2 were correct then the final subset indeed contains a multiple of p_1 or p_2 so that Phase 3 delivers the factorization of n.

4 Basic Scheme

Let the exponent d' be a $(w_1 + 1)$ bit number with $(g_1 + 1)$ ones in its binary representation and similarly, the exponent d'' be a $(w_2 + 1)$ bit number with $(g_2 + 1)$ ones in its binary representation. Then within the for-loops of Exponentiation

algorithm 1 $w_1+w_2+g_1+g_2$ Montgomery multiplications are carried out, in particular, w_1 of type $\Psi_{1*}(\text{temp}^2)$ and g_1 of type $\Psi_{1*}(\text{temp} * \Psi_1(y_1))$ and, similarly, w_2 of type $\Psi_{2*}(\text{temp}^2)$ and g_2 of type $\Psi_{2*}(\text{temp} * \Psi_2(y_2))$. For $y = uR^{-1} (\bmod n)$ we will view the cumulative time needed for these Montgomery multiplications as a realization of a $N(\mu_{\text{MM}}(u), \sigma_{\text{MM}}(u)^2)$-distributed random variable. The temp-values are the intermediate results $\Psi_i(y_i), \Psi_i(y_i^2), \ldots, \Psi_i(y_i^d)$ which behave as realizations of random variables equidistributed on Z_{p_i}. Then (9) is an immediate consequence of Theorem 1. We point out that extra reductions in subsequent Montgomery multiplications are negative correlated and that under mild assumptions a pendant of the well-known central limit theorem also holds for dependent random variables. (The first assertion follows from the fact that after an extra reduction $\text{temp} < (p_i/R)p_i$ or $\text{temp} < (u(\bmod p_i)/R)p_i$, resp.) Consequently, due to GA g) the running time $T(u)$ then may be viewed as a realization of a $N(\mu(u), \sigma(u)^2)$-distributed random variable X_u with

$$\mu(u) = \mu_{\text{CRT}} + \mu_{\text{MM}}(u), \qquad \sigma(u)^2 = {\sigma_{\text{CRT}}}^2 + \sigma_{\text{MM}}(u)^2 \tag{8}$$

$$\mu_{\text{MM}}(u) \approx c \sum_{i=1}^{2} (w_i + g_i) + \text{c}_{\text{ER}} \sum_{i=1}^{2} (w_i \text{pr}_{i*} + g_i \text{pr}_i(u)) \tag{9}$$

Theorem 2.

$$\sigma_{\text{MM}}(u)^2 \approx {\text{c}_{\text{ER}}}^2 \sum_{i=1}^{2} [w_i \text{pr}_{i*}(1 - \text{pr}_{i*}) + g_i \text{pr}_i(u)(1 - \text{pr}_i(u)) \tag{10}$$

$$+2(g_i - 1)\text{cov}_{i;\text{MQ}}(u) + 2g_i \text{cov}_{i;\text{QM}}(u) + 2(w_i - g_i)\text{cov}_{i;\text{QQ}}] \quad \textit{with}$$

$$\text{cov}_{i;\text{MQ}}(u) = 2\text{pr}_i(u)^3 \text{pr}_{i*} - \text{pr}_i(u)\text{pr}_*, \quad \text{cov}_{i;\text{QM}}(u) = \frac{9}{5}\text{pr}_i(u)\text{pr}_{i*}^2 - \text{pr}_i(u)\text{pr}_*,$$

$$\text{cov}_{i;\text{QQ}} = \frac{27}{7}\text{pr}_{i*}^4 - \text{pr}_{i*}^2 \ .$$

Proof. Using the same notation as in the proof of Lemma 1 the left-hand side of (2) equals $\text{Prob}(Ac + Acmm^*(\bmod 1) \geq 1)$. As pointed out there slight deviations in the first summand should cause "vast" variations in the second, i.e. with respect to this particular probability both summands should behave as if they were independent and if the second was equidistributed on $[0, 1)$. Using (1) a similar assertion can be derived for (3). Under these assumptions we may view the temp values in Exponentiation algorithm 1 in Step $i \in \{1, 2\}$ as realizations of the random variables $S_0, S_1, \ldots$ which are recursively defined by $S_0 := \Psi_i(y_i)$ and $S_k := (S_{k-1}\Psi_i(y_i)/R + V_k) \,(\bmod 1)$ or $S_k := (S_{k-1}^2 p_i/R + V_k) \,(\bmod 1)$, resp., if the k^{th} Montgomery multiplication is a multiplication with $\Psi_i(y_i)$ or a squaring, resp. Here $V_1, V_2, \ldots$ denote independent random variables being equidistributed on $[0, 1)$. Further, define the $\{0, 1\}$-valued random variables $W_1, W_2, \ldots$ by $W_k := 1_{\{S_k < S_{k-1}\Psi_i(y_i)/R\}}$ or $W_k := 1_{\{S_k < S_{k-1}^2 p_i/R\}}$, resp. Then $W_k = 1$ iff an extra reduction is necessary in Montgomery multiplication k. As $V_1, V_2, \ldots$ are independent and equidistributed on $[0, 1)$ the same is true for $S_1, S_2 \ldots$. If ν

