SPA-based attack against the modular reduction within a partially secured RSA-CRT implementation

Helmut Kahl

R&D Cryptology
Giesecke & Devrient GmbH
Prinzregentenstr. 159, D-81677 München
helmut.kahl@de.gi-de.com
http://www.gi-de.com

August 11, 2004

Abstract. This note describes an SPA-based side channel attack against a CRT implementation of an RSA function. In contrast with Novak's attack [8], it concentrates on the initial modular reduction. With the help of lattice reduction it applies even to implementations which use a common randomising technique to ensure resistance against certain side channel attacks.

Keywords. RSA, CRT, smartcard, implementation, SPA, modular reduction, lattice reduction

Introduction

In an RSA function using the Chinese Remainder Transformation (CRT), the input text x is reduced modulo the secret prime factors p and q of the RSA modulus $p \cdot q$, before its powers modulo p and q are taken. To prevent Differential Power Analysis like in [2] and Differential Fault Analysis like in [3], in a secured RSA-CRT implementation on smartcards, p and q are multiplied by a secret random integer z. Then $x \bmod (p \cdot z)$ is calculated as the base of the first exponentiation, and likewise with q for the second exponentiation. Let us assume a side channel leaks the integral quotient $s = \lfloor x/(p \cdot z) \rfloor$ during the reduction of x modulo $p \cdot z$. This assumption is conceivable in case of Simple Power Analysis (SPA) of a bitwise 'pencil-and-paper' implementation of the integer division (cf. ch. 3 of [5]), which is used for this modular reduction. Then we can retrieve the secret primes from $n = p \cdot q$ and from s provided that

1. x has approximately the same size as n, i.e. x and n have similar bit lengths,
2. an approximation $\tilde{x}$ of x is known ($\tilde{x} \approx x$),
3. z is not unusually big.

Later we concentrate on a concrete bound of z. Now we illustrate the idea of the attack. Firstly let us assume that z equals one. Then s has approximately the same size as q by assumption 1. Therefore, $u = \lfloor \tilde{x} / s \rfloor$ is an approximation of p by assumption 2. So, in this case, we can find p by checking the candidates in the neighbourhood of u for dividing n. Now let us consider the situation of an unknown number z. Then, under the assumptions 1 to 3, we have

$$\tilde{x} \approx x \approx s \cdot p \cdot z \text{, and therefore } u = \lfloor \tilde{x} / s \rfloor \text{ is an approximation of } p \cdot z .$$

Hence n and u have an "approximate integer common divisor" p in the sense of [4]. By a lattice reduction technique we can retrieve p from n and u as will be illustrated in the following.

Description of Algorithm

In [4] Howgrave-Graham suggests a method with lattices of rather high rank and with rather big entries for searching common divisors. In this section we present a lattice of rank three which is tailor-made for solving the computational problem of finding the prime p in the above context (notation like before):

Input: $n = p \cdot q$, $\tilde{x} \approx x$, $s = \lfloor x/(p \cdot z) \rfloor$ for an unknown number z

Output: p, q

1. Calculate $u = \lfloor \tilde{x} / s \rfloor$ (By assumptions 1 to 3 this implies $u + r \equiv 0 \pmod p$ for a small integer r.)
2. Find a short vector (v_0, v_1, v_2) of the **Z**-lattice with base $(n, 0, 0)$, $(u, 1, 0)$, $(u^2, 2 \cdot u, 1)$. (If regarded as polynomials, like in [4], these base vectors have r as a root modulo p, hence also every vector of the lattice.)
3. Calculate an integer root r of the quadratic polynomial $v_0 + v_1 \cdot r + v_2 \cdot r^2$. (Because of the small coefficients the root exists in **Z**, and not only modulo p.)

4. Output $\gcd(u+r, n)$ and $n/\gcd(u+r, n)$.

For finding a short lattice vector in step 2 you can use the LLL algorithm of [5] or variants of it.

