(In)security Against Fault Injection Attacks for CRT-RSA Implementations

Alexandre Berzati and Cécile Canovas
CEA-LETI/MINATEC
17 rue des Martyrs
38054 Grenoble Cedex 9, France
firstname.familyname@cea.fr

Louis Goubin
Versailles Saint-Quentin-en-Yvelines University
45 Avenue des Etats-Unis
78035 Versailles Cedex, France
Louis.Goubin@prism.uvsq.fr

Abstract

Since its invention in 1977, the celebrated RSA primitive has remained unbroken from a mathematical point of view, and has been widely used to build provably secure encryption or signature protocols.

However, the introduction in 1996 of a new model of attacks – based on fault injections – by Boneh, deMillo and Lipton suggests the use of specific countermeasures to obtain a secure RSA implementation. In the special case of CRT implementations, many protections have been proposed and most of them have been proven insufficient to ensure resistance against DFA. This has motivated the introduction of the "infective computation" concept [18], leading to the only two implementations not broken up to now: Ciet-Joye [10] and Giraud [8].

In the present paper, we show that the Ciet-Joye method proposed in FDTC'2005 does not completely prevent fault injection attacks: for a CRT-RSA with a 1024-bit modulus, we show that 13 faulty signatures are enough to recover the secret exponent with a probability greater than 50%, which can be improved to 99% with 83 faulty signatures.

Keywords: *CRT-RSA, fault attacks, countermeasures, Wagner's attack.*

1. Introduction

The RSA [14] algorithm has been the most widely used public-key cryptosystem for many years. The Chinese Remainder Theorem applied to RSA has considerably increased its efficiency and takes a part of such a success. At the end of the nineties, Bellcore researchers first showed that such an implementation can be endangered by fault injections [5]. This paper not only introduced a new class of side-channel attacks named Differential Fault Analysis but also highlighted its powerful applications to Public-Key based protocols. And CRT-RSA became a preferential target for these attacks. A series of attacks and countermeasures followed. The state of the art shows that only two implementations of CRT-RSA actually remain DFA-resistant: the Ciet & Joye algorithm [10] presented at *FDTC'05* and the patented C. Giraud's SPA/DFA-resistant algorithm [8, 9]. Their respective security was analyzed taking into account the published attacks. However, no formal proof has ever been done to warranty their security.

The purpose of our paper is to show that the Ciet & Joye algorithm is not a really secure implementation of CRT-RSA. We prove that, under a practicable fault model, it is possible to recover the private exponent by injecting a fault before the CRT recombination step.

Our paper is organised as follows. The next section presents published attacks against CRT based RSA implementations and associated countermeasures. Then we will detail the Ciet & Joye algorithm. Finally, we will present our attack against this algorithm and its performance analysis.

2. Previous work

2.1. CRT Based RSA

Let N, the public modulus, be the product of two large prime numbers p and q. The length of N is denoted by n. Let e be the public exponent, coprime to $\varphi(N) = (p-1) \cdot (q-1)$, where $\varphi(\cdot)$ denotes Euler's totient function. The public key exponent e is linked to the private exponent d by the equation $e \cdot d \equiv 1 \bmod \varphi(N)$. The signature on message m is given by:

$$S = \dot{m}^d \bmod N \tag{1}$$

where $\dot{m} = \mu(m)$ for some hash and/or deterministic padding function μ. The improvement brought by the Chinese Remainder Theorem concerns the computation of the modular exponentiation. In CRT mode, instead of computing the d-th exponentiation, two half exponentiations by $d_p \equiv d \bmod (p-1)$ and $d_q \equiv d \bmod (q-1)$ are done. Let

$i_q \equiv q^{-1} \bmod p$ be the inverse of q in $\mathbb{Z}/p\mathbb{Z}$, the signature S is calculated with Garner's algorithm, denoted by CRT:

$$\begin{aligned} S &= \text{CRT}(S_p, S_q) \\ &= S_q + q(i_q(S_p - S_q) \bmod p) \qquad (2) \\ \text{with} \quad &\begin{cases} S_p \equiv \dot{m}^{d_p} \bmod p, \\ S_q \equiv \dot{m}^{d_q} \bmod q. \end{cases} \end{aligned}$$

