

Fault Attacks and Countermeasures on Vigilant’s RSA-CRT Algorithm

Jean-Sébastien Coron^{*}, Christophe Giraud[†], Nicolas Morin[†], Gilles Piret[‡] and David Vigilant[§]

**Université du Luxembourg*
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg, Luxembourg
jean-sebastien.coron@uni.lu

Oberthur Technologies
Crypto and Security Group
[†]4, allée du Doyen Georges Brus, 33 600 Pessac, France
[‡]71-73, rue des Hautes Pâtures, 92 726 Nanterre, France
{c.giraud,n.morin,g.piret}@oberthur.com

[§]Gemalto Security Labs
6 rue de la Verrerie
F-92447 Meudon-sur-Seine, France
david.vigilant@gemalto.com

Abstract—At CHES 2008, Vigilant proposed an efficient way of implementing a CRT-RSA resistant against Fault Analysis. In this paper, we investigate the fault-resistance of this scheme and we show that it is not immune to fault injection. Indeed, we highlight two weaknesses which can lead an attacker to recover the whole private key by using only one faulty signature. We also suggest some modifications with a negligible cost to improve the fault-resistance of Vigilant’s scheme. Therefore the scheme including modifications remains suited to embedded device constraints.

Keywords-Fault Attacks, CRT-RSA.

I. INTRODUCTION

As well as being the first practical public-key cryptosystem published, RSA [19] has also been the most widely used for many years. One of the main particularities of RSA is that it can be used in both encryption and signature mode. Indeed, decryption and signature (resp. encryption and verification) operations are exactly the same.

Let us denote by N the RSA public modulus which is the product of two large prime numbers p and q . In the following we denote by e (resp. by d) the public (resp. private) exponent, the pair (e, d) satisfying the relationship $e \cdot d \equiv 1 \pmod{\text{lcm}(p-1, q-1)}$.

The standard way of computing an RSA signature S of a message m consists in a modular exponentiation with the private exponent: $S = m^d \pmod{N}$. The corresponding signature verification is done by comparing

the message m with the signature S raised to the power of the public exponent: $m \stackrel{?}{=} S^e \pmod{N}$.

In order to improve the performance, the signature is very often computed by using the so-called CRT-RSA which takes advantage of the CRT-recombination proposed by Garner in [10]. Let us denote by d_p (resp. d_q) the value $d \pmod{p-1}$ (resp. $d \pmod{q-1}$) and i_q the inverse of q modulo p . In the case of CRT-RSA, the public key is the pair (e, N) and the private key is the quintuplet (p, q, d_p, d_q, i_q) . To compute the signature in this case, the message is raised to the power of d_p modulo p then to the power of d_q modulo q . The corresponding results S_p and S_q are then combined by using Garner’s formula to obtain the signature: $S = S_q + q(i_q(S_p - S_q) \pmod{p})$.

If used exactly as described above, the RSA is subject to multiple attacks from a theoretical point of view. Indeed, it is possible under some assumptions to recover some information on the plaintext from the ciphertext or to forge fake signatures. To ensure its security, the RSA must be used according to a protocol which mainly consists in formatting the message. Examples of such protocols are the encryption protocol OAEP and the signature protocol PSS, both of them being proved secure and included in the standard PKCS #1 V2.1 [17].

From a practical point of view, RSA is also subject to many attacks if straightforwardly implemented. For instance, Simple Power Analysis, Differential Power Analysis [15] or Fault Analysis (FA) [4] can be used

to recover the RSA private key. To counteract these attacks, many countermeasures have been published over the last decade, e.g. [7], [12], [11].

This paper investigates the security of the countermeasure proposed by Vigilant at CHES 2008 [21] which is one of the most efficient methods published so far to prevent Fault Analysis on CRT-RSA. The rest of this paper is organized as follows. In Section II, we review known fault attacks and countermeasures on CRT-RSA. Section III presents in detail Vigilant’s scheme. Section IV highlights two weaknesses of Vigilant’s method. We then explain how such weaknesses can be used by an attacker to recover the private key by using only one faulty signature. Moreover, we show a practical application of one of these attacks on a smart card. Section V presents simple countermeasures to strengthen the original method against fault attacks. Finally, Section VI concludes this paper.

II. FAULT ATTACKS AND COUNTERMEASURES ON CRT-RSA

In 1996, Boneh, DeMillo and Lipton were the first to present a fault attack on RSA in both standard and CRT mode [4]. In the case of CRT-RSA, if a fault is induced during the computation of S_p then an erroneous value \tilde{S}_p is used during the CRT-recombination leading to an erroneous signature \tilde{S} . As $S \equiv S_p \pmod{p}$ and $S \equiv S_q \pmod{q}$, they noticed that $\tilde{S} \equiv S \pmod{q}$ but $\tilde{S} \not\equiv S \pmod{p}$. Therefore, if p does not divide $S - \tilde{S}$ (which is always the case in practice) then the secret parameter q can be easily obtained by computing the GCD of $S - \tilde{S}$ and N . The other secret parameters of the private key p , d_p , d_q and i_q can then be easily computed. The attack works similarly if a fault is induced during the computation of S_q .

