Federal Office
for Information Security

A New Side-Channel Attack on RSA Prime Generation

Thomas Finke, Max Gebhardt, Werner Schindler

Federal Office for Information Security (BSI), Germany

Lausanne, September 7, 2009

Federal Office for Information Security

Outline

- Introduction and Motivation
- The Attack
  - Basic attack
  - Refinements
- Efficiency (empirical results)
- Experimental results
- Countermeasures
- Final remarks

Federal Office for Information Security

Facts (I)

- Side-channel attacks on RSA implementations have a long tradition.
- Nearly all of these attacks aim at the exponentiation with the private key. Only a few papers consider the key generation process (e.g., Clavier & Coron, 2006).

Facts (II)

- If a smart card generates an RSA key outside the personalisation environment the key generation process may be vulnerable by side-channel attacks.

Side-channel attacks on RSA key generation

- Compared to side-channel attacks within the exponentiation phase the prospects for the attacker seem to be worse since
  - the key is generated only once
  - the generation process does not use any (known or chosen) external input
- The type of the weaknesses and their exploitation may be different from side-channel attacks within the exponentiation phase.

Motivation

- We present a side-channel attack on the RSA key generation process on a straight-forward implementation (proposed e.g. by Brandt et al. (1991), cf. also RSAREF toolkit)
- The goal of our paper is two-fold, namely
  - to demonstrate the fundamental vulnerability of the RSA key generation process against side-channel attacks.
  - to encourage the community to study the key generation process with regard to side-channel attacks

Definition

- $T = \{r_2 := 3, 5, 7, \ldots, r_N\}$
  /* trial base, consists of the first odd N-1 primes */

Federal Office for Information Security

Prime generation algorithm (I)

1. Generate an odd (pseudo-) random number $v \in \{2^{k-1}+1, \ldots, 2^k\}$

2.

   a) $i := 2;$

   b) while ($i \leq N$) do { /*trial divisions*/

```
        if (r_i divides v) then {
                v := v+2;
                GOTO Step 2a; }
        i++;
}
```

Federal Office
for Information Security

Prime generation algorithm (II)

c) m := 1;

d) while (m ≤ t) do {

/* t = max # of primality tests */

apply the Miller-Rabin primality test to v;

if the primality test fails then {

v := v+2;

GOTO Step 2a; }

m++;

}

3. p:= v (resp., q := v)

Assumptions

- Power analysis allows the attacker
  - to identify for each prime candidate v after which trial division the while-loop has terminated
  - whether a Miller-Rabin test has been applied.

NOTE: If all trial divisions need approximately the same run-time it suffices to identify the beginning of the while-loop 2b) or the incrementation step v := v+2.

Remark

- We further assume that
  - the RNG is strong
  - the trial division itself and the Miller-Rabin tests are perfectly protected against side-channel attacks

- Otherwise, even stronger attacks may exist.

Basic attack (I)

- Notation: $v_0 = v$, $v_1 = v_0 + 2, \ldots, v_m = v_0 + 2m := p$
- Basic observation:
  - For v_j loop 2b) terminates after trial division by r
  - $\Rightarrow v_j \equiv 0 \pmod r$
  - $\Rightarrow p = v_m = v_j + 2(m-j) \equiv 2(m-j) \pmod r$

Federal Office for Information Security

Basic attack (II)

- Generation of p:
  $S_p := \{2\} \cup \{ r \in T \mid$ for at least v_j the algorithm terminated after the division by r $\}$

- The CRT gives

$$a_p \equiv p \pmod{s_p} \qquad \text{with } s_p := \prod_{r \in S_p} r \,,$$

and finally

$$a_q \equiv q \equiv a_p^{-1} n \pmod{s_p}$$

Federal Office
for Information Security

Basic attack (III)

- Analogously (observing the generation of q)

$$b_q \equiv q \pmod{s_q} \qquad \text{for } s_q := \prod_{r \in S_q} r$$

and

$$b_p \equiv p \equiv b_q^{-1}\, n \pmod{s_q}$$

- Finally, from (a_p, b_p) and (a_q, b_q) the attacker computes

$$c_p \equiv p \pmod{s},\ c_q \equiv q \pmod{s} \quad \text{with } s := \mathrm{lcm}(s_p, s_q)$$