This trick speeds up the computation by computing two half exponentiations modulo a $n/2$-bit number instead of an exponentiation modulo a n-bit number. Because of the multiplication's quadratic complexity, the CRT computation is four times faster than the standard one. In both modes, the signature S is validated by checking if:

$$S^e \equiv \dot{m} \bmod N \qquad (3)$$

2.2. DFA & Countermeasures on CRT-RSA

Bellcore's attack. In 1996, Bellcore researchers introduced the Differential Fault Analysis by attacking the CRT based implementation of RSA.They showed in [5, 6] that if an error occurs while computing one of the two half exponentiations (*i.e.* S_p or S_q but not both) then, from the faulty signature $\hat{S}$ and the correct one S, it is possible to factor N. Indeed, assume that an error was provoked during the computation of S_p resulting in a faulty value $\hat{S}_p$, then the signature $\hat{S} = \text{CRT}(\hat{S}_p, S_q)$ is faulty too. Moreover, $\hat{S} \equiv S \bmod q$ but $\hat{S} \not\equiv S \bmod p$. So, only $q \mid (\hat{S} - S)$ and:

$$q = gcd((\hat{S} - S) \bmod N, N) \qquad (4)$$

This result was reduced to the mere knowledge of the faulty signature by A. Lenstra [11], noticing that $\hat{S}^e \equiv \dot{m} \bmod q$ but $\hat{S}^e \not\equiv \dot{m} \bmod p$:

$$q = gcd((\hat{S}^e - \dot{m}) \bmod N, N) \qquad (5)$$

Shamir's trick and optimizations. In [15], A. Shamir presents a method to defeat DFA by randomizing the computation of S_p and S_q and adding a checking test before returning the signature. Let r be a κ-bit random value. The cryptographic device computes:

$$\begin{cases} S_{rp} \equiv \dot{m}^d \bmod (r \cdot p) \\ S_{rq} \equiv \dot{m}^d \bmod (r \cdot q) \end{cases} \qquad (6)$$

Then, we check that no error occurs during the computation of the two half exponentiations:

1. **If** $S_{rp} \equiv S_{rq} \bmod r$, **return** $S = \text{CRT}(S_{rp}, S_{rq})$,
2. **Else**, **return** `Error detected`.

The main drawback of this method is that it requires the knowledge of d whereas, on a real cryptographic device that implements CRT-RSA, only $d_p \equiv d \bmod (p-1)$ and $d_q \equiv d \bmod (q-1)$ are available. That is why M. Joye, P. Paillier and S. Yen have proposed in [12] an optimization of Shamir's countermeasure. Let r_1 and r_2 be two κ-bit random integers. The device computes:

$$\begin{aligned} &S_p^* \equiv \dot{m}^{d_p} \bmod (r_1 \cdot p),\ S_1 \equiv \dot{m}^{d_p} \bmod \varphi(r_1) \bmod r_1 \\ &S_q^* \equiv \dot{m}^{d_q} \bmod (r_2 \cdot q),\ S_2 \equiv \dot{m}^{d_q} \bmod \varphi(r_2) \bmod r_2 \end{aligned} \qquad (7)$$

Both half exponentiations are checked separately before the CRT recombination:

1. **If** $S_1 \equiv S_p^* \bmod r_1$ **and** $S_2 \equiv S_q^* \bmod r_2$, **return** $S = \text{CRT}(S_p^*, S_q^*)$,
2. **Else**, **return** `Error detected`.

Thus, this optimization is resistant to fault injection during the two half exponentiations and only needs classical CRT parameters.

DFA on the recombination step. Although the previously presented countermeasures protects CRT based RSA against perturbation of the half exponentiations, they can be endangered by a perturbation of the CRT recombination step. This security flaw was exploited by C. Aumüller, P. Bier, W. Fischer, P. Hofreiter and J-P. Seifert in [1]. Indeed, if the value of S_{rp}, S_{rq} or i_{rq} are modified during the recombination step, the fault is not detected by Shamir's method. Moreover, from the faulty signature $\hat{S}$, the attacker can factor N by computing $gcd((\hat{S}^e - \dot{m}) \bmod N, N)$.

