Combined Attack on CRT-RSA. Why Public Verification Must Not Be Public?

Guillaume Barbu, Alberto Battistelllo, Guillaume Dabosville, Christophe Giraud, Guénaël Renault, Soline Renner, Rina Zeitoun

Combined Attack on CRT-RSA

Why Public Verification Must Not Be Public?

Guillaume Barbu[1°], Alberto Battistello[1°], Guillaume Dabosville[1◇], Christophe Giraud[1°], Guénaël Renault[2⋆], Soline Renner[1°,3], and Rina Zeitoun[1◇,2]

[1] Oberthur Technologies – Security & Crypto Groups
[°]4, allée du Doyen Georges Brus, 33 600 Pessac, France
[◇]71-73, rue des Hautes Pâtures, 92 726 Nanterre Cedex, France

[2] UPMC, Université Paris 6, INRIA, Centre Paris-Rocquencourt, PolSys Project-team, CNRS, UMR 7606, LIP6
4 place Jussieu, 75252 Paris, Cedex 05, France

[3] Institut Mathématiques de Bordeaux, Université Bordeaux I
351, cours de la Libération, 33 405 Talence Cedex, France

<f.lastname>@oberthur.com
guenael.renault@lip6.fr

Abstract. This article introduces a new Combined Attack on a CRT-RSA implementation resistant against Side-Channel Analysis and Fault Injection attacks. Such implementations prevent the attacker from obtaining the signature when a fault has been induced during the computation. Indeed, such a value would allow the attacker to recover the RSA private key by computing the *gcd* of the public modulus and the faulty signature. The principle of our attack is to inject a fault during the signature computation and to perform a Side-Channel Analysis targeting a sensitive value processed during the Fault Injection countermeasure execution. The resulting information is then used to factorize the public modulus, leading to the disclosure of the whole RSA private key. After presenting a detailed account of our attack, we explain how its complexity can be significantly reduced by using lattice reduction techniques. We also provide simulations that confirm the efficiency of our attack as well as two different countermeasures having a very small impact on the performance of the algorithm. As it performs a Side-Channel Analysis during a Fault Injection countermeasure to retrieve the secret value, this article recalls the need for Fault Injection and Side-Channel Analysis countermeasures as monolithic implementations.

Keywords: CRT-RSA, Combined Attacks, Fault Injection, Side-Channel Analysis, Coppersmith's methods.

⋆ This work was partly supported by the French ANR under the Computer Algebra and Cryptography (CAC) project (ANR-09-JCJCJ-0064-01).

1 Introduction

Since the seminal work of Kocher published in 1996 [1], Side-Channel Analysis (SCA) has raised a huge interest in both academic and industrial communities. This kind of attack is based on the fact that side-channel leakages of embedded devices contain information on the values manipulated inside the device. Therefore any sensitive variable carelessly used can be recovered by an attacker using SCA. Originally, time execution was used as side-channel leakage but the exploitation of power consumption and electromagnetic radiation became quickly the most efficient way to attack embedded cryptography [2, 3]. Over the years, many improvements have been made leading to very efficient attacks and very ingenious countermeasures [4].

In parallel to SCA, Fault Injection (FI) provides the attacker with another way to attack embedded devices. Such attacks aim at disturbing cryptographic computations and the analysis of corresponding faulty outputs allows the attacker to recover the secret key [5]. Shortly after the original publication focusing on RSA implementation [6], many other articles have been published to present FI attacks on various cryptosystems such as DES, ElGamal or DSA signature schemes [7,8]. As for SCA, FI has been deeply studied over the last decade [9] and the consequences of both attacks on the industry are huge since secure products must now be certified to prove their resistance against such threats.

Over the last few years, the cryptographic community has investigated the possibility of combining the two previous kinds of attacks. This has resulted in a new class of attacks called Combined Attacks (CA) that can defeat implementations which are meant to resist both SCA and FI. However, as far as we know only four CA have been published since their introduction in 2007, proving the difficulty to conceive such attacks [10–13].

Nowadays, most embedded devices implement a large variety of cryptosystems to ensure the security of the sensitive assets they contain. As well as being the first practical public-key cryptosystem published, RSA [14] has also been the most widely used for many years. In particular, the RSA using the Chinese Remainder Theorem (CRT), providing a speed-up factor of four compared to the original implementation, is available on most ID, banking and mobile smart cards. Obviously, this cryptosystem has been the main target of SCA and FI attackers leading to the development of efficient countermeasures having the smallest impact on both memory consumption and time execution due to the constraints of embedded environment.

In this article, we describe a new Combined Attack against a CRT-RSA implementation resistant to SCA by using blinding countermeasures and protected against FI by verifying the signature using the public exponent. Such an implementation is known to resist each and every kind of attack published so far. However, we demonstrate that when injecting a fault during the signature computation, a value depending on the message and on a multiple of a secret prime is manipulated in plain during the public verification. Therefore, we notice that one can use SCA to gain some information on such a sensitive value. The recovered information can then be used to factorize the RSA modulus and

thus to reveal the whole private key. Besides, we exploit lattice techniques, and in particular Coppersmith's methods for finding small solutions to polynomial equations [15, 16], to significantly reduce the complexity of our CA.

The rest of this paper is organised as follows. In Section 2 we briefly recall some basics on CRT-RSA signature as well as the corresponding attacks and countermeasures. In Section 3 we describe our new CA on a CRT-RSA implementation that is known to resist both SCA and FI attacks. In Section 4 we present the results of our simulations which prove the efficiency of our new attack. We then improve its complexity by using lattice reduction techniques in Section 5. Finally, we suggest in Section 6 possible countermeasures having a negligible penalty on the performance of the algorithm.

