Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures

C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert

Infineon Technologies
Security & ChipCard ICs
D-81609 Munich
Germany
{christian.aumueller, peter.bier, wieland.fischer, peter.hofreiter, jean-pierre.seifert}@infineon.com

Abstract. This article describes concrete results and practically validated countermeasures concerning differential fault attacks on RSA using the CRT. We investigate smartcards with an RSA coprocessor where any hardware countermeasures to defeat fault attacks have been switched off. This scenario was chosen in order to analyze the reliability of software countermeasures.
We start by describing our laboratory setting for the attacks. Hereafter, we describe the experiments and results of a straightforward implementation of a well-known countermeasure. This implementation turned out to be not sufficient. With the data obtained by these experiments we developed a practical error model. This enabled us to specify enhanced software countermeasures for which we were not able to produce any successful attacks on the investigated chips.
Nevertheless, we are convinced that only sophisticated hardware countermeasures (sensors, filters, etc.) in combination with software countermeasures will be able to provide security.

Keywords: Bellcore attack, Chinese Remainder Theorem, Fault attacks, Hardware security, RSA, Spike attacks, Software countermeasures, Transient fault model.

1 Introduction

This paper shows and proves that fault attacks on RSA with the CRT (also known as Bellcore attacks) due to [BDL] are feasible. They are indeed devastating if there are neither hardware mechanisms (sensors, filters, etc.) nor any appropriate software countermeasures implemented in the underlying smartcard ICs. However, this does not imply that modern high-security smartcard ICs are vulnerable to this kind of attacks. Instead, it shows that fault tolerance and especially sophisticated hardware countermeasures are essential for the design of secure hardware. Moreover, we stress that it is very difficult in the field to switch off these sophisticated hardware countermeasures. This has been done

B.S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, pp. 260–275, 2003.

exceptionally for our study concerning software countermeasures against the Bellcore attack.

In order to provide better security for data protection under strong encryption more and more implementations on tamper-proof devices (e.g., smartcard ICs) are proposed. The main reason is that smartcard ICs provide high reliability and security with more memory capacity and better performance characteristics than conventional magnetic stripe cards. With special characteristics of computational ability a large variety of cryptographic applications benefit from smartcard ICs. This attracted a huge amount of research on physical attacks against smartcards in 1996 due to [Koch], [BDL] and again 1999 by [KJJ], followed by [GMO,SQ]. However, most research so far focused on Timing or Power Analysis attacks. This is surprising as the frauds with smartcards by inducing faults are reality, cf., [A,AK1,AK2], whereas no frauds via Timing or Power Analysis attacks have been reported so far. Moreover, research on fault-based cryptanalysis is not very active compared to the other side-channel attacks. Furthermore, no practical investigation of the Bellcore attack is presently known. Indeed, this topic will be publicly addressed within this paper for the first time. It answers a question of Kaliski and Robshaw [KR] *of how practical these attacks might be*, answered definitely here *by physicists, designers and manufactures of secure hardware.*

The present paper is organized as follows: Section 2 briefly repeats RSA using the CRT and its fault-based cryptanalysis according to [BDL,JLQ]; it also includes and discusses the advantages and limitations of so far publicly known software countermeasures to defeat fault attacks on RSA in CRT mode. Section 3 firstly explains so-called spike attacks and their realization on smartcard ICs, their complexity from an attacker's point of view and reveals an appropriate test equipment to implement fault attacks. Secondly, we will present the resulting errors on unprotected hardware and software for RSA in CRT mode. This demonstrates the insufficiency of a straightforward implementation of a well-known countermeasure due to [Sh]. Within section 4 we basically investigate enhanced software countermeasures derived from our practical observations and our proposed model to counteract fault attacks on RSA. Eventually, section 5 adds some practical conclusions concerning software countermeasures to prevent Bellcore attacks.

2 Preliminaries

2.1 The RSA System

Let $N = p \cdot q$ be the product of two large primes of similar length. To sign a message $m \in \mathbb{Z}_N$ using RSA one computes $S := m^d \bmod N$, where d is the private exponent satisfying $e \cdot d \equiv 1 \bmod (p-1)(q-1)$ for the public exponent e. The computationally expensive part of signing is the modular exponentiation. For better efficiency most implementations exponentiate as follows: using repeated square and multiply they first compute $S_p := m^d \bmod p$ and hereafter $S_q := m^d \bmod q$. Then they construct the signature $S = m^d \bmod N$ using the

CRT. This last step takes negligible time compared to the two exponentiations. It is done efficiently by computing

$$S = S_q + \big((S_p - S_q) * (q^{-1} \bmod p) \bmod p\big) * q, \tag{1}$$

using Garner's algorithm, cf. [Kn].