denotes the distribution of S_{g-1} (Dirac measure or equidistribution, resp.) then for $1 \le g < h$ we obtain the covariance $\mathrm{Cov}_{\mathrm{MQ}}(W_g W_h) =$

$$\int_{[0,1)^{h-g+2}} 1_{\{S_g < S_{g-1}\Psi_i(y_i)/R\}}(s_g) \cdot 1_{\{S_h < S_{h-1}^2 p_i/R\}}(s_h)\, ds_h \cdots ds_g \nu(ds_{g-1})$$
$$- \left(\int_{[0,1)^2} 1_{\{S_g < S_{g-1}\Psi_i(y_i)/R\}}(s_g)\, ds_g \nu(ds_{g-1}) \right) \cdot \left(\int_{[0,1)^2} 1_{\{S_h < S_{h-1}^2 p_i/R\}}(s_h)\, ds_h\, ds_{h-1} \right).$$

Here subscripts MQ means that index g belongs to a multiplication with $\Psi_i(y_i)$ and h to a squaring. The covariance $\mathrm{Cov}_{\mathrm{MQ}}(W_g W_h) = 0$ if $h > g+1$ and equals $\mathrm{cov}_{i;\mathrm{MQ}}(u)$ if $h = g+1 > 2$. Equivalent assertions hold for $\mathrm{Cov}_{\mathrm{QM}}(W_g W_h)$, $\mathrm{Cov}_{\mathrm{MM}}(W_g W_h)$ and $\mathrm{Cov}_{\mathrm{QQ}}(W_g W_h)$. Approximating the covariance of W_1 and W_2 by $\mathrm{cov}_{i;\mathrm{QM}}(u)$ or $\mathrm{cov}_{i;\mathrm{QQ}}$, resp., finishes the proof of Theorem 2 as the least-value bits of d' and d'' are 1.

Remark 4. (i) The proof of Theorem 2 is not exact in a mathematical sense as it uses the same heuristic arguments as Lemma 1. However, (10) matches with practical experiments. We point out that the variance $\sigma_{\mathrm{MM}}(u)^2$ does not affect the (single) decision rule defined below but its knowledge enables the choice of appropriate parameters s and N (see Sect. 5 and 7).

Now let $0 < u_1 < u_2 < n$ with $u_2 - u_1 < p_1, p_2$. 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: $\{u_1+1, \ldots, u_2\}$ contains a multiple of one of p_1 or p_2 but not of both.
Case C: $\{u_1+1, \ldots, u_2\}$ contains a multiple of both p_1 or p_2.

Clearly, the expected value of the time difference $T(u_2) - T(u_1)$ equals $E(X_{u_2} - X_{u_1}) = \mu(u_2) - \mu(u_1)$. In Case B p_j denotes the prime factor of which $\{u_1 + 1, \ldots, u_2\}$ contains a multiple. From (9) and Theorem 1 we obtain

$$E(X_{u_2} - X_{u_1}) = \begin{cases} \frac{c_{\mathrm{ER}}}{2R}\left(g_1(u_2-u_1) + g_2(u_2-u_1)\right) & \text{in Case A} \\ \frac{c_{\mathrm{ER}}}{2R}\left(g_j(u_2-u_1-p_j) + g_{3-j}(u_2-u_1)\right) & \text{in Case B} \\ \frac{c_{\mathrm{ER}}}{2R}\left(g_1(u_2-u_1-p_1) + g_2(u_2-u_1-p_2)\right) & \text{in Case C} \end{cases} \quad (11)$$

unless the ratios $R/\gcd(R, u_1(\bmod p_i))$ and $R/\gcd(R, u_2(\bmod p_i))$ are extremely small. (For randomly chosen u_1, u_2 this should not never occur in practice.) If $u_2 - u_1 \ll p_1, p_2$ the expected values differ considerably:

$$E(X_{u_2} - X_{u_1}) \approx \begin{cases} 0 & \text{in Case A} \\ -\frac{c_{\mathrm{ER}}}{2R}\left(g_j p_j\right) & \text{in Case B} \\ -\frac{c_{\mathrm{ER}}}{2R}\left(g_1 p_1 + g_2 p_2\right) & \text{in Case C.} \end{cases} \quad (12)$$

Note that secret RSA exponents are chosen randomly. (In many applications the public exponent is a fixed small value, e.g. 3 or $2^{16}+1$. However, d may be interpreted as a function of the randomly chosen prime factors p_1 and p_2.) It is therefore reasonable to assume

Assumption 1. $w_i \approx \log_2(p_i) \approx 0.5\log_2(n)$ and, similarly, $g_i \approx 0.5\log_2(p_i) \approx 0.25\log_2(n)$ for $i = 1, 2$.

