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

Werner Schindler
Bundesamt für Sicherheit in der Informationstechnik (BSI)
Bonn, Germany

Saint-Malo, September 15, 2015

Outline

- State of the art and motivation
- A new timing attack
  - Attack scenario
  - Theoretical Background
  - Attack algorithm
  - Empirical Results
- Countermeasures
- Conclusion

Timing Attacks on RSA

- Timing attacks on RSA without CRT
  - Kocher (Crypto 1996) [pioneer work]
  - Dhem, Koeune, Leroux, Mestré, Quisquater, Willems (Cardis 1998)
  - Schindler, Koeune, Quisquater (Cryptography and Coding 2001)
- Timing attacks on RSA with CRT
  - Schindler (CHES 2000)
  - Brumley, Boneh (Usenix 2003)
  - Aciiçmez, Schindler, Quisquater (CCS 2005)

NOTE: All these timing attacks are only applicable to unprotected implementations.

Algorithmic countermeasures against side channel attacks

- Base blinding (Kocher 1996)
- Exponent blinding (Kocher 1996)
- Modulus blinding
- Combination of blinding techniques
- ...

Crucial question in the context of security evaluations:
Are these blinding techniques effective against side channel attacks?

Side channel Attacks on blinded implementations

- Aciiçmez, Schindler (2007, 2008): Instruction cache attack on OpenSSL v.0.9.8e, RSA with CRT, *base blinding*
- Fouque et al. (2006), Bauer (2012): Power attacks on RSA without CRT, *exponent blinding*
- Schindler, Itoh (2011), Schindler, Wiemers (2014, 2015): Generic power attacks on *exponent blinding* (RSA, with and without CRT) and *scalar blinding* (ECC), also in combination with base blinding

- It has widely been assumed that blinding techniques would effectively prevent (pure) timing attacks.
- For exponent blinding this assumption is not true in general.

(Additive) exponent blinding

- RSA with CRT
  - $n = p_1 p_2$
  - d = private exponent
  - $d_i = d (\bmod (p_i - 1))$
  - $r_{i,j} \in \{0, \ldots, 2^{eb} - 1\}$ (eb-bit random number = j^{th} blinding factor for the exponentiation modulo p_i)
  - for $i = 1, 2$ compute $y^{d_i + r_{i,j}(p_i - 1)} (\bmod p_i)$ in place of $y^{d_i} (\bmod p_i)$

- Exponent blinding shall prevent that an attack can focus on particular exponent bits.

Montgomery's multiplication algorithm (MM)

- Input: M modulus, $a, b \in \mathrm{Z_M} := \{0, 1, \ldots, \mathrm{M} - 1\}$
- Output: $\mathrm{MM}(a, b; M) := abR^{-1}(\text{mod } M)$

  $M < R = 2^x$ ($R =$ Montgomery constant)

1. $s := 0$
2. for $i = 0$ to $v - 1$ do {

   $u := (s + a_i b_0)m^*(\text{mod } r)$ /* r-adic representation*/

   $s := (s + a_i b + uM)/r$ /* $r = 2^{ws}$*/

   }
3. If $(s \geq M)$ then $s := s - M$ [= extra reduction (ER)]
4. return $\mathrm{MM}(a, b; M)$

- The extra reduction causes timing differences.

Pseudoalgorithm: RSA with CRT, MM, exponent blinding

1.
   - $y_1 := y (\bmod\ p_1)$ and $d_1 := d (\bmod\ (p_1 - 1))$
   - (Exponent blinding) Generate the blinded exponent $d_{1,b} := d_1 + r_1\phi(p_1) = d_1 + r_1(p_1 - 1)$.
   - Compute $v_1 := y_1^{d_{1,b}} (\bmod\ p_1)$ (expo algorithm with MM).
2.
   - $y_2 := y (\bmod\ p_2)$ and $d_2 := d (\bmod\ (p_2 - 1))$
   - (Exponent blinding) Generate the blinded exponent $d_{2,b} := d_2 + r_2\phi(p_2) = d_2 + r_2(p_2 - 1)$.
   - Compute $v_2 := y_2^{d_{2,b}} (\bmod\ p_2)$ (expo algorithm with MM).
3. (Recombination) Compute $v := y^d (\bmod\ n)$ from (v_1, v_2), e.g. with Garner's algorithm