The exponentiation using the CRT is much faster than the full exponentiation. To see this, observe that $S_p = m^d \bmod p = m^{d \bmod (p-1)} \bmod p$. Usually, d is of order N, while $d \bmod (p-1)$ is of order p. Consequently, computing S_p requires half as many multiplications as computing S directly. In addition, intermediate values during the computation of S_p are only half as big — they are in the range $[1, \ldots, p]$, rather than $[1, \ldots, N]$. Clearly, the same arguments are valid for the computation of S_q. When quadratic time complexity is used, multiplying two numbers in $\mathbb{Z}_p$ takes a quarter of the time as multiplying elements in $\mathbb{Z}_N$. Hence, computing S_p takes an eighth of the time of computing S directly. Thus, computing S_p and S_q this way takes a quarter of the time of computing S directly. Thus, CRT exponentiation is four times faster than direct exponentiation. This is the reason for using the CRT for RSA signature generation, cf. [CQ,MvOV].

2.2 The Fault-Based Cryptanalysis of RSA Using CRT

We briefly recall the fault-based cryptanalysis of RSA with the CRT due to [BDL,JLQ]. Assume that during the computation of an RSA signature for a message m a random error occurs during the computation of S_p. This yields a faulty signature part S'_p, whereas the computation of S_q is done correctly. The combination of S'_p and S_q via (1) will yield an incorrect signature S'. For S' it holds that $S - S' \neq 0$ but $S - S' \equiv 0 \bmod q$. Therefore, one obtains the factorization of N by computing

$$\gcd\left((m - (S')^e) \bmod N, N\right) = q.$$

2.3 Simple Software Countermeasure to Defeat the Fault Attack

Some simple ad-hoc countermeasures have been already suggested within [BDL, KR]. One approach is to perform calculations twice and the other approach suggests to verify the correctness of the signature by comparing the inverse result with the input. The first approach is very time-consuming and it cannot always provide a satisfactory solution because a permanent error may be undetectable by computing the function more than once. The second approach is to verify the correctness by comparing the inverse result with the input m. Generally, this is not a satisfactory solution since the parameter e could be a large integer and this checking procedure becomes time-consuming. Additionally, for a real life software implementation the programmer cannot rely on the fact that e is known and a small number. On the other hand, this countermeasure seems to be the safest.

An interesting countermeasure is the introduction of randomness into the RSA signature process. Here, RSA is applied to $F(m, r)$ where F is some formatting function and r is a random string which ensures that the user never signs the same message twice and the attacker does not know the signed message, cf. [BDL,BR,KR]. Other countermeasures are mentioned in [Ro].

2.4 Shamir's Software Countermeasure

Shamir's idea, cf. [Sh], is to select a random integer t and to do the following computations

$$S_{pt} := m^d \bmod p * t,$$
$$S_{qt} := m^d \bmod q * t.$$

In the case of $S_{pt} = S_{qt} \bmod t$ the computation is defined to be error free and S is computed according to the CRT recombination equation (1).

One drawback in Shamir's method, as pointed out in [JPY], is the following: Within the CRT mode of real RSA applications the value d is not known, only the values $d_p = d \bmod (p-1)$ and $d_q = d \bmod (q-1)$ are known. Although d can be efficiently computed from d_p and d_q only, as described in [FS], it will limit the acceptance of Shamir's method. Moreover, his check will be shown to be insufficient anyway. But, our enhanced software countermeasures will resolve the above critical points of his method.

2.5 General Remarks on Methods to Overcome Fault Attacks

Only very recently the field of research on fault attacks countermeasures has been emerged. For instance a series of papers [YJ,YKLM1,YKLM2,JQYY] assume that the attacker has a very precise knowledge about the implementation details and especially an absolute accurate control of the timing of his fault induction. Under this strong assumption the private exponent d can be reconstructed by abusing the implemented correctness check as an oracle for the bits of d. However, all the described fault attacks can easily be prevented by various randomization techniques for the RSA algorithm. In a side-channel secure RSA signature implementation such techniques are present.

Moreover, [YKLM1] proposed the following very interesting countermeasure: Their key idea is to influence the computation of S_q or the overall computation of S when an error occurred during the computation of S_p, or vice versa. The cryptanalysis given in section 2 shows that a successful fault attack is not possible anymore. Unfortunately it was recently shown by [BMS] that their proposal for a so-called infective RSA CRT computation is not secure.

3 Physical Fault Attacks Realization

First of all, we would like to stress again that modern high-end cryptographic devices, e.g., smartcards, are usually protected by means of various and numerous

sophisticated hardware mechanisms to detect any intrusion attempt into their system behavior, cf. [Ma,NR]. This is due to the fact that hardware manufacturers of cryptographic devices such as smartcard ICs have been aware of the importance of protecting against intrusions by, e.g., external voltage variations, external clock variations, etc. for a long time. However, it should be clear that the design of such mechanisms is a very difficult engineering task. Such mechanisms should be able to tolerate slight natural deviations from the standard values of the electrical parameter to be safeguarded. This is necessary to ensure a proper functionality of the underlying device within the specified range, as for example described in [ISO]. On the other hand they also have to detect very fast and unnatural low deviations from the specified standard range. This condition is necessary to detect any attack attempt by modifying the electrical execution conditions to alter a computation's result. For example, the standard specification [ISO] allows for the smartcard IC's contact V_{CC} under normal operating conditions a voltage supply between $4,5$V and $5,5$V.