A countermeasure against transient faults is also presented in [1]. But, S-M. Yen, S. Moon and J. Ha showed in [20] that this countermeasure introduces another vulnerability. The modified CRT-RSA algorithm can be attacked by injecting a permanent fault during the CRT recombination.

Infective computation. The concept of infective computation was introduced in 2001 by S-M. Yen, S. Kim, S. Lim and S. Moon [18]. The idea of infective computation is to ensure that both half exponentiations are faulty when a fault occurs during one of them. That way, $\hat{S} \not\equiv S \bmod p \not\equiv S \bmod q$ and the fault can not be exploited by the gcd. The authors suggested this countermeasure after noticing that decisional checks can be avoided by fault injection. Two protocols using infective computation where proposed in [18] and broken later in [17, 19] by S-M. Yen and D. Kim. In spite of this, the concept of infective computation was reused later in the *BOS* scheme [4] and in the Ciet & Joye algorithm [10].

***BOS* scheme.** Given the flaws of previously described algorithms, J. Blömer, M. Otto and J-P. Seifert proposed in [4] a variant of Shamir's trick. The modification of the CRT recombination and the use of infective computation have made this algorithm immune against attacks previously published. The algorithm works as follow: let t_1 and t_2 be two κ-bit random integers carefully chosen[1]:

1. The device first computes:

$$t_1 \cdot p,\ t_2 \cdot q,\ t_1 \cdot t_2 \cdot N,$$

$$d_1 \equiv d \bmod \varphi(t_1 \cdot p),\ e_1 \equiv {d_1}^{-1} \bmod \varphi(t_1 \cdot p),$$
$$d_2 \equiv d \bmod \varphi(t_2 \cdot q),\ e_2 \equiv {d_2}^{-1} \bmod \varphi(t_2 \cdot q),$$

2. Then, half exponentiations and CRT recombination is computed:

$$S^* = \text{CRT}(S_p^*, S_q^*) \bmod (t_1 \cdot t_2 \cdot N), \quad (8)$$
$$\text{where} \quad \begin{cases} S_p^* \equiv \dot{m}^{d_1} \bmod (t_1.p) \\ S_q^* \equiv \dot{m}^{d_2} \bmod (t_2.q) \end{cases}$$

3. The returned signature S is defined as:

$$S \equiv (S^*)^{c_1 c_2} \bmod N \quad (9)$$
$$\text{with} \quad \begin{cases} c_1 \equiv (\dot{m} - (S^*)^{e_1} + 1) \bmod (t_1.p) \\ c_2 \equiv (\dot{m} - (S^*)^{e_2} + 1) \bmod (t_2.q) \end{cases}$$

In the case of a correct execution, $c_1 = c_2 = 1$, so $S = S^* \bmod N$.

At first sight, this algorithm requires the knowledge of the private exponent d which is not often the case on a cryptographic device. From a security point of view, the use of the result of the CRT recombination – S^* – in the third step makes it immune against attacks previously described.

Wagner's attack. D. Wagner found a way to exploit a vulnerability in *BOS* scheme [16]. The attack consists in injecting a random transient byte fault in $\dot{m}$, just before computing S_p^*, in a way that subsequent accesses to $\dot{m}$ will be *error-free*. The returned signature will be faulty:

$$\hat{S} \equiv (\hat{S^*})^{\hat{c_1}} \bmod N$$

and $\hat{S^*} \equiv (S^*) \bmod q$ but $\hat{S^*} \not\equiv (S^*) \bmod p$. The attacker tries to find the value of $\hat{c_1}$ in order to factor N by computing $gcd((\hat{S^e} - \dot{m}^{\hat{c_1}}) \bmod N, N)$. In [16], D. Wagner claims that the success probability of this attack depends on the byte location of the fault. For a 1024-bit RSA and $\kappa = 80$ bits, the success probability is around 4%. This attack has been discussed by J. Blömer and M. Otto leading to a modified revision of the *BOS* scheme [3].