2 Previous Works

In this section we briefly recall the RSA signature, in particular the CRT mode. Secondly we present the principal attacks on such an algorithm as well as the main countermeasures.

2.1 RSA on Embedded Systems

Since its introduction in 1978, the RSA cryptosystem has become one of the most used public-key cryptosystems, especially in electronic signature schemes [14]. In the following we briefly recall how to compute the RSA signature in both standard and CRT modes.

Let N denote the public modulus being the product of two secret large prime integers p and q. Let d refer to the private exponent and e refer to the public exponent satisfying $de = 1 \bmod \varphi(N)$, where φ denotes Euler's totient function. The RSA signature of a message $m \in \mathbb{Z}_N$ is then obtained by computing $S = m^d \bmod N$. To verify the signature, one computes $S^e \bmod N$ and checks if the corresponding result is equal to m.

In embedded systems, most RSA implementations use the Chinese Remainder Theorem (CRT) which yields an expected speed-up factor of four [17]. Following the CRT-RSA algorithm, the signature generation is composed of two exponentiations $S_p = m^{d_p} \bmod p$ and $S_q = m^{d_q} \bmod q$, where $d_p = d \bmod p - 1$ and $d_q = d \bmod q - 1$. The signature is then obtained by recombining S_p and S_q, which is usually done by using Garner's formula [18]:

$$S = CRT(S_p, S_q) = S_q + q(i_q(S_p - S_q) \bmod p) \ , \tag{1}$$

where $i_q = q^{-1} \bmod p$.

2.2 Attacks and Countermeasures

Side-Channel Analysis Side-Channel Analysis (SCA) has been introduced by the publication of the so-called timing attacks in 1996 [1]. SCA exploits the

dependency between the manipulated data or the executed instruction and the side-channel leakages which can be monitored during the algorithm execution. Examples of such leakages are the power consumption or the electromagnetic radiation of the device. When only one measure is required to exploit sensitive information, the attack is called Simple Passive Analysis (SPA) [2]. In the case of a straightforward *Square-and-Multiply* exponentiation, SPA consists for instance in observing if the squaring and multiplication operations have different patterns in the corresponding side-channel leakages [2]. Hence, the secret exponent can be directly extracted from one measurement. In the literature, a common countermeasures consists in using a so-called *regular* algorithm which performs the same operation whatever the exponent bit value such as the *Square-Always* or *Montgomery ladder* algorithms [19, 20].

Moreover, attacks based on side-channel leakages have evolved to a type of SCA called Differential Passive Analysis (DPA) [2] which requires a large number of measurements. This type of attack applies a statistical treatment on the curves to recover information on the manipulated values.Nowadays a common statistic tool used to perform such a statistical treatment, is the Pearson correlation coefficient:

$$\rho_k = \frac{\mathrm{cov}(\mathcal{L}, H)}{\sigma_{\mathcal{L}}\sigma_H}, \tag{2}$$

where $\mathcal{L}$ is the set of curves and H depends on a known value m and on a guess of a small part of a secret k. Such an attack is called Correlation Power Analysis (CPA) [21]. In the literature, many different CPAs have been published to attack the RSA cryptosystem [22]. For instance in the CRT-mode, an attacker can mount a CPA to recover the private parameter q during the CRT-recombination, *cf.* Rel. (1), by observing the leakage obtained during the manipulation of the value $i_q(S_p - S_q) \bmod p$ and by making a guess on a few bits of q. Hence, an attacker can obtain the whole secret q by performing a CPA for each of its subpart (typically for each byte). However, half of q is sufficient to recover the rest of q by using Coppersmith's attack [23]. Classical countermeasures to resist CPA consist in randomizing the modulus, the message and the exponent [22].

Fault Injection RSA has been the first cryptosystem to succumb to Fault Injection [24]. In the following, we describe such an attack in the CRT case. Assume that a fault is injected during the computation of S_p leading to a faulty signature $\widetilde{S}$. Since $S \equiv S_p \bmod p$ and $S \equiv S_q \bmod q$, one can notice that $\widetilde{S} \equiv S \bmod q$ but $\widetilde{S} \not\equiv S \bmod p$. Therefore, the secret parameter q can be easily recovered by computing the *gcd* of $S - \widetilde{S}$ and N. The rest of the private key can then be straightforwardly deduced.

When it is not possible to sign the same message twice and if the message is known to the attacker, a variant of this attack consists in computing the *gcd* of $\widetilde{S}^e - m$ and N to obtain the secret value q [25].

Moreover, the effect of fault injections on CRT-RSA is not limited to the disturbance of S_p or S_q. Indeed, a fault injected in any part of the key parameters (i.e. p, q, d_p, d_q or i_q), in the message m at the beginning of either S_p or S_q

computation, or even during the CRT-recombination can lead to a useful faulty signature.

In the literature, four different fault models are generally considered to define the attacker's capabilities [26]:

- the *random* fault model: the bits are changed to a uniformly distributed random value;
- the *bit-flip* fault model: in that case, affected bits are flipped to their complementary value;
- the *stuck-at* fault model: the fault sets the bits to 0 or to 1, depending on the underlying hardware;
- the *unknown constant* fault model: the fault always sets the bits to the same unknown value.

Moreover, these faults do not necessarily modify a whole temporary result. Indeed, it is generally considered that the number of bits affected by the fault is linked to the CPU word-size which is generally 8, 16 or 32 bits.

The most natural way to counteract fault injection on RSA-type signature is to check the correctness of the signature S before outputting it [24]. More precisely, the signature is returned iff $S^e \bmod N = m$. Moreover, such a method requires very little overhead since the public exponent e is usually small in practice (typically 3, 17 or $2^{16} + 1$).