Although there are lots of possibilities to introduce an error during the cryptographic operation of an unprotected smartcard hardware, we will only explain in detail the so-called spike attacks. The reason is that spike attacks are non invasive attacks. Thus, they require no physical opening and no chemical preparation of the smartcard IC. For further information on various methods how to enforce erroneous computations of chips we refer to [A,AK1,AK2,Gu1,Gu2, Koca,Ma].

3.1 Spikes

A smartcard of voltage class type A should be able to tolerate on the contact V_{CC} a supply voltage between $4,5$V and $5,5$V, where the standard voltage is specified at 5V. Within this range the smartcard will be able to work properly. However, a deviation of the external power supply of much more than the specified 10% tolerance could cause problems with the smartcard IC. Indeed, it could then lead to a wrong computation result, provided that the smartcard IC is still able to finish its computation completely. But most often this is not possible, as the spike causes too much trouble to the CPU of the smartcard IC. Although a spike with the explanation above seems very simple, a specific type of a power spike is determined by nine parameters. Using picture 1 we will explain them:

1. Initial value of the power supply V_2.
2. Starting point t_1 of the spike.
3. Rise time $t_2 - t_1$ of the spike.
4. Shape of the rising transition.
5. Height $V_3 - V_1$ of the power spike.
6. Length of the power spike $t_3 - t_2$.
7. Falling time $t_4 - t_3$ of the spike.
8. Shape of the falling transition.
9. Final value V_1 of the power supply.

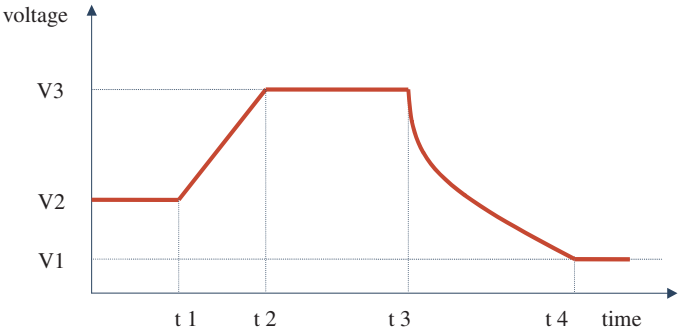

Fig. 1. Spike-parameters defining the shape of a specific spike.

This indicates the huge range of different parameters which must be scanned for penetration attacks against cryptographic devices. On the other hand, it also reveals the strong demands on the corresponding sensor and filter mechanisms. From the former discussion of spike attacks, one can envision the difficulties an attacker is confronted with, when he wants to overcome all the activated hardware countermeasures within modern high-security smartcard ICs.

3.2 Laboratory Setting

In order to systematically investigate the effects of spikes and especially our proposed countermeasures, we basically used the following spike enforcing hardware set-up, which is shown in figure 2.

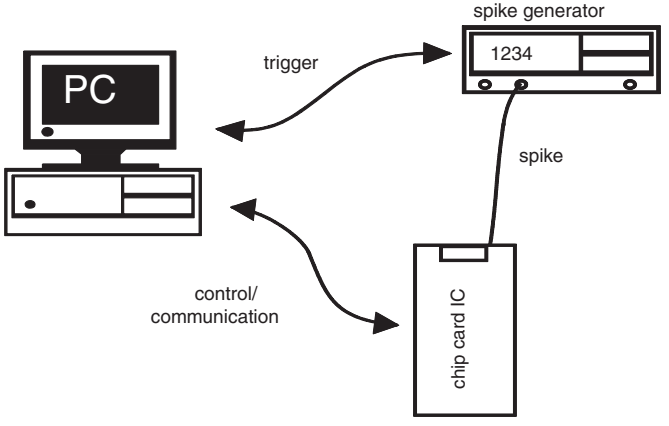

Fig. 2. Diagram of our test equipment.

With such a test set-up it is indeed possible to enforce a spike with a very high accuracy. This is necessary, if the spike shall just only enforce a tiny ran-

dom computation fault rather than a complete destruction of the smartcard's computation, which would make the smartcard's computation result unusable for a successful attack. Through the coupling of the control and communication of the smartcard with a PC, which is running a dedicated test-software, it is possible to observe and analyze the smartcard's reaction with respect to the applied spike-form as discussed above, e.g., answering with a correct/wrong answer sequence. Furthermore, the PC is responsible for the stimuli, timing and controlling of the above spike parameters. Coupled with an interface card, the spike generator is triggered by the PC which provides the time and voltage information for the specific spike to be applied to the card. The spike generator is directly connected to the power supply V_{CC} of the smartcard and provides its IC with the necessary operating voltage including the voltage drop of the spike. By means of the synchronization of the PC, the spike generator and the chipcard itself a very high attack reproducibility of more than 90% can be achieved.