Bound

In step 2 the LLL algorithm produces a vector of norm $\leq 2^{3/2} \cdot n^{1/3}$. So the deviation $|r|$ of the approximation u of $p \cdot z$ should be significantly smaller than $n^{1/3}$. By assumption 1 and 2 this is the case when $p \cdot z$ is significantly smaller than $n^{2/3}$. To see this define $m = p \cdot z$ and $t = x \pmod m$ so that $0 \leq t < m$. Then

$$u \approx x/s = x \cdot m/(x-t)\text{, and therefore } u - m \approx \frac{x \cdot m}{x-t} - \frac{x \cdot m}{x} = \frac{t \cdot m}{x-t} < \frac{m^2}{x-m} = \frac{m^2}{x} + \frac{m^3}{x \cdot (x-m)}.$$

But with x of size n and $m << n^{2/3}$ the last summand can be neglected and the first summand is significantly smaller than $n^{1/3}$. This asserts the upper bound $n^{2/3}$ of $p \cdot z$. Hence, for RSA primes p and q of about the same length, the security parameter z must be significantly smaller than $n^{1/6}$ to ensure correctness of the algorithm. In practice the length of z is even not more than 1/16 of the length of the RSA modulus n.

Experimental Results

We implemented the algorithm with Maple 6.0 on a Pentium III 700 MHz machine and tested RSA moduli of different bit lengths up to 2048. For the bit lengths of n, $|x - \tilde{x}|$, z in the ratio 16 : 2 : 1 we applied the program several thousands of times to random input numbers x and z, each time with success and each time with a few seconds of computation only.

Countermeasures

The two following countermeasures obviously prevent from the presented attack:

- SPA protected implementation of the modular reduction,
- blinding of x by addition of a random multiple of the RSA modulus n.

Algorithm D in chapter 4.3.1 of [6] can serve as a draft for implementation of an SPA protected division. It operates with words of any length given by the respective platform.

For the latter countermeasure longer operands in the RSA implementation must be taken into account.

Conclusion

We have shown a very efficient SPA-based attack against an RSA-CRT implementation which uses random factors of the prime moduli. The presented algorithm overcomes the protection with the random factors by using a lattice reduction technique. Our attack applies to RSA signing and verifying as well as to RSA decryption, since these operations do not keep secret the exponentiation base x. Even probabilistic signature schemes like that in [1] provide no inherent security against our attack. In these schemes random 'salt' is used in the signing algorithm for preparation of x. But it can not be regarded as secret, since it is revealed by the verifying algorithm.

References

1. M. Bellare and P. Rogaway, The exact security of digital signatures – How to sign with RSA and Rabin. Proceedings of Eurocrypt '96, LNCS **1070** (1996) Springer, pp. 399-416
2. B. den Boer, K. Lemke and G. Wicke, A DPA attack against the modular reduction within a CRT implementation of RSA. CHES 2002, LNCS **2523** (2003) Springer, pp. 228-243
3. D. Boneh, R. A. DeMillo and R. J. Lipton, On the importance of checking cryptographic protocols for faults. Proceedings of Eurocrypt '97, LNCS **1233** (1997) Springer, pp. 37-51
4. N. Howgrave-Graham, Approximate integer common divisors. CaLC 2001, LNCS **2146** (2001) Springer, pp. 51-66
5. M. Joye and Karine Villegas, A protected division algorithm. Fifth Smart Card Research and Advanced Application Conference (CARDIS'02), Usenix Association (2002) P. Honeyman, pp. 69-74
6. D. E. Knuth, The art of computer programming, vol. 2: seminumerical algorithms, 3rd edition (1998) Addison-Wesley
7. A.K. Lenstra, H.W. Lenstra and L. Lovasz, Factoring polynomials with rational coefficients. Mathematische Annalen **261** (1982) pp. 513-534
8. R. Novak, SPA-based adaptive chosen-ciphertext attack on RSA implementation. PKC 2002, LNCS **2274** (2002) Springer, pp. 252-262