Other methods getting rid of e have also been proposed but they do not offer the same level of security and are generally slower than the public verification [27–29].

Combined Attacks The idea to combine SCA and FI appeared in 2007 when Amiel *et al.* proposed a so-called Combined Attack (CA) on an RSA implementation protected against FI and SPA [10]. They noticed that by setting to zero one of the temporary registers used in the Montgomery ladder, its structure becomes unbalanced, revealing the value of the secret exponent by SPA. Following this publication, three other papers have been published taking advantage of this new way of defeating embedded security. Two of them present a CA against a secured AES implementation [12, 13]. The third one focuses on the elliptic curve scalar multiplication [11].

Despite its theoretical effectiveness, the combination of SCA and FI is very difficult in practice, explaining the lack of practical experiments in the current literature.

Lattices Randomized RSA encoding schemes are usually considered to be resistant to traditional FI attacks since a part of the message is unknown to the attacker and varies for each signature computation. However this common assumption has to be mitigated regarding the works of [30] and [31] which defeat two randomised RSA encoding schemes. These attacks use Coppersmiths

method to solve a bivariate polynomial whose coefficients are built thanks to the generated faulty signatures.

More recently in [32], the authors present an attack taking advantage of the disturbance of the public modulus. The generated faulty signatures allow them to build a lattice, which in turns leads to factorize the public modulus.

Although [30], [31] and [32] also apply lattice reduction techniques as carried out in this paper, the attack presented hereafter does not share the same context since we consider a secured implementation which never returns the faulty signature.

3 A New Combined Attack on CRT-RSA

3.1 Context and Principle

As stated in Section 2, several countermeasures have been developed to protect CRT-RSA embedded implementations against both SCA and FI. In the framework of this article, we consider an algorithm protected:

- against SCA by using message and exponent blinding as suggested in [33], a regular exponentiation algorithm such as the Square Always [20] and a mask refreshing method along the exponentiation such as the one presented in [34]. Moreover, the blinding is kept all along the CRT-recombination.
- against FI by verifying the signature using the public exponent e [24]. In addition, we also use the approach presented in [35] which mainly consists in checking the result of the verification twice to counteract double FI attacks.

Fig. 1 depicts the main steps of such an implementation where the k_i's are random values (typically of 64 bits) generated at each execution of the algorithm and S'_p, S'_q and S' represent the blinded version of S_p, S_q and S respectively.

In the following, we assume that the fault injected by the attacker follows either the *bit-fault*, the *stuck-at* or the *unknown constant* fault models (cf. Section 2.2). Moreover, we assume the attacker is able to choose which byte of the message is affected by the fault.

As mentioned in Section 2.2, injecting a fault during the signature computation leads to a faulty signature that allows the attacker to recover the private key. However in the implementation considered in this paper, the verification with the public exponent detects such a disturbance and the faulty signature is never revealed to the attacker. The main contribution of this paper is to show that in this case, an SCA can still allow the attacker to gain enough information on the faulty signature to recover the private key.

At first glance, it seems impossible to perform such an attack during the signature process due to the blinding countermeasure. However by observing Fig. 1, one may note that the faulty signature $\widetilde{S}$ remains blinded until the end of exponentiation with e modulo N. Therefore if we can express $\widetilde{S}^e \bmod N$ in terms of the message m and of the private key then we can perform an SCA on this value. In the next section, we exhibit such a relation allowing us to mount a CA on an SCA-FI-resistant CRT-RSA implementation.

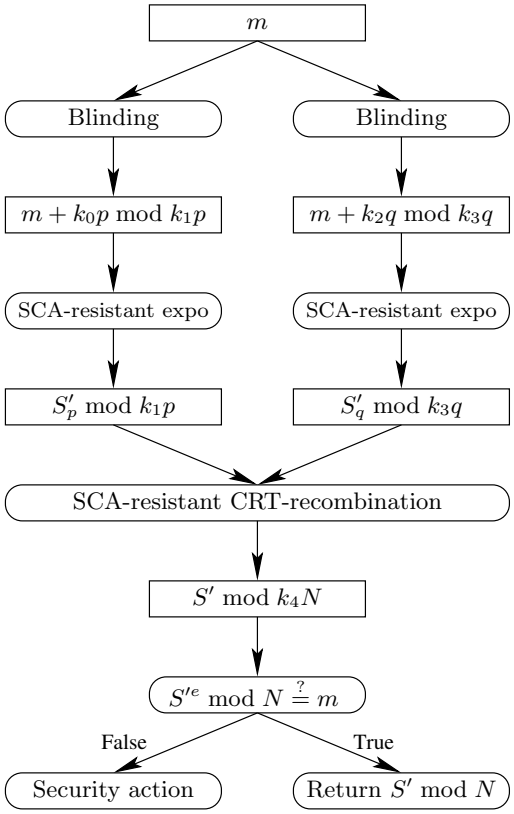

Fig. 1. Main steps of a CRT-RSA implementation secure against SCA and FI.

3.2 A Useful Relation

Proposition 1. *If a fault ε is induced in m such that the faulty message $\widetilde{m}$ is equal to $m + \varepsilon$ at the very beginning of the computation of S_p then*

$$\widetilde{S}^e \equiv m + \varepsilon q i_q \bmod N \ , \tag{3}$$

where $\widetilde{S}$ corresponds to the faulty signature.