Now, one has to find parameters for such a spike which enables a tiny random computation fault, but leaves the main computation untouched.

3.3 Results on Unprotected Hardware and Software

We will now discuss our results of successfully applied spike-attacks on unprotected smartcards, i.e., ICs where any hardware countermeasures against fault attacks have been switched off. Moreover, we have also switched off any (hardware and software) countermeasures against other classical side-channel attacks, like Timing Analysis [Koch], Power Analysis [KJJ], Electromagnetic Analysis [SQ,GMO], etc.

However, to introduce a spike at the right position of the RSA with the CRT, one should investigate the power profile of the critical computation first. Such a power profile of our investigated smartcard equipped with an RSA coprocessor is shown in figure 3. Let us explain this power profile a little bit more: The upper line represents the profile of the smartcard's I/O behavior. The first I/O activity is the start impulse for the smartcard and the second peak is the answer sequence given by the smartcard. Between these two peaks the smartcard is computing a 2048-bit RSA signature using the CRT. This is shown in the lower line where the main power profile of the smartcard is depicted.

The RSA-CRT computation starts at the time block 1.5 and ends at the time block 9.2. In the figure the blocks are numbered from 0 to 9. This is shown by the fact that the power consumption increases — due to the coprocessors activity. One immediately recognizes the two different exponentiations as they are the main power consumers.

In our case the first exponentiation lies in the time frame 1.6 to 5.1, and the second exponentiation lies in the time frame 5.3 to 8.8. Before the first exponentiation one recognizes the loading of the data into the crypto coprocessor for the first exponentiation, after the first exponentiation the corresponding correctness checks and as well the loading of the data into the crypto coprocessor and for the second exponentiation and after the second exponentiation again the correctness checks of the second exponentiation. Finally, one sees the CRT combination of

the two partial exponentiations followed eventually by an additional correctness check for the CRT combination.

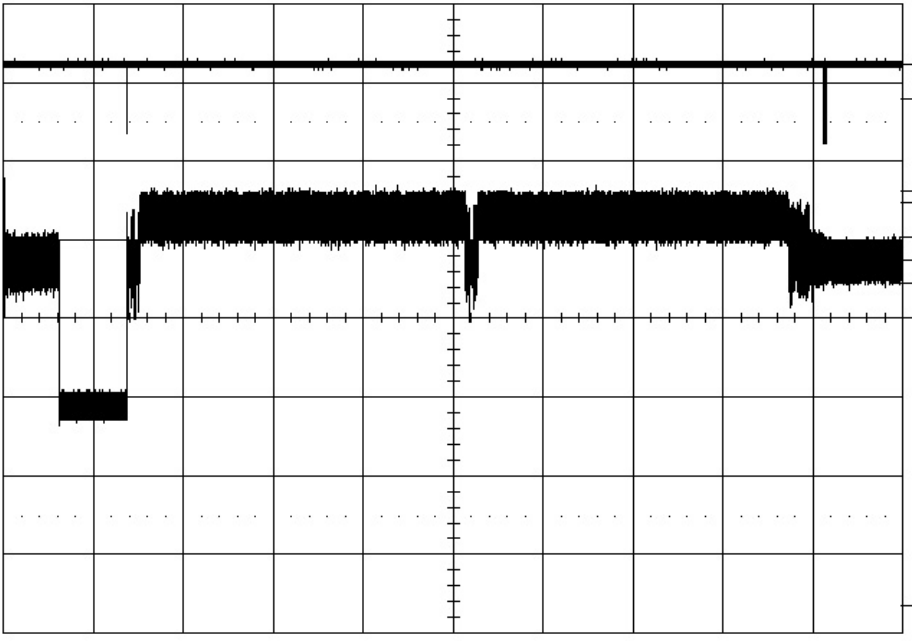

Fig. 3. Power profile of RSA with the CRT.

