Bellcore attack in practice

Andrey Sidorenko, Joachim van den Berg, Remko Foekema,
Michiel Grashuis, Jaap de Vos

Brightsight BV
Delftechpark 1, 2628 XJ Delft, the Netherlands
http://www.brightsight.com

Abstract. In this paper we analyze practical aspects of the differential fault attack on RSA published by Boneh, Demillo and Lipton from Bellcore. We focus on the CRT variant, which requires only one faulty signature to be entirely broken provided that no DFA countermeasures are in use. Usually the easiest approach for the attacker is to introduce a fault in one of the two RSA-CRT exponentiations. These are time-consuming and often clearly visible in the power profiles. However, protection of the exponentiations against faults does not always circumvent the Bellcore attack. Our goal is to investigate and classify other possible targets of the attack.

1. Introduction

RSA algorithm [RSA78] has been introduced more than thirty years ago. Since that moment the algorithm has become very popular and as a result a lot of attacks on RSA have been published. Among the most powerful are differential fault attacks based on the assumption that it is possible to inject hardware faults when the victim device performs RSA. The pioneer differential fault attack on RSA is the one published by Boneh, Demillo and Lipton from Bellcore [BDL01]. It is often referred to as the Bellcore attack.

This paper is devoted to the analysis of practical aspects of the Bellcore attack. We investigate steps of the RSA implementation that can serve as a target for the attack.

2. Bellcore attack

The attack is applicable both for classical RSA that involves one modular exponentiation and for RSA-CRT. In the first case the attack requires several faulty signatures while in the second case only one faulty signature may suffice. We consider only the CRT variant.

RSA-CRT transforms message m into signature s using private key p, q, d_p, d_q as follows:

$$\begin{aligned} s_p &= (\, m_p \,)^{d_p} \bmod p, \\ s_q &= (\, m_q \,)^{d_q} \bmod q, \\ s &= (\, (\, (s_q - s_p) \cdot p_{inv}) \bmod q) \cdot p + s_p, \end{aligned}$$

where $m_p = m \bmod p$, $m_q = m \bmod q$, $p_{inv} = p^{-1} \bmod q$. For a detailed overview of RSA-CRT the reader is referred e.g. to [BDL01].

Suppose either s_p or s_q is computed with a fault. Assume that the resulting faulty signature s' together with the correct signature s are known to the attacker. Then he can retrieve the private key by computing

$$\gcd(s - s', N)$$

for the publicly known RSA modulus N. Alternatively, the attacker can recover the private key from s' and m by computing

$$\gcd(m - (\, (\, s')^e \bmod N\,), N),$$

where e is the public exponent. Boneh et al. [BDL01] point out that the latter variant is suggested by A. K. Lenstra.

3. Possible targets of the attack

3.1 CRT exponentiations

A natural way to put the Bellcore attack in practice is to perturb one of the RSA-CRT exponentiations. These operations usually take relatively long time giving the attacker a wide window of opportunity. In the case of smart cards an RSA exponentiation typically takes several milliseconds. Furthermore, normally the exponentiations are clearly distinguishable in the power traces, which helps the attacker to choose the right moment e.g. for light manipulation or voltage manipulation.

However in order to avoid the Bellcore attack it is not always sufficient to protect only the exponentiations. Nearly all the other steps of RSA-CRT should be protected as well. These steps can be divided into two classes: those that are performed before the CRT-exponentiations and afterwards. Some important examples of each of these two classes are discussed below.

3.2 Steps performed before the exponentiations

Key loading

Even though key loading usually takes significantly less time than an exponentiation, this action is often visible in the power profiles. If a fault is injected while loading (p, d_p) or (q, d_q) but not in both of the pairs the attacker is likely to get a faulty signature s' suitable for retrieving the private key.

Note that replacing p by a faulty $p' \neq p$ during key loading leads to the signature s' such that

$$s' \neq s \bmod p \text{ and}$$
$$s' \neq s \bmod q,$$

which is not beneficial for the attacker, with overwhelming probability. On the other hand, inducing a fault in q will mean that $s' = A \cdot p + s_p$, where $A \neq (((s_q - s_p) \cdot p_{inv}) \bmod q)$, and thus

$$s' = s \bmod p \text{ and}$$
$$s' \neq s \bmod q$$

so computing the greatest common divisor $\gcd(s - s', N)$ will reveal the private key. The reason for this behaviour is that p and q are used in the CRT recombination in an asymmetric way, see Section 2.