Proof. By definition of the CRT-RSA signature, we have:

$$\begin{cases} \widetilde{S} \equiv (m+\varepsilon)^d \bmod p \\ \widetilde{S} \equiv m^d \bmod q \end{cases} \tag{4}$$

It comes then straightforwardly that:

$$\begin{cases} \widetilde{S}^e \equiv m + \varepsilon \bmod p \\ \widetilde{S}^e \equiv m \bmod q \end{cases} \tag{5}$$

Finally, applying Gauss recombination to (5) leads to (3) since:

$$\widetilde{S}^e \equiv pi_p m + qi_q(m+\varepsilon) \bmod N \quad (6)$$
$$\equiv (pi_p m + qi_q m) + \varepsilon q i_q \bmod N \quad (7)$$
$$\equiv m + \varepsilon q i_q \bmod N \ , \quad (8)$$

where $i_p = p^{-1} \bmod q$. □

One may note that a similar relation holds if m is disturbed at the very beginning of S_q computation due to the symmetrical roles of p and q in both branches of the CRT-RSA. For the sake of simplicity, we will use the case where S_p computation is disturbed in the rest of this paper.

3.3 Recovering the Private Key

Following the attack's principle depicted in Section 3.1 and using Proposition 1, we will now present in detail the main steps of our attack.

Firstly, the attacker asks the embedded device to sign several messages m_i through a CRT-RSA implemented as described in Section 3.1. For each signature, the computation of S_q is performed correctly and a constant additive error ε is injected on the message m_i at the beginning of each S_p computation. Then during each signature verification, the attacker monitors the corresponding side-channel leakage $\mathcal{L}_i$ which represents the manipulation of $\widetilde{S}_i^e \bmod N$.

From Proposition 1, we know that there exists a sensitive value k satisfying the relation $\widetilde{S}_i^e \bmod N = m_i + k$. Therefore, the attacker will perform a CPA to recover this sensitive value by computing $\rho_k(m_i + k, \mathcal{L}_i)$ for all the possible values of k (cf. Section 2.2).

Depending on the set $\{(m_i, \widetilde{S}_i^e \bmod N)\}_i$, it follows from Rel. (3) that k will be equal either to $\varepsilon q i_q \bmod N$ or to $\varepsilon q i_q \bmod N - N$. Therefore, the value $\widehat{k}$ producing the strongest correlation at the end of the CPA will be one of these two values. Once $\widehat{k}$ recovered, the attacker must then compute the *gcd* between $\widehat{k}$ and N, which leads to the disclosure of q. From this value, the private key is straightforwardly computed.

Regarding the practicality of our fault model (i.e. a constant additive fault), one may note that by fixing a small part of the message (e.g. a byte), the disturbance of such a part in either the stuck-at, the bit-flip or the unknown constant fault model results in a constant additive error during the different signature computations. Therefore our fault model is definitely valid if the attacker can choose the messages to sign, or even if she can only have the knowledge of the messages and attack only those with a given common part.

Finally, one may note that it is not possible to perform a statistical attack targeting the full value of k at once due to its large size (i.e. $\lceil \log_2(N) \rceil$ bits). However, one can attack each subpart of this value, for instance by attacking byte per byte starting with the least significant one in order to be able to propagate easily the carry. It is worth noticing that CPA only applies when the corresponding part of the message varies. Therefore, if the attacker fixes the MSB of the

message, then the corresponding set of measurements can be used to recover the whole but last byte of $\hat{k}$. In such a case, a brute force search can be used to recover the missing byte.

In the next section, we present simulations of our attack which prove the efficiency of our method and which are based on the attacker's capability to inject the same fault and on the noise of the side-channel measurements.

4 Experiments

The success of the attack presented in Section 3 relies on the ability of the attacker to both measure the side-channel leakage of the system during the signature verification and induce the same fault ε on the different manipulated messages.

In order to evaluate the effectiveness of this attack, we have experimented it on simulated curves of the side-channel leakage $\mathcal{L}$, according to the following leakage model:

$$\mathcal{L}(d) = HW(d) + \mathcal{N}(\mu, \sigma) \tag{9}$$

with $\mathcal{N}(\mu, \sigma)$ a Gaussian noise of mean μ and standard deviation σ, and $HW(d)$ the Hamming weight function evaluated for the manipulated data d. In the framework of our experiments, we consider that the processor manipulates 8-bit words and we use three different levels of noise, namely $\sigma = 0.1$, 1 and 5.

As well as the side-channel leakage, the faults were also simulated by setting the most significant word of the message m to all-0 at the very beginning of the S_p computation. These faults were induced with a given success rate r, varying in our different experiment campaigns (namely 50%, 10% and 1%).

Depending on the experimental settings, all the different words of the secret value will be equivalently correlated with the simulated curves. The graphs presented in Fig. 2 present the convergence of the correlation for each possible value k of one particular byte (the 5^{th} least-significant byte) of the secret depending on the number of side-channel measurements with different simulation settings σ and r.

As exposed in Fig. 2, the number of traces required to recover the secret value depends essentially on the fault injection success rate. This comes from the fact that every wrongly-faulted computation can be considered as noise in the scope of our statistical analysis. The number of curves required to retrieve the secret word grows as the fault injection success rate decreases and to a fewer extent as the noise of the side-channel leakage increases.