Results on completely unprotected RSA using the CRT. The first algorithm we attacked with our spike equipment was the pure RSA signature algorithm using the CRT:

input: $m, p, q, d_p, d_q, q^{-1} \bmod p$

$S_p := m^{d_p} \bmod p$
$S_q := m^{d_q} \bmod q$
$S := S_q + ((S_p - S_q) * q^{-1} \bmod p) * q$
return(S)

output: $m^d \bmod N$

Before discussing the results of our spike attacks on the above algorithm, we note that the inputs $p, q, d_p, d_q, q^{-1} \bmod p$ are usually stored in EEPROM, while the message m is stored in RAM. However, in order to work with the data

$p, q, d_p, d_q, q^{-1} \bmod p$ they must be moved from EEPROM into RAM or the crypto coprocessor. By varying the time when we applied the appropriate spike to the smartcard IC's power supply V_{CC}, we were able to induce the following different errors:

Observed error	Mainly due to
modification of p, q	Moving data from E^2 to coprocessor
modification of d_p, d_q	Handling data within CPU
wrong exponentiation modp, q	Error within CPU or coprocessor
modification of $q^{-1} \bmod p$	Moving data from E^2 to coprocessor
wrong combination of S_p and S_q	All listed errors
faulty signature mod$p$ *and* mod$q$	Moving data from coprocessor
wrong answer of smartcard	Fatal error within CPU

Note that the first five errors may lead to a successful attack, whereas the last two do not. Thus, we can conclude that it is absolutely necessary to have sophisticated hardware and software countermeasures to avoid such kinds of attacks. Within the remaining sections we will analyze already existing software countermeasures and also develop new and more reliable countermeasures.

Results on unprotected hardware with simple software countermeasures. Motivated by the devastating results obtained within the previous section, we hereafter tested the reliability of the naively implemented software countermeasures due to [Sh] as desribed in section 2. Thus, we applied spikes to the unprotected smartcard while computing the following RSA signature algorithm shown in figure 4.

Again, we firstly summarize some of the observed errors.

	Observed error scenarios	A	B	C
1	modification of p', q'	time dep.	time dep.	no
2	modification of d	time dep.	time dep.	yes
3	modification of d'_p, d'_q	yes	yes	yes
4	modification of r	time dep.	time dep.	yes
5	wrong exponentiation modp, q	prob. $1 - 1/r$	yes	yes
6	modification of S_p or S_q	time dep.	yes	no
7	modification of $q^{-1} \bmod p$	no	yes	no
8	error during comb. of S_p and S_q	no	yes	no
9	faulty signature mod$p$ *and* mod$q$	no	no	yes

The above table is organized as follows. The second column denotes the kind of error which might occur. Column A indicates whether the countermeasure recognizes the induced fault, column B indicates whether the corresponding faulty signature S reveals the secret key and column C says whether the countermeasure is correctly working in the corresponding case. We will briefly comment the observed errors row by row:

input: $m, p, q, d, q^{-1} \bmod p$

randomly choose a short prime r of, e.g., 32 bits
$p' := p * r$
$d'_p := d \bmod ((p-1) * (r-1))$
$q' := q * r$
$d'_q := d \bmod ((q-1) * (r-1))$

$S'_p := (m \bmod p')^{d'_p} \bmod p'$
$S'_q := (m \bmod q')^{d'_q} \bmod q'$

$S_p := S'_p \bmod p$
$S_q := S'_q \bmod q$
$S := S_q + ((S_p - S_q) * q^{-1} \bmod p) * q$

if $((S'_p \bmod r) \neq (S'_q \bmod r))$ **then**
  return(error)
else
  return(S)

output: $S = m^d \bmod (p * q)$

Fig. 4. Shamir's countermeasure.

1. During the computation of p' the value of p may be changed to some value $\tilde{p}$, such that $p' = \tilde{p}r$. Then S'_p is computed correctly modulo r, but not modulo p. If p' is destroyed later, then the check reveals the attack. If a destroyed $\tilde{p}$ will be used for the computation of d'_p then the check will not recognize this relevant fault.
2. If d is changed before the first two reductions this will not be detected but is not security relevant. If d is changed between the first two reductions, this will be recognized by the check.
3. If d'_p or d'_q is destroyed the check will detect this modification.
4. Depending on the time r is destroyed, various things can happen: either the errors will be recognized or they are not security relevant.
5. The destruction of one of the two exponentiations is the classical Bellcore attack. This will be recognized.
6. If S_p will be changed before the combination to S then the check will fail.
7. If $q^{-1} \bmod p$ will be changed then the faulty signature will reveal the key. The check will not recognize the attack.
8. Cf. last row.
9. If the correct signature is destroyed S reveals no information about the key.

4 Practical Fault Attacks Countermeasures for Unprotected Hardware

Within this section we will use the formerly discussed errors to propose a simple practical error model. Hereafter, we propose enhanced countermeasures.