Basic attack (IV)

- $p = sx_p + c_p$, $q = sy_q + c_q$ for unknown integers x_p, y_q

- The pair (x_p, y_q) is a zero of the irreducible bivariate integer polynomial $f: Z \times Z \rightarrow Z$,
  $f(x,y) := sxy + c_p y + c_q x - t$ with $t := (n - c_p c_q) / s$

- If $\log_2(s) > k/2$ the LLL algorithm finds the pair (x_p, y_p) in time polynomial in k (k = bit length of p and q).

Empirical results

- Simulations of the attack with Magma (≅ perfect measurements)
- k = 512; LLL requires at least $\log_2(s) > 256$
- Trial bases: $T_1 = \{3,5,\ldots,251\}$, /* odd primes < 2^8 */
  $T_2 = \{3,5,\ldots,281\}$, $T_3 = \{3,5,\ldots,349\}$

Success Probabilities (basic attack)

	T_1	T_2	T_3
$\mathrm{Prob}(\log_2(s) > 256)$	0.118	0.188	0.283
$\mathrm{Prob}(\log_2(s) > 277)$	0.055	0.120	0.208

Remark

- If $\log_2(s) < k/2$ the LLL-algorithm will not find the zero (x_p, y_p).
- One may guess the remainder p (mod r_i') for some further primes $r_1', .., r_m'$ so that $s' := s \cdot r_1' \cdot \cdot \cdot r_m'$ is sufficiently large.
- <u>Drawback:</u> In the worst case the LLL algorithm has to be applied to all $r_1' \cdot \cdot \cdot r_m'$ admissible candidates (c_p', c_q') for (p (mod s'), q (mod s'))

Refinements of the attack

By exploiting further side-channel information from

- the trial divisions
- the extended Euclidean algorithm (computation of d (mod (p-1)) and d (mod (q-1))

many candidates for c_p' can be excluded.

Remark: For k=512 this provides about 10-15 bits additional information.

Experimental results (I)

- Sample implementation on an ATMEL ATmega microcontroller

```
rnd2r( );               /* generates a random number */
testdiv512 (v,3);       /* trial division by 3 */
testdiv512 (v,5);
testdiv512 (v,7);
incrnd (v);             /* increments v by 2 */
testdiv512 (v,3);
incrnd (v);
testdiv512 (v,3);
testdiv512 (v,5);
```

Experimental results (II)

- Notation: $x_1, x_2, \ldots, x_N$: power consumption during the particular clock cycles
- Goal: Find a characteristic sample that identifies a trial division or an incrementation step

$$x_1, x_2, \ldots, x_{t-1}, \boxed{x_t, x_{t+1}, \ldots, x_{t+M-1}}, x_{t+M}, \ldots, x_N$$

$$y_1, y_2, \ldots, y_M \quad \text{characteristic sample}$$

Experimental results (III)