With regards to the results obtained when $\sigma = 5$ and $r = 10\%$, which appear to be plausible values in practice, it took us 3.35 seconds to retrieve one byte of the secret value by performing the CPA on $15,000$ curves of 128 points each[4]. Assuming a genuine curve should be made of at least $5,000$ points, we can estimate the time required to practically perform the attack to about 1 minute

[4] The execution time given here and in Section 5.2 have been obtained on a 32-bit CPU @3.2GHz

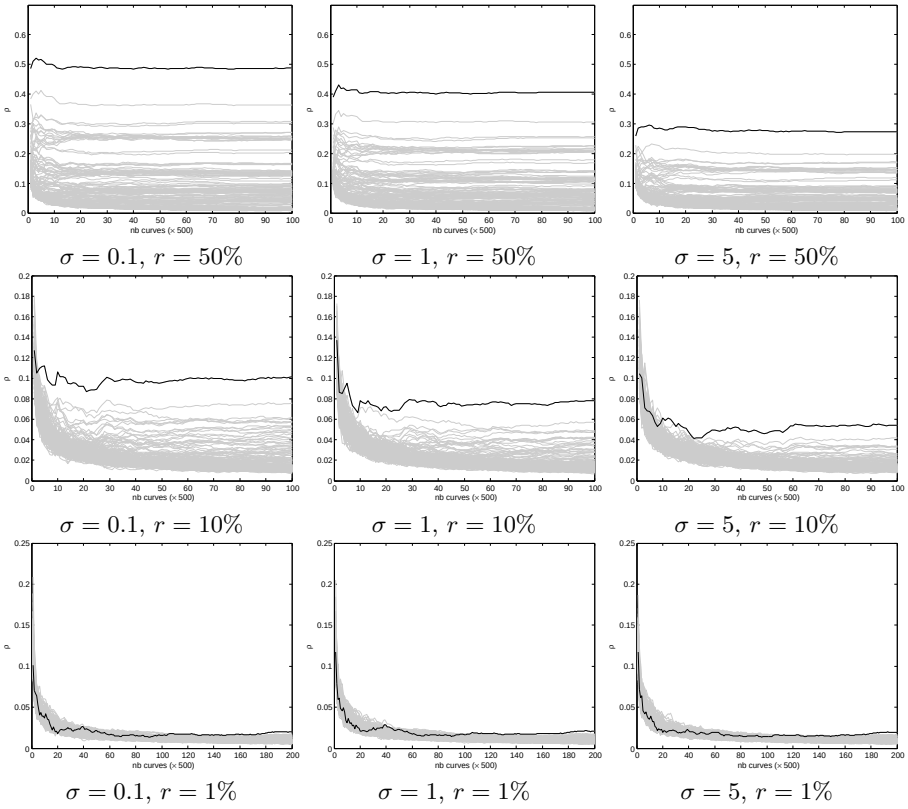

Fig. 2. Convergence of the correlation for the 256 possible values k_i for the secret (the correct one being depicted in black) depending on the number of side-channel measurements ($\times 500$) for different levels of noise σ and fault injection success rates r.

5 seconds per byte. That is to say, it takes about 2 hours 20 minutes to recover the complete secret value if we consider a 1024-bit RSA module.

For the sake of clarity, we restrained the experiments presented here to the case where the processor manipulates 8-bit words, and thus ε is an 8-bit error. The same experiments have been run for processor word-size up to 32 bits with success. Besides, about the same number of curves were necessary for the CPA to highlight the correct secret byte.

Section 5 shows how it is possible to considerably reduce the complexity of our attack thanks to the use of lattice techniques.

5 Reducing the Attack Complexity Using Coppersmith's Methods

This section aims at improving the attack complexity using Coppersmith's methods. It is in line with the problem of factorizing N knowing half part of prime p

(or q), that was solved in [16]. With respect to our case, we highlight that if the CA presented in Section 3 provides about half of the secret $\varepsilon qi_q \bmod N$, then the other half part can be straightforwardly computed by solving a well-designed modular polynomial equation that we elaborate in the sequel. Besides, we deal with two cases (ε known and unknown), depending on the fault model that is considered.

5.1 Bringing Up the Original Problem to Solving a Modular Equation

Suppose we are given the t least significant bits (LSB) of the secret $\varepsilon qi_q \bmod N$. The latter value can be rewritten as follows:

$$\varepsilon qi_q \equiv 2^t x_0 + k \bmod N \ , \tag{10}$$

where t and k are known values, and x_0 is the $\lceil \log_2(N) - t \rceil$-bit unknown integer that is to be recovered.

Lemma 1. *The unknown secret part x_0 is solution of the polynomial $P_\varepsilon(x)$:*

$$P_\varepsilon(x) = x^2 + c\ (2^{t+1}k - 2^t\varepsilon)\ x + c\ (k^2 - k\varepsilon) \equiv 0 \bmod N \ , \tag{11}$$

where $c = (2^{2t})^{-1} \bmod N$, k, t, N are known, and ε is the induced fault.

Proof. The Bézout identity applied to our context yields that primes p and q interrelate with integers $i_p = p^{-1} \bmod q$ and $i_q = q^{-1} \bmod p$ by the following relation:

$$pi_p + qi_q \equiv 1 \bmod N \ . \tag{12}$$

Multiplying (12) by ε leads to the relation $\varepsilon pi_p + \varepsilon qi_q \equiv \varepsilon \bmod N$, or equivalently to $\varepsilon pi_p \equiv \varepsilon - \varepsilon qi_q \bmod N$. Therefore, replacing εqi_q using (10) allows us to deduce an equivalence for εpi_p:

$$\varepsilon pi_p \equiv \varepsilon - 2^t x_0 - k \bmod N \ . \tag{13}$$

As $N = pq$, we then multiply (10) by (13), to get the relation:

$$\varepsilon qi_q \cdot \varepsilon pi_p \equiv (2^t x_0 + k) \cdot (\varepsilon - 2^t x_0 - k) \equiv 0 \bmod N \ . \tag{14}$$

Eventually, developing the right-hand side of (14), and multiplying it by $c = (2^{2t})^{-1} \bmod N$ leads to the obtention of the monic polynomial $P_\varepsilon(x)$. □

The initial problem of retrieving the unknown part of $\varepsilon qi_q \bmod N$ is thereby altered in solving the modular polynomial equation (11). In the sequel, we deal with two possible cases regarding ε, whether it is known to the adversary or not.