Example 1. Let $n \approx 0.7 \cdot 2^{1024}$ *and* $R = 2^{512}$. *If we assume* $p_i/R \approx \beta$ *then* $E(X_{u_2} - X_{u_1}) \approx -107\,\mathrm{c_{ER}}$ *in Case B, and* $E(X_{u_2} - X_{u_1}) \approx -214\,\mathrm{c_{ER}}$ *in Case C.*

The basic scheme of our attack is stated below. It will be completed in Sect. 5. Note that $-\mathrm{c_{ER}}\log_2(n)\beta/16 \approx 0.5[(E(X_{u_2} - X_{u_1} \mid \text{Case A is true}) + (E(X_{u_2} - X_{u_1} \mid \text{Case B is true})]$.

The attack — basic scheme

Phase 1: Choose an integer u with $\beta R \le u < n$ and set (e.g.) $\Delta := 2^{-6}R$

$u_2 := u, \quad u_1 := u_2 - \Delta$

while $\left(T(u_2) - T(u_1) > -\mathrm{c_{ER}}\frac{\log_2(n)\beta}{16}\right)^{**}$ do {

$u_2 := u_1, \quad u_1 := u_1 - \Delta$ }

Phase 2: while $(u_2 - u_1 > 1000)$ do {

$u_3 := \lfloor (u_1 + u_2)/2 \rfloor$

if$\left(T(u_2) - T(u_3) > -\mathrm{c_{ER}}\frac{\log_2(n)\beta}{16}\right)$ then $u_2 := u_3$(decision for Case A)

else $u_1 := u_3$ (decision for Case B or C)}

Phase 3: Compute $\gcd(u, n)$ for each $u \in \{u_1 + 1, \ldots, u_2\}$.

** The attacker believes that Case A is correct

5 Error Probabilities, Error Detection, and Correction

In Sect. 4 we formulated the basic scheme of our attack. Next, we approximate the probabilities that the attacker decides for Case A although Case B or C was correct and, vice versa, that Case A was correct but the attacker decides for Case B or C. Since we need not distinguish between Cases B and C we will restrict our attention to the Cases A and B. Note that Case C is rather unlikely and if it occurs the situation for the attacker obviously is even better than in Case B.

Decisions were based on time differences $T(u_2) - T(u_1)$ (or, equivalently, on $T(u_2) - T(u_3)$, resp.) which we viewed as realizations of $N(\mu(u_1) - \mu(u_2), \sigma(u_1)^2 + \sigma(u_2)^2)$-distributed random variables $X_{u_2} - X_{u_1}$. Again we assume $u_2 - u_1 \ll p_1, p_2$, and p_j denotes the prime factor of which in Phase 2 a multiple is contained in $\{u_1 + 1, \ldots, u_2\}$. Clearly, as $u_2 (\mathrm{mod}\, p_i)$ and $u_1 (\mathrm{mod}\, p_i)$ (or $u_3 (\mathrm{mod}\, p_i)$, resp.) are not known we cannot compute the exact variances. Note that

$$u_2 (\mathrm{mod}\, p_j), u_3 (\mathrm{mod}\, p_j) \approx 0 \qquad \text{Phase 2, Case A} \qquad (13)$$

$$u_2 (\mathrm{mod}\, p_j) \approx 0,\ u_3 (\mathrm{mod}\, p_j) \approx p_j \qquad \text{Case B} \qquad (14)$$

which will be put in the respective variance and covariance terms of (10). Otherwise the factor $u_k (\mathrm{mod}\, p_i)$ is not under control so that we approximate the respective variance term $\mathrm{pr}_i(u_k)(1 - \mathrm{pr}_i(u_k))$ by its average value

$$\frac{2R}{p_i}\int_0^{p_i/2R} x(1-x)\,dx = \frac{p_i}{4R} - \frac{p_i^2}{12R^2} \quad . \qquad (15)$$

Then we approximate p_1/R and p_2/R by β. Similarly, the covariance terms $\mathrm{cov}_{i;\mathrm{MQ}}(u_k)$ and $\mathrm{cov}_{i;\mathrm{QM}}(u_k)$ are approximated by their average values $\beta^4/48 - \beta^2/12$ and $\beta^3/20 - \beta^2/12$. Further, using Assumption 1 elementary computations lead to $\sigma_{\mathrm{MM}}(u_1)^2 + \sigma_{\mathrm{MM}}(u_2)^2$ (or, $\sigma_{\mathrm{MM}}(u_3)^2 + \sigma_{\mathrm{MM}}(u_2)^2$, resp.)