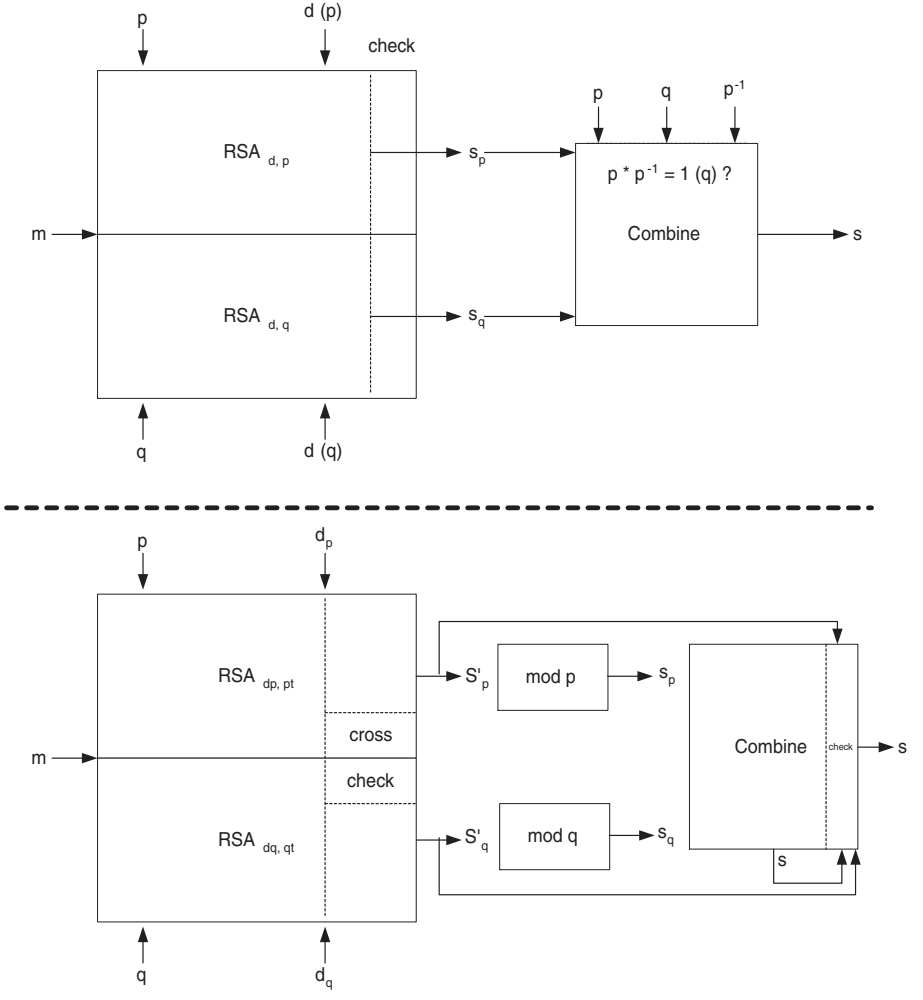

Fig. 5. Information flow during checking.

4.1 Model to Understand Resulting/Possible Faults

From the observed error scenario, we have learned by an extensive data analysis the following facts:

- During the computation, every input value to the RSA signature algorithm can be altered to a value different from the original value.
- During the computation, every variable can be changed.
- The instruction sent to the CPU or a peripheral can be changed.
- The only values to trust, are the values which are stored in ROM or EEPROM.

Armed with this knowledge, we formulated the following checking philosophy:

Check (at least in a probabilistic sense) every computed intermediate result with respect to its correctness by relying on trusted values only.

In a rough sense, this is reflected by figure 5. In this context we adapt the *transient fault model* due to [BDL] which assumes that our power spikes introduce arbitrary errors. Additionally, we assume that the attacker can induce only one spike but at a specific time chosen by himself.

4.2 Software Countermeasures Derived According to the Model

Inspired by the previous section, we developed the following countermeasures (shown in figure 6) to counteract fault attacks. It takes into account that in a practical application only d_p and d_q are given. Also, it avoids the use of the public exponent e, which in real applications is most often not known to the signature software.

We will briefly comment on this algorithm. The check after the CRT combination ensures that S is correctly computed from the data S'_p and S'_q. Therefore, it remains to guarantee that the latter ones are correct. The central check ($S_{pt}^{d_{qt}} \equiv S_{qt}^{d_{pt}} \bmod t$) proves that the two big exponentiations itself where processed in a correct way — assuming that the inputs are not compromised. Note that an erroneous pass of this check can only be due to some very subtle modifications of these input values. Such errors will be intercepted by the first two checking blocks. Finally, we would like to point out the following important advice for a careful implementation: for the two checking blocks the secret parameters d_p and d_q have to be reloaded from a secure area (EEPROM).