[1] According to [4], t_1 and t_2 must satisfy: (i) $gcd(t_1, t_2) = 1$, (ii) $gcd(d, \varphi(t_1)) = gcd(d, \varphi(t_2)) = 1$, (iii) $t_1 = t_2 = 3 \bmod 4$ (iv) t_1 and t_2 are square-free, (v) $t_2 \nmid (t_1 \cdot p) \cdot [(t_1 \cdot p)^{-1} \bmod (t_2 \cdot q)]$

3. Ciet & Joye algorithm

This algorithm was first presented in [10] as a countermeasure for D. Wagner's attack against *BOS* scheme [16]. M. Ciet and M. Joye claimed that this algorithm was secure against all previously published attacks against Chinese remaindering based implementations of RSA. But its security was not formally proved. C. Giraud highlighted, in his PhD thesis [9], that this algorithm was one of the two remaining secure implementations of CRT-RSA against fault based attacks. But, C. Kim and J-J. Quisquater [13] revealed a flaw against double-fault attack and fixed it with a *cost-free* method.

This algorithm is inspired by Shamir's countermeasure [15], improved to avoid decisional tests by using infective computation methodology [18]. The algorithm works as follows: let r_1 and r_2 be two κ-bit carefully chosen random integers and r_3 a l-bit random integer:

1. The cryptographic device first computes:
   (a) $p^* = r_1 \cdot p$
   (b) $q^* = r_2 \cdot q$
   (c) $i_{q^*} = (q^*)^{-1} \bmod p^*$
   (d) $N = p \cdot q$

2. Then, it computes the CRT parameters:

   (a) $S_{p^*} \equiv \dot{m}^{d_p} \bmod p^*, s_2 \equiv \dot{m}^{d_q} \bmod \varphi(r_2) \bmod r_2$
   (b) $S_{q^*} \equiv \dot{m}^{d_q} \bmod q^*, s_1 \equiv \dot{m}^{d_p} \bmod \varphi(r_1) \bmod r_1$ (10)

   The order of computation has to be carefully respected in order to protect the algorithm against injection of permanent random faults on $\dot{m}$.

3. The CRT recombination is applied to get S^*:

$$S^* = \text{CRT}(S_{p^*}, S_{q^*})$$

4. Finally, the algorithm returns the signature S such as:

$$S \equiv (S^*)^{\gamma} \bmod N \begin{cases} c_1 \equiv (S^* - s_1 + 1) \bmod r_1 \\ c_2 \equiv (S^* - s_2 + 1) \bmod r_2 \\ \gamma = \lfloor \dfrac{(r_3 \cdot c_1 + (2^l - r_3) \cdot c_2)}{2^l} \rfloor \end{cases}$$

According to [10], the secret parameters of the algorithm (p, q, d_p, d_q) and the values computed during the first step are assumed to be *error-free*. It means that these values are implicitly protected against fault injection.
Moreover, several parts of this algorithm can be adapted to the cryptographic device architecture. The length of random values (l and κ) can be tuned to suit the length of crypto registers. Finally, one can notice that γ is a κ-bit value.

4. Attack principle

4.1. Fault model

The principle of the attack is inspired by Wagner's Attack on *BOS* scheme [16]. It consists in inserting any transient fault on an unknown byte of S_{p^*}. According to [10], the value of S_{p^*} is not protected. Then, our fault model can be considered. We have chosen such a model because it has already been successfully applied in the smart card context [2, 3, 7]. The byte modification induced by the attacker affects the computation of S, S^*, γ and c_1. It can be seen as the addition of an error value ε to the correct one S_{p^*}:

$$\hat{S_{p^*}} = S_{p^*} \oplus \varepsilon \tag{11}$$

$\varepsilon = R_8 \cdot 2^{8i}$ where R_8 is a random byte value and $i \in [\![0; \frac{(n/2)+\kappa}{8} - 1]\!]$ is the location of the byte modified by the perturbation. These two values result from the effects of the perturbation and are – a priori – not known by the attacker.

4.2. Faulty execution

The attacker corrupts the value of S_p^* between the first partial exponentiation and the CRT recombination. This faulty value of a variable x is noted $\hat{x}$.