$$\approx \begin{cases} \log_2(n){c_{\mathrm{ER}}}^2\left(\frac{11\beta}{12} - \frac{31\beta^2}{36} + \frac{\beta^3}{10} + \frac{23\beta^4}{168}\right) & \text{Phase 1, Case A} \\ \log_2(n){c_{\mathrm{ER}}}^2\left(\frac{19\beta}{24} - \frac{47\beta^2}{72} + \frac{\beta^3}{20} + \frac{13\beta^4}{112}\right) & \text{Phase 2, Case A} \\ \log_2(n){c_{\mathrm{ER}}}^2\left(\frac{11\beta}{12} - \frac{127\beta^2}{144} + \frac{\beta^3}{10} + \frac{53\beta^4}{336}\right) & \text{Case B .} \end{cases} \tag{16}$$

As usually, let $\Phi(\cdot)$ denote the cumulative distribution function of the $N(0,1)$-distribution. From (12) we derive the approximate average error probability for a single decision

$$p_{\mathrm{err}} \approx \Phi\left(-\frac{c_{\mathrm{ER}} \log_2(n)\beta}{16\sqrt{\sigma(u_{1|3})^2 + \sigma(u_2)^2}}\right) . \tag{17}$$

Example 2. If ${\sigma_{\mathrm{CRT}}}^2$ is negligible average error probabilities (Phase 1, Case A; Phase 2, Case A; Case B) are about
(i) 0.00094, 0.00097, and 0.00087 for $n \approx 0.7 \cdot 2^{1024}$ and $R = 2^{512}$.
(ii) 0.00022, 0.00027, and 0.00021 for $n \approx 0.9 \cdot 2^{1024}$ and $R = 2^{512}$.
(iii) 0.000005, 0.000005, and 0.000005 for $n \approx 0.7 \cdot 2^{2048}$ and $R = 2^{1024}$.

The error probability for a single decision decreases if the modulus n or the parameter β increases. Although for realistic modulus size n the probability for an erroneous decision is rather small we cannot be sure that all decisions within an attack are correct. (Of course, in Phase 2 a large sequence of decisions for Case A may be an indicator for an erroneous decision in the past.) However, at any instant within Phase 2 we can verify with high probability whether our decisions have been correct so far, i.e. whether the interval $\{u_1+1, \ldots, u_2\}$ really contains a multiple of p_1 or p_2. We just have to apply our decision rule from the basic scheme in Sect. 4 to a time difference for neighbouring values of u_1 and u_2, resp., e.g. to $T(u_2 - 1) - T(u_1 + 1)$. If this leads to the same decision it is confirmed with overwhelming probability that the interval $\{u_1 + 1, \ldots, u_2\}$ truly contains a multiple of p_1 or p_2. (Thus, we call $\{u_1 + 1, \ldots, u_2\}$ a *confirmed interval.*) Otherwise, evaluate a third time difference. If the third decision confirms that $\{u_1 + 1, \ldots, u_2\}$ contains a multiple of p_1 or p_2 then we have established a confirmed interval after all. If not, we have to go back to the preceding confirmed interval or at least to the first "close" decision thereafter to restart the attack at this point with a neighbouring value $\bar{u}$ of the one previously used. Anyway, it is indispensable to complete the basic scheme by regular attempts to establish confirmed intervals, the first at the beginning if Phase 2.

Now let suggestively denote $p_{\mathrm{err;1A}}$, $p_{\mathrm{err;2A}}$ and $p_{\mathrm{err;B}}$ the error probabilities for a single decision in the various cases. If we choose the starting value u_1 randomly, for $\Delta = 2^{-6}R$ Phase 1 requires $(64\beta/3) + 1$ time measurements on average if all decisions are correct. An erroneous decision for Case B or C within

Phase 1 should (at a cost of 4 time measurements) immediately be detected when trying to establish $\{u_1+1,\ldots,u_2\}$ as a confirmed interval. If Case B or C is correct (Phase 1 could end here!) an error costs some extra time measurements but the attack will find a multiple of the other prime factor. Altogether, Phase 1 costs about

$$1+\frac{64\beta}{3}(1+4p_{\mathrm{err;1A}})(1-p_{\mathrm{err;B}})\sum_{k=1}^{\infty}kp_{\mathrm{err;B}}^{k-1}=1+\frac{64\beta}{3}\cdot\frac{1+4p_{\mathrm{err;1A}}}{1-p_{\mathrm{err;B}}} \tag{18}$$

time measurements. Now assume that the attacker wants to establish a confirmed interval each time after s decisions. If s consecutive decisions after a confirmed interval are correct it requires $s+2$ (sometimes $(s+4)$) time measurements to establish the next confirmed interval. This event occurs with average probability $\bar{q}\approx(1-\bar{p}_{\mathrm{err}})^s$ where $\bar{p}_{\mathrm{err}}=0.5(p_{\mathrm{err;2A}}+p_{\mathrm{err;B}})$. If any of the s decisions was wrong the attacker has performed $s+4$ time measurements "for nothing" as he has to restart his attack from the preceding confirmed interval. Similar as above one concludes that establishing a confirmed interval within Phase 2 costs about $(s+4)/\bar{q}-2\bar{q}$ time measurements. Consequently, on average the whole attack (Phase 1 and Phase 2) needs about