4.3 Measurement Results for Enhanced Software Countermeasures

By extensive penetration tests via spikes on the algorithm shown in figure 6 we obtained the following table. It proves empirically the reliability of our software countermeasures.

input: $m, p, q, d_p, d_q, q^{-1} \bmod p$

let t be a short prime number, e.g., 32 bits

$p' := p * t$
$d'_p := d_p + \text{random}_1 * (p-1)$
$S'_p := m^{d'_p} \bmod p'$
if $\neg(p' \bmod p \equiv 0 \ \wedge \ d'_p \bmod (p-1) \equiv d_p)$ **then return**(error)

$q' := q * t$
$d'_q := d_q + \text{random}_2 * (q-1)$
$S'_q := m^{d'_q} \bmod q'$
if $\neg(q' \bmod q \equiv 0 \ \wedge \ d'_q \bmod (q-1) \equiv d_q)$ **then return**(error)

$S_p := S'_p \bmod p$
$S_q := S'_q \bmod q$
$S := S_q + ((S_p - S_q) * q^{-1} \bmod p) * q$
if $\neg((S - S'_p \bmod p \equiv 0) \ \wedge \ (S - S'_q \bmod q \equiv 0))$ **then return**(error)

$S_{pt} := S'_p \bmod t$
$d_{pt} := d'_p \bmod (t-1)$
$S_{qt} := S'_q \bmod t$
$d_{qt} := d'_q \bmod (t-1)$
if $(S_{pt}^{d_{qt}} \equiv S_{qt}^{d_{pt}} \bmod t)$ **then**
  return(S)
else
  return(error)

output: $m^d \bmod (p * q)$

Fig. 6. Practically secured RSA with CRT.

Observed error scenarios	A	B	C
modification of p, p', q, q'	yes	yes	yes
modification of d'_p, d'_q	yes	yes	yes
modification of t	yes	yes	yes
wrong exp. modp, q	prob. $1 - 1/t$	yes	yes
modification of S_p or S_q	yes	yes	yes
modification of $q^{-1} \bmod p$	yes	yes	yes
error during comb. of S_p and S_q	yes	yes	yes
faulty signature mod$p$ *and* mod$q$	prob. $1 - 1/t$	no	yes

Clearly, the probability that an error is undetected is equal to $1/t$. For t a 32-bit integer, this probability is small enough; t can thus be seen as a security parameter.

5 Conclusion

We have shown that the classical Bellcore fault attack is in principal feasible when using completely unprotected microcontrollers. Moreover, it also shows that unskilled implementations of countermeasures are not always reliable. It again answers a question of Kaliski and Robshaw [KR], and shows that these attacks are indeed practical. Our investigation also reveals that one should test any conceivable countermeasures in reality against all possible attack scenarios before trusting them. This was especially done with our newly developed software countermeasures.

Although our software countermeasure seems to be very promising, we are strongly convinced that cryptographic hardware should never be used without appropriate hardware countermeasures in combination with software countermeasures. As a result, we finish with an advice given by Kaliski and Robshaw [KR] from the RSA Laboratories stating that *good engineering practices in the design of secure hardware are essential*.

References

[A] R. Anderson, *Security Engineering*, John Wiley & Sons, New York, 2001.

[AK1] R. Anderson, M. Kuhn, "Tamper Resistance – a cautionary note", *Proc. of 2nd USENIX Workshop on Electronic Commerce*, pp. 1–11, 1996.

[AK2] R. Anderson, M. Kuhn, "Low cost attacks attacks on tamper resistant devices", *Proc. of 1997 Security Protocols Workshop*, Springer LNCS vol. 1361, pp. 125–136, 1997.

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

[BDHJ$^+$] F. Bao, R. H. Deng, Y. Han, A. Jeng, A. D. Narasimbalu, T. Ngair, "Breaking public key cryptosystems on tamper resistant dives in the presence of transient faults", *Proc. of 1997 Security Protocols Workshop*, Springer LNCS vol. 1361, pp. 115–124, 1997.

[BR] M. Bellare, P. Rogaway, "The exact security of digital signatures — how to sign with RSA and Rabin", *Proc. of EUROCRYPTO '96*, Springer LNCS vol. 1070, pp. 399–416, 1996.

[BS] E. Biham, A. Shamir, "Differential fault analysis of secret key cryptosystems", *Proc. of CRYPTO '97*, Springer LNCS vol. 1294, pp. 513–525, 1997.

[BMM] I. Biehl, B. Meyer, V. Müller, "Differential fault attacks on elliptic curve cryptosystems", *Proc. of CRYPTO '00*, Springer LNCS vol. 1880, pp. 131–146, 2000.

[BMS] J. Blömer, A. May, J.-P. Seifert, personal communication, April 2002.

[CQ] C. Couvreur, J.-J. Quisquater, "Fast decipherment algorithm for RSA public-key cryptosystem", *Electronics Letters* **18**(21):905–907, 1982.

[FS] W. Fischer, J.-P. Seifert, "Note on fast computation of secret RSA exponents", *Proc. of ACISP '02*, Springer LNCS vol. 2384, pp. 136–143, 2002.

[GMO] K. Gandolfi, C. Mourtel, F. Olivier, "Electromagnetic analysis: Concrete results", *Proc. of CHES '01*, Springer LNCS vol. 2162, pp. 255–265, 2001.

[Gu1] P. Gutmann, "Secure deletion of data from magnetic and solid-state memory", *Proc. of 6th USENIX Security Symposium*, pp. 77–89, 1997.