Case 1. **The fault ε is known to the adversary**

This case corresponds to the *bit-flip* and *stuck-at* fault models (Section 2.2) since the message is known to the attacker and the fault location can be chosen. In both cases, since the fault ε is known, the problem is reduced to solving a univariate modular polynomial equation, *cf.* Rel. ((11)). This problem is known to be hard. However, when the integer solution x is small, Coppersmith showed [23] that it can be retrieved using the well-known LLL algorithm. Accordingly, we induce the following proposition:

Proposition 2. *Given $N = pq$ and the low order $1/2\ \log_2(N)$ bits of $\varepsilon qi_q \bmod N$ and assuming ε is known, one can recover in time polynomial in $(\log_2(N), d)$ the factorization of N.*

Proof. From Coppersmith's Theorem [16], we know that, given a monic polynomial $P(x)$ of degree d, modulo an integer N of unknown factorization, and an upper bound X on the desired solution x_0, one can find in polynomial time all integers x_0 such that

$$P(x_0) \equiv 0 \bmod N \qquad \text{and} \qquad |X| < N^{1/d} \ . \tag{15}$$

In our case we have $d = 2$, and since x_0 is a $\lceil \log_2(N) - t \rceil$-bit integer, we know that $|x_0| < X = 2^{\lceil \log_2(N) - t \rceil}$. Thus, the condition in (15) becomes $2^{\lceil \log_2(N) - t \rceil} < N^{1/2}$, *i.e.*

$$t > \frac{1}{2} \log_2(N) \ . \tag{16}$$

Therefore, knowing at least half part of the secret $\varepsilon qi_q \bmod N$ allows to recover the whole secret. As previously done, computing $gcd(\varepsilon qi_q \bmod N, N)$ provides the factorization of N. □

Note that the method is deterministic, and as will be seen further (Table 1), it is reasonably fast.

Case 2. **The fault ε is unknown to the adversary**

This case is met in the *unknown constant* fault model (Section 2.2). In such a case, one can consider the polynomial $P_\varepsilon(x)$ as a bivariate modular polynomial equation with unknown values x and ε. This specific scheme has also been studied by Coppersmith and includes an additional difficulty of algebraic dependency of vectors which induces the heuristic characteristic of the method [15]. As depicted in Section 5.2, in our experiments nearly 100% of the tests verified the favorable property of independency. Accordingly, in this vast majority of cases, the following proposition holds:

Proposition 3. *Under an hypothesis of independency (see discussion above), given $N = pq$ and the low order $1/2\ \log_2(N) + s$ bits of $\varepsilon qi_q \bmod N$, where s denotes the bitsize of ε, and assuming ε is unknown, one can recover in time polynomial in $(\log_2(N), d)$ the factorization of N.*

Proof. Coppersmith's Theorem for the bivariate modular case [15] notifies that given a polynomial $P(x, \varepsilon)$ of total degree d, modulo an integer N of unknown factorization, and upper bounds X and E on the desired solutions x_0, ε_0, it may be possible (heuristic) to find in polynomial time all integer couples (x_0, ε_0) such that

$$P(x_0, \varepsilon_0) \equiv 0 \bmod N \qquad \text{and} \qquad |X \cdot E| < N^{1/d} \ . \tag{17}$$

In our case, we have $d = 2$ and $E = 2^s$. The integer x_0 is $\lceil \log_2(N) - t \rceil$-bit long, therefore we have $X = 2^{\lceil \log_2(N) - t \rceil}$. Thus, the condition in (17) becomes $2^{\lceil \log_2(N) - t \rceil} \cdot 2^s < N^{1/2}$, *i.e.*

$$t > \frac{1}{2} \log_2(N) + s \ . \tag{18}$$

This means that knowing s more bits of the secret $\varepsilon q i_q \bmod N$ than before, would allow the recovering of the whole secret. □

Remark 1. The bound of success in Proposition 3 can actually be slightly improved using results of [36]. Indeed, Coppersmith's bound applies to polynomials whose monomials shape is rectangular, while in our case the monomial ε^2 does not appear in $P(x, \varepsilon)$ which corresponds to what they called an *extended rectangle* in [36]. For the sake of simplicity, we only mentioned Coppersmith's bound since practical results are similar.

5.2 Results From Our Implementation

We have implemented this lattice-based improvement using Magma Software [37], with N a 1024-bit integer *i.e.* 128 bytes long, in the cases where ε is an 8-bit known value (for Case 1) and a 32-bit unknown value (for Case 2). We chose Howgrave-Graham's method [38] for the univariate case, and its generalization by Jochemsz *et al.* [39] for the bivariate case since both have the same bound of success as Coppersmith's method (sometimes even better for [39]) and they are easier to implement. As we know, the theoretical bound given in Coppersmith's method is only asymptotic [16]. Thus, we report in Table 1 (for Case 1) and in Table 2 (for Case 2) the size t (in bytes) of the secret $\varepsilon q i_q \bmod N$ that is known to the attacker before applying Coppersmith's method, the lattice dimension used to solve (11) and finally the timings of our attack.

As depicted in Table 1, and combining these results with the experiments of Section 4, the best trade-off is to perform a CPA on the 66 first bytes, taking $66 \times 1m05s = 1h11m30s$, and to retrieve the 62 remaining bytes using lattices in $34.25s$, bringing the total time up to 1 hour 12 minutes, instead of the previous 2 hours 20 minutes.