$$1+\frac{64\beta}{3}\cdot\frac{1+4p_{\mathrm{err;1A}}}{1-p_{\mathrm{err;B}}}+\frac{0.5\log_2(n)-16}{s}\left(\frac{s+4}{\bar{q}}-2\bar{q}\right) \tag{19}$$

time measurements. The attacker should, of course, choose a parameter value s for which (19) is minimal. If we assume that ${\sigma_{\mathrm{CRT}}}^2$ is negligible for $n\approx 0.7\cdot 2^{1024}$ and $R=2^{512}$, for example, the parameter $s=46$ is optimal. The whole attack then requires about $560\approx 0.55\log_2(n)$ time measurements on average. Similarly, for $n\approx 0.7\cdot 2^{512}$ ($n\approx 0.5\cdot 2^{1024}$, $n\approx 0.9\cdot 2^{1024}$, $n\approx 0.7\cdot 2^{2048}$) for $s=11$ ($s=22$, $s=91$, $s=625$) about $0.71\log_2(n)$ ($0.60\log_2(n)$, $0.53\log_2(n)$, $0.51\log_2(n)$) time measurements are necessary. We did neither take the rare event into account that it erroneously failed to establish a confirmed interval (due to two wrong verifying decisions) nor that the preceding confirmed interval was erroneously established (see Remark 5) as their influence on (19) is small.

Remark 5. (i) If it fails to establish a confirmed interval at a certain stage of the attack for the third time it seems to be likely that the preceding confirmed interval had erroneously been confirmed (rare event!). To avoid a deadlock one simply jumps back to the last one confirmed interval.
(ii) Neighbouring values of the optimal parameter s do not yield considerably worse results.

Remark 6. In Phase 2 of our attack we successively recover the binary representation of an integer multiple of one prime factor p_j. If the attacker starts with $u_1\in[\beta R,R]$ it is likely (for $\beta>\sqrt{0.50}$ it is sure) that he will find p_j rather than a multiple of it. Indeed, if the attacker knows at least $0.25\log_2(n)$ high-order bits of p_j he may refrain from further time measurements but compute the remaining bits with a lattice-based algorithm introduced in [2] (Sect. 10 and 11). Its

running time is polynomial in $\log_2(n)$. Making use of Coppersmith's algorithm obviously nearly halves the number of time measurements. As $u(\mathrm{mod}\, p_j) = u$ for $u < p_j$ due to (7) we recommend to avoid values u_1, u_2, u_3 which are multiples of large powers of 2.

Remark 7. (i) As the terms $\Psi_i(y_i)$ and $\Psi_{i*}(\mathrm{temp})$ usually are interpreted as Montgomery multiplications with factors $a' := y_i$ and $b' := R^2(\mathrm{mod}\, p_i)$ (precomputed value!) and $a' :=$ temp and $b' := 1$, resp., their cumulative variance is negligible. The variance of the reductions $y \mapsto y(\mathrm{mod}\, p_i)$ (e.g. computed with Barrett's reduction algorithm) and of the final CRT computation $b_1x_1 + b_2x_2(\mathrm{mod}\, n)$ depends on the chosen algorithms but should be small in general.
(ii) So far we have tacitly assumed that the attacker is able to measure the running times exactly. A $N(0, {\sigma_{\mathrm{Err}}}^2)$-distributed random measurement error increases the variance of $X_{u_2} - X_{u_1}$ by $2{\sigma_{\mathrm{Err}}}^2$ which in turn increases error probability. Similarly, approximately normally distributed random external influences (or, equivalently, randomly chosen dummy operations) increase the variance of $X_{u_2} - X_{u_1}$ by $2{\sigma_{\mathrm{Ext}}}^2$.
(iii) If ${\sigma_{\mathrm{Err}}}^2 + {\sigma_{\mathrm{Ext}}}^2 + {\sigma_{\mathrm{CRT}}}^2$ is not negligible this does not prevent our attack. In fact, $E(X_{u_2} - X_{u_1})$ and thus the decision rule remains unchanged and (17) remains valid. (Of course, if $\gamma := 2({\sigma_{\mathrm{Err}}}^2 + {\sigma_{\mathrm{Ext}}}^2 + {\sigma_{\mathrm{CRT}}}^2)/(\sigma_{\mathrm{MM}}(u_{1|3})^2 + \sigma_{\mathrm{MM}}(u_2)^2)$ is too large the attack may become *practically infeasible*.) Anyway, the attacker has to establish more confirmed intervals. For $n \approx 0.7 \cdot 2^{1024}$, $R = 2^{512}$ and $\gamma = 1.0$, e.g., using $s = 11$ requires about 730 time measurements. For large γ it may be necessary to apply sequential sampling methods (see Sect. 7) or at least to apply the decision rule from the basic scheme at each step separately to three time differences, e.g. to $T(u_2) - T(u_1), T(u_2+1) - T(u_1-1)$ and $T(u_2+2) - T(u_1-2)$ (reuse existing time measurements!). The attacker decides for the majority of these (pre-)decisions which doubles to triples the number of time measurements while the probability for a wrong decision decreases from p_{err} to $(3 - 2p_{\mathrm{err}}) * {p_{\mathrm{err}}}^2$.