- The *similarity function*

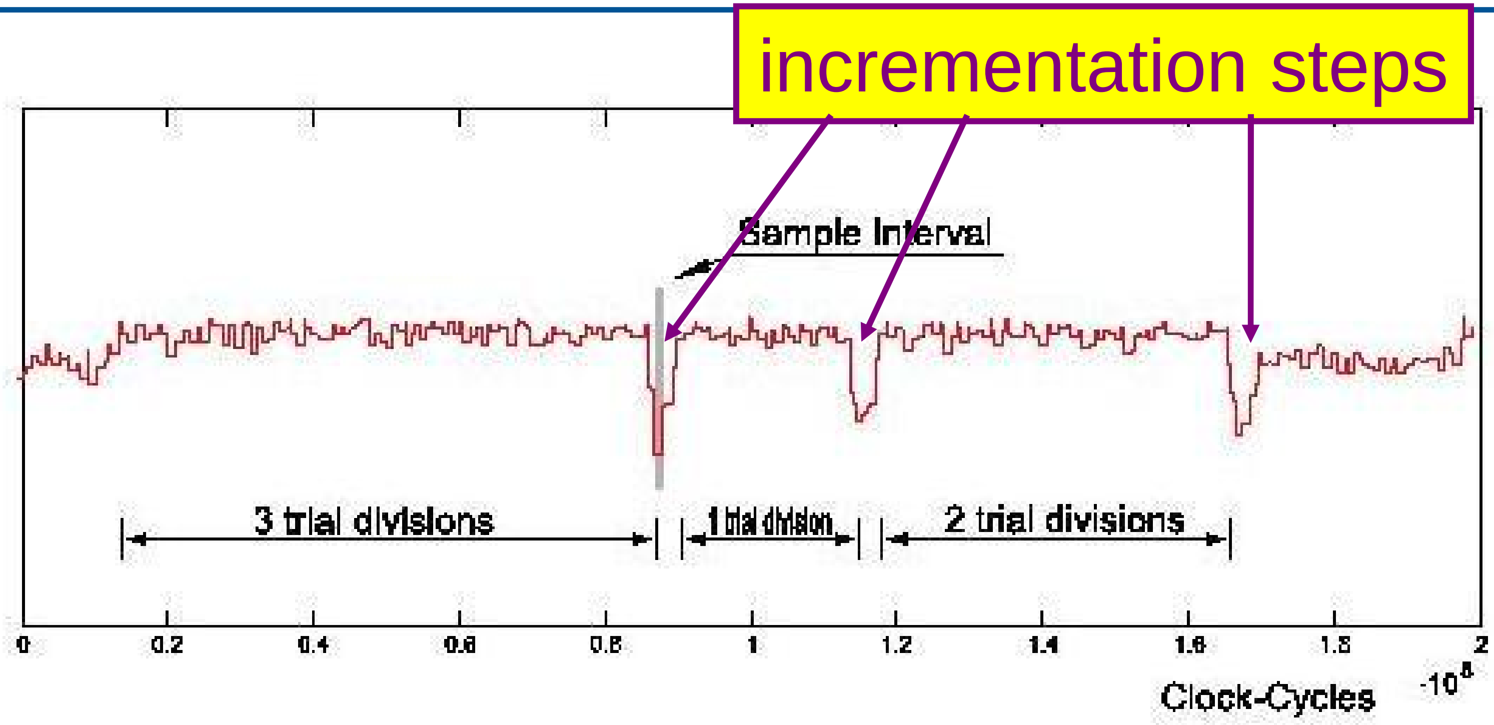
$$a_j = \frac{1}{M}\sum_{i=1}^{M} | x_{i+j} - y_i | \qquad \text{for } j \in \{1,\ldots, N\text{-}M\}$$

compares $(y_1, \ldots, y_M)$ with the power consumption subsequence $(x_{j+1}, \ldots, x_{j+M})$ for all shift parameters j.

To compensate random local effects we finally applied

$b_j := \min \{a_j,\ldots,a_{j+F-1}\}$ /*minimum over a 'window'*/

Experimental results (IV)



- low peaks: large similarity, high peaks: large dissimilarity
- sample sequence within the first incrementation step
  low peaks = positions of incrementation steps

Possible countermeasure

- regular refreshment of the prime candidates v_j by updating some bytes (e.g., XORing 8 bytes of every 10^{th} candidate v_j with random bytes)

Final remarks

- We have demonstrated the power of a side-channel attack on a straight-forward prime generation algorithm.
- Simulations yielded success probabilities of 10 – 15 %, and practical experiments verified that the above-mentioned assumptions are indeed realistic.
- Moreover, this paper shall motivate the community to devote more attention to the key generation step.

Federal Office
for Information Security

Contact

Federal Office for Information Security (BSI)

Werner Schindler
Godesberger Allee 185-189
53175 Bonn

Tel: +49 (0)22899 - 9582-5652
Fax: +49 (0)22899 - 10-9582-5652

Werner.Schindler@bsi.bund.de
www.bsi.bund.de
www.bsi-fuer-buerger.de