1. The fault first infects the CRT recombination:

$$\hat{S^*} = CRT(\hat{S_{p^*}}, S_{q^*})$$

2. The output of that faulty execution is:

$$\hat{S} \equiv (\hat{S^*})^{\hat{\gamma}} \bmod N$$

$$\text{where} \quad \begin{cases} \hat{c_1} \equiv (\hat{S^*} - s_1 + 1) \bmod r_1 \\ c_2 \equiv (\hat{S^*} - s_2 + 1) \bmod r_2 \\ \hat{\gamma} = \lfloor \dfrac{(r_3.\hat{c_1} + (2^l - r_3).c_2)}{2^l} \rfloor \end{cases}$$

and r_3 stands for a l-bit random value

4.3. Cryptanalysis

The two half exponentiations of (10) are expressed modulo $q^* = r_2 \cdot q$ and $p^* = r_1 \cdot p$. So, the induced asymmetry can be explained as $\hat{S^*} \not\equiv S^* \bmod r_1$ and $\hat{S^*} \equiv S^* \bmod r_2$. Furthermore, (10) implies that $\hat{S^*} \equiv S^* \bmod q$ but $\hat{S^*} \not\equiv S^* \bmod p$. So, $\hat{S^*}^e \equiv \dot{m} \bmod q$ but $\hat{S^*} \not\equiv \dot{m} \bmod p$ and, by using the final result of the faulty execution $\hat{S} \equiv (\hat{S^*})^{\hat{\gamma}} \bmod N$, we also have:

$$\begin{cases} \hat{S}^e \equiv \dot{m}^{\hat{\gamma}} \bmod q \\ \hat{S}^e \not\equiv \dot{m}^{\hat{\gamma}} \bmod p \end{cases} \tag{12}$$

If the value of $\hat{\gamma}$ is known by the attacker, it's possible for him to factor N by computing $gcd((\hat{S}^e - \dot{m}^{\hat{\gamma}}) \bmod N, N)$. In this case, the problem of factoring N is based on guessing the value of $\hat{\gamma}$.

4.4. How to guess $\hat{\gamma}$?

By observing its formula, one can notice that $\hat{\gamma}$ depends on three parameters that are c_1, c_2 and r_3. The application of our fault model make us recover the value of the checks: c_1 and c_2. Indeed, because of the chosen fault model, only the result of the first half exponentiation – $\hat{S_p^*}$ – is faulty. As a consequence, using (10):

$$\begin{cases} c_1 \equiv (\hat{S^*} - s_1 + 1) \bmod r_1 \equiv (1 + \varepsilon) \bmod r_1 \\ c_2 \equiv (\hat{S^*} - s_2 + 1) \bmod r_2 \equiv 1 \bmod r_2 \end{cases} \tag{13}$$

The knowledge of ε makes us directly deduce the value of c_1 if no modular reduction occurs during its computation. It needs that $0 \leq \varepsilon < (r_1 - 1)$. Assuming that the previous condition is satisfied, we have $c_1 = 1 + \varepsilon$. Hence, the value of $\hat{\gamma}$ only depends on the error ε and the random value r_3:

$$\hat{\gamma} = \lfloor \frac{(r_3 \cdot c_1 + (2^l - r_3) \cdot c_2)}{2^l} \rfloor = 1 + \lfloor \frac{r_3 \cdot \varepsilon}{2^l} \rfloor \tag{14}$$

At first sight $\hat{\gamma}$ remains a κ-bit random value, but if we apply our fault model and rewrite ε:

$$\hat{\gamma} = 1 + \lfloor r_3 \cdot A_8 \cdot 2^{8i-l} \rfloor \tag{15}$$

In equation (15), i is unknown but, because of the previous supposition, we can assume that the size of epsilon has to be smaller than the size of r_1 :

$$2^{8(i+1)} < 2^{\kappa} \tag{16}$$

So, we have $i \in [\![0; \frac{\kappa}{8} - 1]\!]$. Now, we are facing two cases that depends on the sign of $(l - \kappa)$:

1. $(l - \kappa) > 0$. Then $\forall i \in [\![0; \frac{\kappa}{8} - 1]\!], (8i - l) < 0$. It means that $r_3 \cdot A_8$, which is a $(l+8)$-bit random value, will be shifted to the right. Hence, the value of $\hat{\gamma}$ will be a $(l+8)+(8i-l) = 8(i+1)$-bit random value that is located on the least significant bits: $\hat{\gamma} = A_{(8(i+1),0)}$.