6 Experimental Results

We implemented modular exponentiation with CRT and Montgomery multiplication in software. Output was the total number of extra reductions within the Montgomery multiplications. (Recall that we are only interested in time differences.) This scenario corresponds to a "real" timing attack where $\sigma^2_{\mathrm{Err}} + \sigma^2_{\mathrm{Ext}} + {\sigma_{\mathrm{CRT}}}^2$ is negligible since then differences in running times are proportional to differences in the respective numbers of extra reductions. Note that the error probabilities then do not depend on the constants c and c_{ER}. Hence its efficiency does not depend on specific hardware characteristics of the attacked device.

We carried the attack through for various 1024 bit moduli with randomly chosen primes $\in [0.8, 0.85] * 2^{512}$, private key $d = 3^{-1}(\mathrm{mod}\ (p_1-1)(p_2-1))$, and $R = 2^{512}$. We always started our attack with $u = 2R$. We used the basic scheme

introduced at the end of Sect. 4. Additionally, we established confirmed intervals at the beginning of Phase 2 and then always after 42 decisions. If it failed to establish a confirmed interval at the same stage of the attack for the third time we restarted the attack from the last but one confirmed interval (see Remark 5). On average, about 570 time measurements were needed to carry through an attack. All attacks were successful. We did not make use of Coppersmith's algorithm mentioned in Remark 6. Coppersmith's algorithm would have reduced the average number of time measurements to about 300.

7 Extension to Advanced Exponentiation Algorithms

Many RSA-implementations use more efficient exponentiation algorithms than square and multiply (see e.g. [Mov], Sect. 14.6.1). Therefore, usually b-bit-tables ($b > 1$) are generated which contain powers of the respective base. Combined with CRT and Montgomery's algorithm b-bit table i ($i = 1, 2$) stores $\Psi_i(v)(= u(\mathrm{mod}\, p_i)), \Psi_i(v^2), \ldots, \Psi_i(v^{2^b-1})$ with $v = uR^{-1}(\mathrm{mod}\, n)$, or at least a subset of these values. Using a b-ary exponentiation scheme ([Mov], Alg. 14.82 and 14.83) b modular squarings are accompanied by only one multiplication with a table entry. Our attack can be extended to those table methods. The underlying idea is simple but due to lack of space we can only sketch the technical details. For the sake of efficiency we recommend to make use of Coppersmith's algorithm.

For b-ary exponentiation schemes the essential part of the exponentiation $(uR^{-1})^d(\mathrm{mod}\, n)$ requires about $\log_2(n) - 2$ squarings and $\log_2(n)/b2^{b+1}$ Montgomery multiplications with both $u(\mathrm{mod} p_1)$ and $u(\mathrm{mod} p_2)$. Additionally, $\log_2(2^b - 2)/b2^b$ multiplications with $\Psi_i(v^k)$ are necessary where (i, k) range through $\{1, 2\} \times \{2, \ldots, 2^b - 1\}$. If the table entries are calculated straightforward, i.e. if $\Psi_i(v^{k+1}) = \Psi_{i*}(\Psi_i(v^k)\Psi_i(v))$ for $k = 2, \ldots, 2^b - 1$ ([Mov], Alg. 14.82), then this additionally costs $2^b - 3$ Montgomery multiplications with both $u(\mathrm{mod}\, p_1)$ and $u(\mathrm{mod}\, p_2)$. (Concerning the probability for an extra reduction the computation of $\Psi_i(v^2) = \Psi_{i*}(\Psi_i(v)^2)$ should be viewed as a squaring.)

As the standard variant (attacking the square and multiply exponentiation scheme) also the extended version exploits time difference $T(u_2) - T(u_1)$. A careful computation yields

$$E(X_{u_2} - X_{u_1}) \approx \begin{cases} 0 & \text{in Case A} \\ -\frac{\mathrm{c_{ER}} p_i}{2R}\left(\frac{\log_2(n)}{b2^{b+1}} + 2^b - 3\right) & \text{in Case B} \\ -2\frac{\mathrm{c_{ER}} p_i}{2R}\left(\frac{\log_2(n)}{b2^{b+1}} + 2^b - 3\right) & \text{in Case C.} \end{cases} \tag{20}$$

Assuming $p_i/R \approx \beta$ we derive the following *predecision rule*: Decide for case A iff $T(u_2) - T(u_1) < -0.25\mathrm{c_{ER}}\beta\left(\frac{\log_2(n)}{b2^{b+1}} + 2^b - 3\right)$.