Theoretical background (I)

- Our attack targets the exponentiation steps
  - Compute $v_1 := y_1^{d_{1,b}} (\bmod p_1)$
  - Compute $v_2 := y_2^{d_{2,b}} (\bmod p_2)$
- In the following we assume

$$\mathrm{Time}(\mathrm{MM}(a,b;p_i)) \in \{c, c + c_{\mathrm{ER}}\} \quad \text{for all } a,b \in Z_{p_i}$$

c = time for MM without extra reduction

c_{ER} = time for an extra reduction

- $\mathrm{Time}(v_i := y_i^{d_{i,b}} (\bmod p_i)) =$ const $+ c * \#$(squarings and multiplications) $+ c_{\mathrm{ER}} * \#$ERs.

Theoretical background (II)

- Central task: Understand how the blinding and the input data affect the number of squarings, multiplications and ERs.
- Problems & Difficulties: The moduli p_i and the bases $y_i = y (\text{mod } p_i)$ are unknown. Addititionally to the unblinded case the secret exponents $d_{i,b}$ change in every exponentiation.

Theoretical background (III)

- Our attack is an adaptive chosen-input attack with input values $y_u := uR^{-1}(\bmod n)$.
- The execution times $\mathrm{Time}((y_u)^d(\bmod n))$ are interpreted as realizations of a random variable $Z(u)$.
- The computation of $E(Z(u))$ and $\mathrm{Var}(Z(u))$ requires extensive calculations (details: paper).

- We assume $0 < u_1 < u_2 < n$ and $u_2 - u_1 \ll p_1, p_2$. Three cases are possible:
  - 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.

Theoretical background (IV)

- For square & multiply exponentiation we have

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

- This property allows to construct a distinguisher to decide whether some interval $(u_1, u_2]$ contains a multiple of p_1 or p_2. The decision boundary is given by

$$\mathrm{decbound} := -\frac{1}{8}\left((\log_2(\mathrm{R}) + \mathrm{eb} - 1)\frac{\sqrt{\mathrm{n}}}{\mathrm{R}} - 1\right) \mathrm{c}_{\mathrm{ER}}$$

The distinguisher

- Since $\mathrm{Var}(Z(u_2) - Z(u_1))$ is large each individual decision requires many timing measurements.

$$\mathrm{MeanTime}(\mathrm{u}, \mathrm{N}) := \frac{1}{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{N}} \mathsf{Time}(\mathrm{y}_\mathrm{j}^\mathrm{d} (\mathsf{mod}\ \mathrm{n}))$$

with $y_j := uR^{-1} (\mathrm{mod}\ n)$

- **Decision rule:**
  - If $(\mathrm{MeanTime}(\mathrm{u}_2, \mathrm{N}) - \mathrm{MeanTime}(\mathrm{u}_1, \mathrm{N}) > \mathrm{decbound})$ decide for
    '$(u_1, u_2]$ *does not contain a multiple of* p_1 *or* p_2'
  - else decide for
    '$(u_1, u_2]$ *contains a multiple of* p_1 *or* p_2'.

Bundesamt für Sicherheit in der Informationstechnik

The Attack: Phase 1

Goal: Find an interval, which contains the larger prime p_2.

Set (e.g.) $u_1 := \lfloor\sqrt{n}\rfloor$ and $\Delta := 2^{-6}R$
$u_2 := u_1 + \Delta$
while $(\mathrm{MeanTime(u_2, N) - MeanTime(u_1, N) > decbound})$
do$^*\{$
$u_1 := u_2, u_2 := u_2 + \Delta$
$\}$

$* \equiv$ The attacker believes that Case A is correct

- Status: The interval $(u_1, u_2]$ contains p_2.

The Attack: Phase 2

- Action: Adjust $\mathrm{decbound}$ ($\leftarrow$ more precise info on p_2)
- Strategy: Bisect $(u_1, u_2]$ until a little bit more than the upper halve of the bits of p_2 are known.

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

- $* \equiv$ The attacker believes that Case A is correct

- Status: The interval $(u_1, u_2]$ contains p_2, and $\log_2(u_2 - u_1) \approx 0.5\log_2(p) - 10$.