2. $(l - \kappa) < 0$. Then, if $(8i - l) < 0$, $\hat{\gamma}$ takes the previously described form. Otherwise, for $(8i - l) > 0$, $\hat{\gamma}$ remains a $8(i + 1)$ random value, but located in the most significant bits: $\hat{\gamma} = A_{(\kappa,\kappa-8(i+1))} + 1$. In this case, we can advantageously notice that $\hat{\gamma}$ is an odd value.

That way, we have shown that the form of $\hat{\gamma}$ depends on the injected error. The location of the modified byte influences the length of the random value that composes $\hat{\gamma}$. Obviously,

the value of $\hat{\gamma}$ remains unknown, but, if i is small enough (*i.e.* $i \leq 4$), it can be recovered by a brute force search. Given a faulty signature $\hat{S}$, the attacker will try to factor N by testing if $gcd((\hat{S}^e - \dot{m}^{\hat{\gamma}_j}) \bmod N, N)$ returns a value different from 1, for one of the $\hat{\gamma}_j$ candidates. If it fails, the attacker has to obtain other faulty signatures until the gcd test works.

4.5. Attack algorithm

We detail here the algorithm which implements the fault injection attack described above.

Algorithm 1. DFA against Ciet & Joye algorithm

INPUT: $\dot{m}$, N, κ, l, $\hat{S}$, B_f the brute force length
OUTPUT: a factor of N

//Whatever the sign of $(l - \kappa)$, LSB case is treated
for $\hat{\gamma}_j$ **from** 1 **upto** 2^{B_f}
  fact := $gcd((\hat{S}^e - \dot{m}^{\hat{\gamma}_j}) \bmod N, N)$;
  if (fact $\neq$ 1)
    return fact;
  endif
endfor
//If $\hat{\gamma}$ is still unknown and $l < \kappa$, we search on MSB
if $(l < \kappa)$
  for $\hat{\gamma}_j$ **from** 1 **upto** 2^{B_f}
    //Bits are shifted to the MSB
    $\hat{\gamma}_j := (\hat{\gamma}_j \ll (\kappa - B_f - 1)) + 1$
    fact := $gcd((\hat{S}^e - \dot{m}^{\hat{\gamma}_j}) \bmod N, N)$;
    if (fact $\neq$ 1)
      return fact;
    endif
    //No γ candidate suits
    return -1;
  endfor
else
  //The algorithm finishes without factoring N
  return -1;
endif

4.6. Performance

To calculate the performance of our attack, we implicitly assume that the random values induced by the fault (*i.e.* the location i and the byte value A_8) are uniformly distributed over their respective spaces. This is not exactly the case in practice but this is a quite good assumption.

As mentioned in the previous parts, the attack works if two conditions on the fault are respected:

1. No modular reduction must occur at the first check, so $(1 + \epsilon) < r_1$. According to (16) the probability of this event to occur can be approximate by:

$$\Pr[(1 + \epsilon) < r_1] \approx \Pr[8(i + 1) < \kappa] \quad (17)$$

Knowing that $i \in [\![0; \frac{(n/2)+\kappa}{8} - 1]\!]$,

$$\Pr[8(i + 1) < \kappa] \approx \frac{2 \cdot (\kappa - 8)}{n + 2\kappa} \quad (18)$$