[Gu2] P. Gutmann, "Data Remanence in Semiconductor Devices", *Proc. of 7th USENIX Security Symposium*, 1998.

[HP1] H. Handschuh, P. Pailler, "Smart Card Crypto-Coprocessors for Public-Key Cryptography", *CryptoBytes* **4**(1):6–11, 1998.

[HP2] H. Handschuh, P. Pailler, "Smart Card Crypto-Coprocessors for Public-Key Cryptography", *Proc. of CARDIS '98*, Springer LNCS vol. 1820, pp. 372–379, 1998.

[ISO] International Organization for Standardization, "ISO/IEC 7816-3: Electronic signals and transmission protocols", `http://www.iso.ch`, 2002.

[JLQ] M. Joye, A. K. Lenstra, J.-J. Quisquater, "Chinese remaindering based cryptosystem in the presence of faults", *Journal of Cryptology* **12**(4):241–245, 1999.

[JPY] M. Joye, P. Pailler, S.-M. Yen, "Secure Evaluation of Modular Functions", *Proc. of 2001 International Workshop on Cryptology and Network Security*, pp. 227–229, 2001.

[JQBD] M. Joye, J.-J. Quisquater, F. Bao, R. H. Deng, "RSA-type signatures in the presence of transient faults", *Cryptography and Coding*, Springer LNCS vol. 1335, pp. 155–160, 1997.

[JQYY] M. Joye, J.-J. Quisquater, S. M. Yen, M. Yung, "Observability analysis — detecting when improved cryptosystems fail", *Proc. of CT-RSA Conference 2002*, Springer LNCS vol. 2271, pp. 17–29, 2002.

[KR] B. Kaliski, M. J. B. Robshaw, "Comments on some new attacks on cryptographic devices", *RSA Laboratories Bulletin* **5**, July 1997.

[Kn] D. E. Knuth, *The Art of Computer Programming, Vol.2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Reading MA, 1999.

[Koca] O. Kocar, "Hardwaresicherheit von Mikrochips in Chipkarten", *Datenschutz und Datensicherheit* **20**(7):421–424, 1996.

[Koch] P. Kocher, "Timing attacks on implementations of Diffie-Hellmann, RSA, DSS and other systems", *Proc. of CYRPTO '97*, Springer LNCS vol. 1109, pp. 104–113, 1997.

[KJJ] P. Kocher, J. Jaffe, J. Jun, "Differential Power Analysis", *Proc. of CYRPTO '99*, Springer LNCS vol. 1666, pp. 388–397, 1999.

[Ma] D. P. Maher, "Fault induction attacks, tamper resistance, and hostile reverse engineering in perspective", *Proc. of Financial Cryptography*, Springer LNCS vol. 1318, pp. 109–121, 1997.

[MvOV] A. J. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, New York, 1997.

[NR] D. Naccache, D. M'Raihi, "Cryptographic smart cards", *IEEE Micro*, pp. 14–24, 1996.

[Pe] I. Petersen, "Chinks in digital armor — Exploiting faults to break smartcard cryptosystems", *Science News* **151**(5):78–79, 1997.

[Ro] T. Rosa, "Future Cryptography: Standards are not enough", *Proc. of Security and Protection of Information 2001*, pp. 237–245, 2001.

[RSA] R. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Comm. of the ACM* **21**:120–126, 1978.

[SQ] D. Samyde, J.-J. Quisquater, "ElectroMagnetic Analysis (EMA): Measures and Countermeasures for Smart Cards", *Proc. of Int. Conf. on Research in Smart Cards, E-Smart 2001*, Springer LNCS vol. 2140, pp. 200–210, 2001.

[Sh] A. Shamir, "Method and Apparatus for protecting public key schemes from timing and fault attacks", U.S. Patent Number 5,991,415, November 1999; also presented at the rump session of EUROCRYPT'97.

[YJ] S.-M. Yen, M. Joye, "Checking before output may not be enough against fault-based cryptanalysis", *IEEE Trans. on Computers* **49**:967–970, 2000.

[YKLM1] S.-M. Yen, S.-J. Kim, S.-G. Lim, S.-J. Moon, "RSA Speedup with Residue Number System immune from Hardware fault cryptanalysis", *Proc. of the ICISC 2001*, Springer LNCS vol. 2288, pp. 397–413, 2001.

[YKLM2] S.-M. Yen, S.-J. Kim, S.-G. Lim, S.-J. Moon, "A countermeasure against one physical cryptanalysis may benefit another attack", *Proc. of the ICISC 2001*, Springer LNCS vol. 2288, pp. 414–427, 2001.

[ZM] Y. Zheng, T. Matsumoto, "Breaking real-world implementations of cryptosystems by manipulating their random number generation", *Proc. of the 1997 Symposium on Cryptography and Information Security*, 1997.