Reformatting

The implementation may require changing the format of the key after loading. For instance, the format may be changed from big to little endian. The reformatting should be secure against perturbation attacks. Similarly to the case of key loading described above, here the perturbation of p and perturbation of q have different effects.

Modular reduction

As a preparation step for the exponentiations message m is reduced modulo p and modulo q to compute m_p and m_q, respectively. Both of the modular reductions should be protected since any perturbation of one of them may result in a successful attack.

Transition to Montgomery representation

If Montgomery multiplication (see e.g. [MOV97] for a thorough overview) is used care should be taken when transforming the data from one representation to the other. In the beginning of the Montgomery exponentiation m_p and m_q are multiplied by the Montgomery constant R modulo p and modulo q. Both of the modular multiplications are potential targets of the Bellcore attack.

On the other hand, if R is perturbed before transforming m_p and m_q into Montgomery representation the Bellcore attack will not work since we will end up in the situation when

$$s' \neq s \bmod p \text{ and}$$
$$s' \neq s \bmod q.$$

Data loading

Sometimes it is required to load m_p and m_q into the memory of an arithmetic coprocessor. The memory copy function should be implemented in a secure way.

Blinding

As a countermeasure against side channel attacks (SCA) the developer may decide to blind m_p, m_q, p, q, d_p, d_q. A well-known example is multiplicative message blinding:

$$m_p \rightarrow \mu_p = m_p \cdot r_p \bmod p,$$
$$m_q \rightarrow \mu_q = m_q \cdot r_q \bmod q,$$

where numbers r_p and r_q are chosen at random for every signature. This is an efficient countermeasure against differential power analysis aimed at modular exponentiation.

Remarkably, in this situation SCA-countermeasures may ease the fault attack. A fault induced in any of the blinding operations may imply the retrieval of the private key.

It is worth noticing that this is not the only situation of this sort. SCA-countermeasures require extra effort; the additional steps immediately become potential targets of the perturbation attacks. To give another example, Yen en Joye [YJ00] observed that the square-and-always-multiply method provides an opportunity to distinguish dummy multiplications from real ones by inducing faults at the moments when these exponentiations are performed.

3.3 Steps performed after the exponentiations

Steps similar to those performed before the exponentiations

These are the following. Whenever Montgomery exponentiation is used the final result s_p and s_q, will be transformed back from Montgomery representation into the regular one. If a coprocessor is used it might be necessary to transfer s_p and s_q from its memory to the other location. A developer may decide to unblind s_p and s_q as soon as they are computed. All these operations should be carefully protected against perturbation attacks.

CRT recombination

The last step of the RSA-CRT algorithm, namely, the CRT recombination is also a potential target of the Bellcore attack. As opposed to some other targets this one is quite difficult since the fault has to be induced very precisely such that, for instance, $(((s_q - s_p) \cdot q_{inv}) \bmod q)$ is manipulated while s_p remains unchanged.

4. Conclusion

The Bellcore attack on RSA-CRT is one of the most prominent attacks on RSA known so far. Whenever RSA-CRT is implemented in secure embedded software, this attack has to be circumvented by appropriate countermeasures. Most of the steps of RSA-CRT have a property that their perturbation may result in a signature s' such that

$$s' = s \bmod p$$
$$\text{while } s' \neq s \bmod q$$

or vice versa. It means that the protection mechanisms should cover not only the CRT exponentiations but nearly all the other steps of the algorithm that influence the computation of s_p, s_q or the CRT recombination.

A powerful countermeasure against the Bellcore attack is to raise the signature to the power e and to compare the result with the input message m. If the signature is faulty the comparison will detect it. A possible obstacle is that in practice e is not always available to the signer function.

Another well-known countermeasure against the Bellcore attack is to perform the signature computation twice and to compare the two results. Remarkably, this countermeasure does not necessarily help to prevent the attack. For instance, it might happen that both computations will be performed using the same erroneous private key and thus the two results will match.

References

[BDL01] D. Boneh, R. A. DeMillo, R. Lipton. On the Importance of Eliminating Errors in Cryptographic Computations. In Journal of Cryptology 14(2), pages 101–120, 2001.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, volume 21(2): pages 120–126, 1978.

[MOV97] A. Menezes, P. van Oorschot and S. Vanstone. Handbook of applied cryptography. CRC Press, 1997.

[YJ00] S.-M. Yen, M. Joye. Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis. IEEE Trans. Computers 49(9): 967–970 (2000).