2. $\hat{\gamma}$ must be recoverable by brute force search. It means that the size of $\hat{\gamma}$ must satisfy $8(i + 1) < B_f$, where B_f stands for a computable limit for the searched bits of $\hat{\gamma}$ (*i.e.* $B_f \leq 40$ bits). The probability of this event is denoted by $\Pr[8(i + 1) < B_f]$. But, $8(i + 1) < B_f$ is equivalent to $i < (\frac{B_f}{8} - 1)$. We also know that $i \in [\![0; \frac{(n/2)+\kappa}{8} - 1]\!]$, as a result:

$$\Pr[8(i + 1) < B_f] = \frac{2 \cdot (B_f - 8)}{n + 2\kappa} \quad (19)$$

One can notice that the value of ε depends on i. So, these probabilities are not independent. As a consequence, given a faulty signature, the success probability of our attack is defined by:

$$\begin{aligned} \Pr(success) &\approx \Pr[8(i + 1) < \kappa \text{ and } 8(i + 1) < B_f] \\ &\approx \Pr[8(i + 1) < \min(\kappa, B_f)] \end{aligned} \quad (20)$$

For a CRT-RSA with parameters $n = 1024$ bits , $\kappa = 80$ bits and $B_f = 40$ bits, the success probability of our attack is about 5.4%. This result shows that our attack against Ciet & Joye algorithm is more efficient than Wagner's one against a weaker CRT-RSA implementation. Furthermore, if one wants to lengthen the random parameter κ (*i.e.* $\kappa > B_f$), the success probability, will always be at least $2 \cdot (B_f - 8)/(n + 2\kappa)$. On the other hand, if one chooses a short random length for κ (*i.e.* $\kappa < B_f$), the success probability of the presented attack will be minimized. But, noticing that γ is by definition a κ-bit value, its brute force search will be dramatically shorten without considering any fault model.

In terms of fault count, 13 faulty signatures are enough to recover the secret exponent with a probability greater than 50%. But with 83 faulty signatures, the success probability is about 99%.

This algorithm has been successfully implemented on a PC using the GMP Library and simulating the chosen fault model. The results obtained have shown that the public modulus N can be factor by applying our method.

4.7. Discussion

By analysing our proposed attack, one can notice that the critical steps are the check computations: c_1 and c_2.

Indeed, we have shown that a fault is usable if and only if no modular reduction occurs during the targeted check's computation. In this case one of the check values directly depends on the fault's value (see equation (13)). As a consequence, forcing the modular reduction to occur can be an idea to efficiently protect the Ciet & Joye algorithm against our presented attack. This can be achieved by modifying the check computations as follows. Let α be a κ-bit random value such as $\alpha > r_1$ and $\alpha > r_2$:

$$\begin{cases} c_1 \equiv [\alpha(S^* - s_1) + 1] \bmod r_1 \\ c_2 \equiv [\alpha(S^* - s_2) + 1] \bmod r_2 \end{cases} \quad (21)$$

That way, if no error occurs during the computation, c_1 and c_2 are still equal to one. But, in a case of a faulty computation, we have $\forall \varepsilon > 0$, $\alpha \cdot \varepsilon > c_1$ and $\alpha \cdot \varepsilon > c_2$. So, a modular reduction will be done during the computation of the checks.

5. Conclusion

In the present paper we describe a fault injection attack against the CRT-RSA implementation proposed by M. Ciet and M. Joye in FDTC'2005. We provide a detailed theoretical analysis of the attack, and implemented it on a PC using the GMP Library. Our fault model is very simple and very realistic. For a CRT-RSA with a 1024-bit modulus, we show that under this fault model, 13 faulty signatures are enough to recover the secret exponent with a probability greater than 50%, which can be improved to 99% with 83 faulty signatures.

As a consequence, the only patent-free method to implement CRT-RSA, with the expected DFA-resistance property, has been shown to still have vulnerabilities against fault injection attacks. We finally suggest a variant of the Ciet & Joye method to ensure fault injection resistance. A formal proof of this resistance property is still a challenging open problem.

References

[1] C. Aumüler, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert. Fault Attack on RSA with CRT : Concrete Results and Practical Countermeasures. In B. K. Jr., Ç.K. Koç, and C. Parr, editors, *Cryptographic Hardware and Embedded Systems (CHES 2002)*, volume 2523 of *Lecture Notes in Computer Science*, pages 260–275. Springer, 2002.

[2] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan. The Sorcerer's Apprentice Guide to Fault Attacks. Cryptology ePrint Archive, Report 2004/100, 2004.