Similarly as in (10) we first express $\sigma_{\mathrm{MM}}(u)^2$ as a sum of variances and covariances. Especially, $\mathrm{cov}_{i;\mathrm{MM}}(u) = 4\mathrm{pr}_i(u)^3 - \mathrm{pr}_{*i}^2$ with average value $\beta^3/24 - \beta^2/12$. Note that the particular variances and covariances equal the expressions in (10) with $\Psi_i(v^k)/2R$ instead of $\mathrm{pr}_i(u)$. As in Sect. 5 we approximate these

variance and covariance terms in $\sigma_{\mathrm{MM}}(u_{1|3})^2+\sigma_{\mathrm{MM}}(u_2)^2$ by their average values unless $\Psi_i(v^k) = u_k(\bmod p_j)$ where we use (13) and (14). For simplicity we again assume that ${\sigma_{\mathrm{CRT}}}^2$ is negligible. Then for $n \approx 0.7 \cdot 2^{1024}$, $R = 2^{512}$ and $b = 2$, for example, we obtain approximate error probabilities 0.204 (Phase 1, Case A), 0.204 (Phase 2, Case A), and 0.204 (Case B).

To perform a successful attack, however, we need more trustworthy decisions. Therefore we apply the predecision rule to several time differences $T(u_{2(1)}) - T(u_{1(1)}), T(u_{2(2)}) - T(u_{1(2)}), \ldots$ where $u_{1(1)}, u_{1(2)}, \ldots$ and $u_{2(1)}, u_{2(2)}, \ldots$ denote arbitrary bases (if possible, reuse existing time measurements) within intervals I_1 and I_2 around the lower and the upper bound of the interval $[u_1, u_2)$ (or $[u_3, u_2)$ within Phase 2 of our attack, resp.). (As we use Coppersmith's algorithm $u_2 - u_1, u_2 - u_3 \geq R^{0.5}$ the interval lengths may be chosen fairly large.) To minimize the number of time measurements we use sequential sampling. That is, we proceed applying our predecision rule until the number of predecisions for Case A and the number against Case A differ by a fixed integer $N > 1$. Our decision rule, of course, is to accept the majority of these predecisions. The probability for a wrong decision and the expected number of predecisions follow from formulas 2.4 and 3.4 in Chap. XIV of [4]. (Therefore we interpret the decision procedere as a gambler's ruin problem with initial capital N. A wrong predecision reduces, a correct predecision increases the capital by 1 unit.) For our example we choose $N = 3$ which leads to approximate error probabilities for a single decision $p_{\mathrm{err,1A}} \approx 0.017$, $p_{\mathrm{err,2A}} \approx 0.017$ and $p_{\mathrm{err,B}} \approx 0.016$. The expected number of predecisions needed per decision is about 6.0 in all cases.

The attack itself is organised as in Sect. 4 and 5. Replacing the nominator "$0.5\log_2(n)-16$" of (19) by "$0.25\log_2(n)-6$" (we use Coppersmith's algorithm), choosing a parameter s minimizing this term (in our example, $s = 10$) and multiplying the obtained result by the average numbers of predecisions yields a first estimate for the total number of time measurements needed for our attack (=1920 in our example). However, decisions which are not used to establish a confirmed interval reuse existing time measurements. Of course, if the respective earlier decision had needed fewer predecisions this obviously costs some additional time measurements. (Its mean value may be derived from the generating function for the number of predecisions needed for one decision ([4], Sect. XIV.4) or simply be estimated with a stochastic simulation.) In our example we have to augment the number of time measurements from 1920 to about 2320. For $b = 3$, $b = 4$ and $b = 5$ altogether about 11160 (with $N = 5$ and $s = 5$), 17700 (with $N = 7$ and $s = 6$), 7050 (with $N = 4$ and $s = 5$) time measurements, resp., are necessary if again $n \approx 0.7 \cdot 2^{1024}$ and $R = 2^{512}$. The last result may be surprising at first sight but for $b = 5$ computing the tables entries requires 29 multiplications with both $u(\bmod p_1)$ and $u(\bmod p_2)$.

To improve efficiency at least for $b = 5$ a modified b-ary exponentiation may be used ([Mov], Alg. 14.83) which only stores $\Psi_i(y^k)$ for odd exponents k. Building up table i costs $2^{b-1} - 1$ multiplications with $\Psi_i(v^2)$ and one squaring with $\Psi_i(v)$. For $b = 5$ (and $n \approx 0.7 \cdot 2^{1024}$) the attack requires about 28200 time measurements. For $b = 4$ the situation for the attacker is more comfortable than

in the non-modified case since on average about 32 Montgomery multiplications with both $u(\bmod p_1)$ and $u(\bmod p_2)$ are carried out. About 7250 time measurements should be sufficient. We finally remark that sliding window exponentiation ([Mov], Alg.14.85) can be attacked with similar methods.