In order to illustrate Case 2, we have chosen to rather show our results for ε being a 32-bit value, since when ε is 8-bit long, we obtained slightly better results by considering the 255 possible values of the variable ε together with their corresponding polynomials $P_\varepsilon(x)$, and by running the method on each of the polynomials until finding the solution x_0 that allows to factorize N. This

Table 1. Size t required (in bytes) for the method to work and timings (Magma V2.17-1), as a function of the lattice dimension in Case 1 (ε known, being an 8-bit integer).

Size t required (bytes)	86	72	70	69	68	67	**66**	65	64
Dimension of the lattice	3	9	11	15	17	23	**37**	73	Theoretical
Time for LLL (seconds)	< 0.01	0.03	0.07	0.29	0.52	2.63	**34.25**	2587.7	bound

Table 2. Size t required (in bytes) for the method to work and timings (Magma V2.17-1), as a function of the lattice dimension in Case 2 (ε unknown, being a 32-bit integer).

Size t required (bytes)	86	78	76	74	73	**72**	71	70	69
Dimension of the lattice	5	12	22	35	51	**70**	117	201	Theoretical
Time for LLL (seconds)	< 0.01	0.02	0.16	1.17	5.88	**30.22**	605.9	12071.1	bound

indeed leads to a best trade-off of 70 bytes required from the CPA and the 58 remaining bytes computed with lattices by performing 255 times the LLL algorithm in the worst case, for a total of $68 \times 1m05s + 255 \times 0.52s$, *i.e.* 1 hour 16 minutes instead of 2 hours 20 minutes. Besides, this exhaustive search can be performed in parallel and it also has the advantage to be deterministic.
However, when ε is 32-bit long, an exhaustive search becomes impractical and, as depicted in Table 2, the best trade-off would be to perform a CPA on 72 bytes and to compute the 56 remaining bytes with lattices (even if heuristic, it worked in nearly 100% of the tests in practice), resulting in a total of $72 \times 1m05s + 30.22s$, *i.e.* 1 hour 18 minutes instead of the previous 2 hours 20 minutes.

6 Countermeasures

In this section, we describe different countermeasures to protect an implementation against the CA presented in Section 3.

6.1 Blind Before Splitting

Our first proposition consists in avoiding the possibility to inject the same fault during several signature computations. To do so, we deport the blinding of the input message m before executing the two exponentiations modulo p and q:

$$m' = m + k_0 N \bmod k_1 N \ , \tag{19}$$

with k_0 and k_1 two n-bit random values generated at each algorithm execution (n being typically 64). Hence $S'_p = m'^{d_p} \bmod k_2 p$ and $S'_q = m'^{d_q} \bmod k_3 q$.

This countermeasure prevents an attacker from injecting always the same error during the signature computation. Indeed if the fault is injected on m at the very beginning of one exponentiation, then the corresponding error cannot be fixed due to the blinding injected by Rel. (19).

Moreover, if the fault is injected when the message m is manipulated during (19), then the error ε impacts the computation of both S'_p and S'_q, leading to inexploitable faulty outputs.

Such a countermeasure induces a small overhead in terms of memory space since m' must be kept in memory during the first exponentiation but the execution time remains the same.

6.2 Verification Blinding

Our second countermeasure aims at annihilating the second hypothesis of our attack: a predictive variable is manipulated in plain during the verification. To do so, we inject a $\lceil \log_2(N) \rceil$-bit random r before performing the final reduction with N, *cf.* Rel. (20). Therefore, each and every variable manipulated during the verification is blinded.

$$((\widetilde{S}^e + r - m) \bmod k_1 N) \bmod N \stackrel{?}{=} r \ . \tag{20}$$

One may note that the final comparison should be performed securely with regards to the attack described in [40] since information on $\varepsilon q i_q$ could leak if such a comparison was performed through a substraction.

The cost of such a countermeasure is negligible since it mainly consists in generating a $\lceil \log_2(N) \rceil$-bit random variable.

7 Conclusion

This paper introduces a new Combined Attack on CRT-RSA. Even if a secure implementation does not return the faulty signature when the computation is disturbed, we show how to combine FI with SCA during the verification process to obtain information on the faulty signature. Such information allows us to factorize the public modulus and thus to recover the whole private key. We also show that Coppersmith's methods to solve univariate and bivariate modular polynomial equations can be used to significantly reduce the complexity of our new attack. Finally, we provide simulations to confirm the efficiency of our method and we present two countermeasures which have a very small penalty on the performance of the algorithm.

Our main objective was to prove that stacking several countermeasures does not provide global security despite addressing each and every attack separately. Therefore, the main consequence of this paper is that fault injection countermeasures must also be designed to resist SCA and vice versa.

Acknowledgements. The authors would like to thank Emmanuelle Dottax for her useful comments on the preliminary version of this article. We would also like to thank Jean-Sébastien Coron and the anonymous reviewers of PKC'13 for their valuable comments and suggestions.