[3] J. Blömer and M. Otto. Wagner's Attack on a secure CRT-RSA Algorithm Reconsidered. In L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography (FDTC 2006)*, volume 4236 of *Lecture Notes in Computer Science*, pages 13–23. Springer-Verlag, 2006.

[4] J. Blömer, M. Otto, and J.-P. Seifert. A New CRT-RSA Algorithm Secure Against Bellcore Attack. In *ACM Conference on Computer and Communication Security (CCS 2003)*, pages 311–320. ACM Press, 2003.

[5] D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In W. Fumy, editor, *EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997.

[6] D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Eliminating Errors in Cryptographic Computations. *Journal of Cryptology*, 14(2):101–119, 2001.

[7] C. Giraud. DFA on AES. In V. Rijmen, H. Dobbertin, and A. Sowa, editors, *Fourth Conference on the Advanced Encryption Standard (AES4)*, volume 3373 of *Lecture Notes in Computer Science*, pages 27–41. Springer-Verlag, 2005.

[8] C. Giraud. Procédé de traitement de données impliquant une exponentiation modulaire et un dispositif associé, March 2005. Numéro de publication: FR0503083, WO2006103341.

[9] C. Giraud. *Attaques de Cryptosystèmes Embarqués et Contre-Mesures Associées*. PhD thesis, Université de Versailles Saint-Quentin, 2007.

[10] M. Joye and M. Ciet. Practical Fault Countermeasures for Chinese Remaindering Based RSA. In L. Breveglieri and I. Koren, editors, *Fault Diagnosis and Tolerance in Cryptography (FDTC 2005)*, pages 124–132, 2005.

[11] M. Joye, A. Lenstra, and J.-J. Quisquater. Chinese Remaindering Based Cryptosystems in the Presence of Faults. *Journal of Cryptology*, 12(4):241–245, 1999.

[12] M. Joye, P. Paillier, and S.-M. Yen. Secure Evaluation of Modular Functions. In R. Hwang and C. Wu, editors, *2001 International Workshop on Cryptology and Network Security*, pages 227–229, Taipei,Taiwan, 2001.

[13] C. Kim and J.-J. Quisquater. Fault Attacks for CRT Based RSA : New Attacks, New Results and New Countermeasures. In *Information Security Theory and Practices, Smart Cards, Mobile and Ubiquitus Computing Systems*, volume 4462 of *Lecture Notes in Computer Science*, pages 215–228. Springer-Verlag, 2007.

[14] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signature and Public-Key Cryptosystems. In *Communications of the ACM*, volume 21, pages 120–126, 1978.

[15] A. Shamir. Improved Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attacks. *Presented at the Rump Session of Eurocrypt'97*, 1997.

[16] D. Wagner. Cryptanalysis of a provably secure CRT-RSA algorithm. In *Proceedings of the 11th ACM Conference on Computer Security (CCS 2004)*, pages 92–97. ACM, 2004.

[17] S.-M. Yen and D. Kim. Cryptanalysis of Two Protocols for RSA with CRT Based on Fault Infection. In *Fault Diagnosis and Tolerance in Cryptography (FDTC 2004)*, pages 381–385, 2004. IEEE Computer Society.

[18] S.-M. Yen, D. Kim, S. Lim, and S. Moon. RSA Speedup with Residue Number System Immune Against Hardware

Fault Cryptanalysis. In K. Kim, editor, *Information Security and Cryptology (ISISC 2001)*, volume 2288 of *Lecture Notes in Computer Science*, pages 397–413. Springer-Verlag, 2001.

[19] S.-M. Yen, D. Kim, and S. Moon. Cryptanalysis of Two Protocols for RSA with CRT Based on Fault Infection. In L. Breveglieri, I. Koren, D. Naccache, and J.-P. Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography (FDTC 2006)*, volume 4236 of *Lecture Notes in Computer Science*, pages 53–61. Springer-Verlag, 2006.

[20] S.-M. Yen, S. Moon, and J.-C. Ha. Hardware Fault Attack on RSA with CRT Revisited. In C. Lim and P. Lee, editors, *Information Security and Cryptology (ICISC 2002)*, volume 2587 of *Lecture Notes in Computer Science*, pages 374–388. Springer, 2002.