8 Fields of Application

As our attack requires chosen input it cannot used be used to attack signature applications with fixed padding. Our attack works, however, if the attacker can choose the complete base y provided that there is no integrity check at all or if random padding bits are used (to prevent the Bellcore attack ([1])), eventually combined with mild integrity conditions (e.g. given by two information bytes). If the attacker then would like to measure $T(u)$ he first uses a PC or a laptop to determine the integer z with smallest absolute value such that $(u+z)R^{-1}(\bmod n)$ meets the integrity condition. Then he measures $T(u+z)$ instead of $T(u)$.

However, our attack can always be applied if $y^d(\bmod n)$ decrypts a secret message $y := r^e(\bmod n)$ where e denotes the public key of the recipient. (The message r might contain a symmetric session key, for example.) Of course, the integrity of r cannot be checked until the exponentiation $y^d(\bmod n)$ has been carried out. However, we are not interested in $y^d(\bmod n)$ itself but in the respective running time (eventually inclusive an integrity check). Hence we do not need to care about integrity conditions.

9 Countermeasures

The most obvious way to prevent our attack is to carry out an extra reduction within each Montgomery multiplication (if not needed by the algorithm then as a dummy operation). Alternatively, provided that R is sufficiently large (compared with the moduli p_i) the extra reduction step may be missed out entirely ([10,5]). In fact, the intermediate results of the respective exponentiation algorithm then are bounded by $2p_i$ and the reduction will be automatically carried out within the final operation temp $\mapsto \Psi_{i*}(\text{temp})$. Using any of these countermeasures, of course, it has to be taken care that eventual time differences caused by the remaining arithmetical operations do not reveal the factorization of n or the secret exponent d.

A more general approach to prevent timing attacks is to use blinding techniques ([6], Sect. 10). Instead of $y^d(\bmod n)$ the device then internally computes $(yh_a(\bmod n))^d(\bmod n)$, followed by a modular multiplication with $h_b := h_a^{-d}(\bmod n)$. To protect the blinding factors h_a and h_b themselves against timing attacks they are updated before the next exponentiation via $h_a \mapsto h_a^2(\bmod n)$ and $h_b \mapsto h_b^2(\bmod n)$.

10 Concluding Remarks

In this article we introduced and investigated a new type timing attack which works if RSA exponentiation with the secret exponent uses CRT and Montgomery's algorithm. Our attack is very efficient and (at the expense of efficiency) tolerates measurement errors and variance caused by arithmetical operations besides the Montgomery multiplications or external influences. The central idea of our attack may also be transferred to CRT implementations using other multiplication algorithms than Montgomery's provided that mean or variance of the time needed for a multiplication of $u \in Z_{p_i}$ with a randomly chosen cofactor is significantly different for $u \approx 0$ and $u \approx p_i$. As a consequence, also for RSA applications using the CRT either constant running times or blinding techniques are indispensable.

Acknowledgement

The author wants to thank the referees for valuable comments and suggestions which helped to improve the presentation of this paper.

References

1. D. Boneh, R.A. Demillo, R.J. Lipton: On the Importance of Checking Cryptographic Protocols for Faults. In: W. Fumy (ed.): Advances in Cryptology — Eurocrypt '97, Lecture Notes in Computer Science, Vol. **1233**. Springer, New York (1997) 37–51.
2. D. Coppersmith: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptology **10** (no. 4) (1997) 233–260.
3. J.-F. Dhern, 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, B. Schneier (eds.): CARDIS 1998, Third Smart Card Research and Advanced Application Conference (PreProceedings). Université catholique de Louvain (1998).
4. W. Feller: An Introduction to Probability Theory and Its Applications (Vol. 1). 3rd edition, revised printing, Wiley, New York (1970).
5. G. Hachez, J.-J. Quisquater: Montgomery Exponentiation with no Final Subtractions: Improved Results. To appear in: Ç.K. Koç, C. Paar (eds.): Workshop on Cryptographic Hardware and Embedded Systems, 2000 (Proceedings).
6. P. Kocher: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS and Other Systems. In: N. Koblitz (ed.): Advances in Cryptology – Crypto '96, Lecture Notes in Computer Science, Vol. **1109**. Springer, Heidelberg (1996), 104–113.
7. A.J. Menezes, P.C. van Oorschot, S.C. Vanstone: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997).
8. P.L. Montgomery: Modular Multiplication without Trial Division. Math. Comp. **44** (no. 170) (1985) 519–521.
9. W. Schindler: Optimized Timing Attacks against Public Key Cryptosystems. Submitted.
10. C.D. Walter: Montgomery's Exponentiation Needs no Final Subtractions. Electronics letters **35** (no. 21) (1999), 1831–1832.