The Attack: Phase 3

- Determine p_1 and p_2 with Coppersmith's algorithm (1997)

- **NOTE** This attack algorithm is rather similar to the algorithm for unblinded implementations.

Scaling

- Let $eb \ll \log_2(R)$ and $\sigma_N^2(=$ variance of additional noise$) \approx 0$.
  - The overall number of timing measurements is to a large extent independent of the size of the RSA modulus n
  - The number of timing measurements increases as $O\left(\left(\frac{c_{\mathrm{ER}}}{c}\right)^{-2}\right)$.
- The attack efficiency increases as p_2/R increases.
- Our attack may even tolerate minor formatting restrictions, which affect some input bits.

Experimental Results (I)

- Simulation results for $\sigma_N^2 = 0$ (no additional noise)
- square & multiply exponentiation algorithm (s&m)

$\log_2(R)$	eb	c_{ER}/c	$\frac{p_1}{R}$	$\frac{p_2}{R}$	success	$av.\#$expos
512	64	0.02	0.75	0.85	24/25	830,000
512	64	0.025	0.75	0.85	24/25	541,000
512	64	0.03	0.75	0.85	24/25	395,000
512	64	0.05	0.75	0.85	25/25	140,000
512	64	0.05	0.70	0.70	24/25	203,000
512	64	0.05	0.80	0.80	24/25	141,000
512	64	0.05	0.85	0.85	25/25	140,000
512	64	0.05	0.90	0.90	23/25	127,000

Table: Simulation results: 512-bit primes

Experimental Results (II)

$\log_2(R)$	eb	$c_{\mathrm{ER}}\,/\,c$	$\frac{p_1}{R}$	$\frac{p_2}{R}$	success	$av.\#$expos
512	64	0.02	0.75	0.85	24/25	830,000
512	64	0.025	0.75	0.85	24/25	541,000
512	64	0.03	0.75	0.85	24/25	395,000
512	64	0.05	0.75	0.85	25/25	140,000
768	64	0.03	0.75	0.85	23/25	382,000
768	64	0.05	0.75	0.85	23/25	139,000
1024	64	0.025	0.75	0.85	24/25	590,000
1024	64	0.03	0.75	0.85	24/25	410,000
1024	64	0.05	0.75	0.85	24/25	152,000

Table: Simulation results: 512-bit primes, 768-bit primes, and 1024-bit primes; s&m, $\sigma_N^2 = 0$

Extension to table-based exponentiation algorithms

- Our attack works against table-based exponentiation algorithms as well.
- The efficiency decreases because the signal-to-noise ratio drops down.
- The table provides the number of timing measurements in multiples of the figures for the s&m case.

algorithm ‖window size	$b=2$	$b=3$	$b=4$	$b=5$	$b=6$
fixed window exp.	$16\times$	$104\times$	$277\times$	$189\times$	$59\times$
sliding window exp.	$8\times$	$54\times$	$322\times$	$1032\times$	$240\times$

Table: 2048-bit RSA, 64-bit blinding, $p/R \approx 0.8$, $\sigma_N^2 = 0$; coarse estimates

Countermeasure

- If $R > 4p_1, 4p_2$ one may entirely resign on the extra reductions (Walter 2002). This is the most solid countermeasure and was e.g. selected 2007 for OpenSSL as response on an I-cache attack.
- Combining exponent blinding with base blinding prevents this timing attack, too. However, the first option is clearly preferable since it definitely prevents any timing attack.

- **NOTE:** Larger blinding factors do not prevent our attack!

Conclusion

- It has been assumed that (exclusive) exponent blinding would prevent any timing attack on RSA.
- The presented attack shows that this assumption is not true in general.
- In the presence of moderate noise this attack is practical against s&m exponentiation.
- The attack is also applicable against table-based exponentiation algorithms, though with significant lower efficiency.
- Fortunately, effective countermeasures exist.

Contact

Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany

Werner Schindler

P.O. Box 200363, 53133 Bonn, Germany
Tel.: +49 (0)228-9582-5652
Fax: +49 (0)228-10-9582-5652

Werner.Schindler@bsi.bund.de

https://www.bsi.bund.de

https://www.bsi-fuer-buerger.de