References

1. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In CRYPTO '96. Vol. 1109 of LNCS, Springer (1996) 104–113
2. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In CRYPTO '99. Vol. 1666 of LNCS, Springer (1999) 388–397
3. Quisquater, J.J., Samyde, D.: A New Tool for Non-intrusive Analysis of Smart Cards Based on Electro-magnetic Emissions, the SEMA and DEMA Methods. Presented at EUROCRYPT '00 Rump Session (2000)
4. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks – Revealing the Secrets of Smartcards. Springer (2007)
5. Giraud, C., Thiebeauld, H.: A Survey on Fault Attacks. In CARDIS '04, Kluwer Academic Publishers (2004) 159–176
6. Bellcore: New Threat Model Breaks Crypto Codes. Press Release (1996)
7. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystem. In CRYPTO '97. Vol. 1294 of LNCS, Springer (1997) 513–525
8. Bao, F., Deng, R., Han, Y., Jeng, A., Narasimhalu, A.D., Ngair, T.H.: Breaking Public Key Cryptosystems an Tamper Resistance Devices in the Presence of Transient Fault. In 5th Security Protocols Workshop. Vol. 1361 of LNCS, Springer (1997) 115–124
9. Joye, M., Tunstall, M.: Fault Analysis in Cryptography. Information Security and Cryptography. Springer (2012)
10. Amiel, F., Feix, B., Marcel, L., Villegas, K.: Passive and Active Combined Attacks – Combining Fault Attacks and Side Channel Analysis –. In FDTC '07, IEEE Computer Society (2007) 92–99
11. Fang, J., Gierlichs, B., Vercauteren, F.: To Infinity and Beyond: Combined Attack on ECC Using Points of Low Order. In CHES '11. Vol. 6917 of LNCS, Springer (2011) 143–159
12. Roche, T., Lomné, V., Khalfallah, K.: Combined Fault and Side-Channel Attack on Protected Implementations of AES. In CARDIS '11. LNCS, Springer (2011) 152–169
13. Robisson, B., Manet, P.: Differential Behavioral Analysis. In CHES '07. Vol. 4727 of LNCS, Springer (2007) 413–426
14. Rivest, R., Shamir, A., Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Communications of the ACM **21** (1978) 120–126
15. Coppersmith, D.: Finding a small root of a univariate modular equation. In EUROCRYPT '96. Vol. 1070 of LNCS, Springer (1996) 155–165
16. Coppersmith, D.: Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. Journal of Cryptology **10** (1997) 233–260
17. Couvreur, C., Quisquater, J.J.: Fast Decipherment Algorithm for RSA Public-Key Cryptosystem. Electronics Letters **18** (1982) 905–907
18. Garner, H.: The Residue Number System. IRE Transactions on Electronic Computers **8** (1959) 140–147
19. Joye, M., Yen, S.M.: The Montgomery Powering Ladder. In CHES '02. Vol. 2523 of LNCS, Springer (2002) 291–302

20. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Square Always Exponentiation. In INDOCRYPT '11. Vol. 7107 of LNCS, Springer (2011) 40–57
21. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In CHES '04. Vol. 3156 of LNCS, Springer (2004) 16–29
22. Amiel, F., Feix, B., Villegas, K.: Power Analysis for Secret Recovering and Reverse Engineering of Public Key Algorithms. In SAC '07. LNCS, Springer (2007) 110–125
23. Coppersmith, D.: Finding a small root of a bivariate integer equation; factoring with high bits known. In EUROCRYPT '96. Vol. 1070 of LNCS, Springer (1996) 178–189
24. Boneh, D., DeMillo, R., Lipton, R.: On the Importance of Checking Cryptographic Protocols for Faults. In EUROCRYPT '97. Vol. 1233 of LNCS, Springer (1997) 37–51
25. Lenstra, A.: Memo on RSA Signature Generation in the Presence of Faults. Manuscript (1996)
26. Blömer, J., Otto, M., Seifert, J.P.: A New RSA-CRT Algorithm Secure against Bellcore Attacks. In CCS'03 ACM Conference, ACM Press (2003) 311–320
27. Giraud, C.: An RSA Implementation Resistant to Fault Attacks and to Simple Power Analysis. IEEE Transactions on Computers **55** (2006) 1116–1120
28. Vigilant, D.: RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks. In CHES '08. Vol. 5154 of LNCS, Springer (2008) 130–145
29. Rivain, M.: Securing RSA against Fault Analysis by Double Addition Chain Exponentiation. In CT-RSA '09. Vol. 5473 of LNCS, Springer (2009) 459–480
30. Coron, J.S., Joux, A., Kizhvatov, I., Naccache, D., Paillier, P.: Fault Attacks on RSA Signatures with Partially Unknown Messages. In CHES '09. Vol. 5747 of LNCS, Springer (2009) 444–456
31. Coron, J.S., Naccache, D., Tibouchi, M.: Fault Attacks against EMV Signatures. In CT-RSA '10. Vol. 5985 of LNCS, Springer (2010) 208–220
32. Brier, E., Nguyen, D.N.P., Tibouchi, M.: Modulus Fault Attack against RSA-CRT Signatures. In CHES '11. Vol. 6917 of LNCS, Springer (2011) 192–206
33. Witteman, M., van Woudenberg, J., Menarini, F.: Defeating RSA Multiply-Always and Message Blinding Countermeasures. In CT-RSA '11. Vol. 6558 of LNCS, Springer (2011) 77–88
34. Dupaquis, V., Venelli, A.: Redundant Modular Reduction Algorithms. In CARDIS '11. LNCS, Springer (2011) 102–114
35. Dottax, E., Giraud, C., Rivain, M., Sierra, Y.: On Second-Order Fault Analysis Resistance for CRT-RSA Implementations. In WISTP '09. Vol. 5746 of LNCS, Springer (2009) 68–83
36. Blomer, J., May, A.: A Tool Kit for Finding Small Roots of Bivariate Polynomials over the Integers. In EUROCRYPT '05. Vol. 3494 of LNCS, Springer (2005) 251–267
37. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. J. Symbolic Comput. **24** (1997) 235–265
38. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In 6th IMA Int. Conf. Vol. 1355, Springer (1997) 131 – 142
39. Jochemsz, E., May, A.: A strategy for finding roots of multivariate polynomials with new applications in attacking rsa variants. In ASIACRYPT '06. Vol. 4284 of LNCS, Springer (2006) 267–282
40. Lomne, V., Roche, T., Thillard, A.: On the Need of Randomness in Fault Attack Countermeasures – Application to AES. In FDTC '12, IEEE Computer Society (2012) 85–94