A few months later, Lenstra improved this attack [16]: if a fault is induced during the computation of S_p then $\tilde{S}^e \equiv m \pmod{q}$ and $\tilde{S}^e \not\equiv m \pmod{p}$. Therefore, the secret parameter q can be obtained by computing the GCD of $\tilde{S}^e - m$ and N . By using this attack, only one execution of the cryptographic algorithm is required to recover the private key. This method is very useful when the attacker cannot sign the same message twice due to protocol restrictions.

However, the attack above does not apply when the message is partially unknown to the attacker. Recently, Coron et al. published an attack [8] which allows the attacker to recover the private key in such a context.

An obvious countermeasure to thwart these attacks is to verify the signature by using the public key (e, N) . This method, which implies a second modular exponentiation, is very costly if the public exponent e is large. Moreover, the public exponent is not always available, such as in JavaCard context.

To bypass this inconvenience, Shamir proposed in [20] another method. He suggested to choose a small integer r and to compute $S_{pr} = m^d \pmod{pr}$ and $S_{qr} = m^d \pmod{qr}$. The integrity of these two exponentiations is then ensured by testing whether $S_{pr} \equiv S_{qr} \pmod{r}$ before combining S_{pr} and S_{qr} with the CRT-recombination formula. However, this method does not protect the CRT-recombination; this has been exploited by Aumüller et al. in [1] to break this scheme. Shamir’s method has then been improved in [24], [1], [3], [6], [14], [13] but all these methods have been broken in [23], [25], [22], [2], [9], [9] respectively. Amongst Shamir’s method variants, only one algorithm is satisfactory from a security point of view: the one proposed by Vigilant [21] at CHES 2008.

Another methodology has been proposed by Giraud in which the fault detection comes from the exponentiation algorithm itself [11]. He pointed out that by using the Montgomery powering ladder [12] to compute $m^d \pmod{N}$, both values $m^{d-1} \pmod{N}$ and $m^d \pmod{N}$ are available at the end of the computation. These values can then be used to verify the integrity of the exponentiation by testing if m times the first value is equal to the second one. This method has been extended by Boscher et al. to a right-to-left exponentiation algorithm [5] and it has been improved by Rivain in [18].

As explained above, very few schemes offered an efficient way of protecting CRT-RSA against fault attacks without using the public exponent. In the next section, we describe in detail one of them.

III. VIGILANT’S CRT-RSA

Vigilant proposed at CHES 2008 an efficient method to protect a modular exponentiation against fault attacks and extended this result to the case of CRT-RSA [21]. As pointed out by Rivain in [18], compared to other techniques, Vigilant’s countermeasure seems to be one of the most cost-effective regarding all embedded device constraints. Indeed in this method, the public exponent is not required, there is no pre-computation, no extra parameters and no need of a per-

sonalization incompatible with the JavaCard standard. Moreover, the overhead added by the countermeasure in terms of performance and memory consumption is acceptable.

A. Vigilant's generic secure exponentiation

The principle of Vigilant's secure exponentiation method consists in computing $m^d \bmod N$ in \mathbb{Z}_{Nr^2} where r is a small random integer coprime with N . Then the base m is transformed into m' such that:

$$m' \equiv \begin{cases} m \bmod N \\ 1 + r \bmod r^2 \end{cases}$$

This implies that:

$$S' = m'^d \bmod Nr^2 \equiv \begin{cases} m^d \bmod N \\ 1 + dr \bmod r^2 \end{cases}$$

Therefore a consistency check of the result S' can be performed modulo r^2 from d and r . If the verification $S' = 1 + dr \bmod r^2$ is successful, then the final result $S = S' \bmod N$ is returned.

B. Vigilant's application to RSA with CRT

The application to RSA with CRT is derived from Shamir's method published in 1997 [20]. The principle is to perform both half-exponentiations modulo pr^2 and qr^2 . Therefore it is possible to perform a final consistency check after recombination, guaranteeing that no error occurred during the computations of S_p or S_q and during the recombination. Algorithm 1 explains the steps to perform more precisely; this figure is a pure copy of Vigilant's scheme found in [21].

This method has some real advantages:

- No special property is needed on the random integer r except to be odd and of enough entropy;
- No precomputation is needed;
- Only p, q, d_p, d_q, i_q and the input message m are needed for the calculation. No extra parameter is required;
- The overhead for the performance and the memory consumption implied by the countermeasure is reasonable.

Algorithm 1 Vigilant's CRT-RSA scheme [21]

INPUTS: The message to sign m , the private key (p, q, d_p, d_q, i_q) , a 32-bit random integer r , four 64-bit random integers R_1, R_2, R_3 and R_4

OUTPUT: The signature $S = m^d \bmod N$

1. $p' = pr^2, m_p = m \bmod p'$
 2. $i_{pr} = p^{-1} \bmod r^2, \beta_p = pi_{pr}$ and $\alpha_p = 1 - \beta_p \bmod p'$
 3. $\hat{m}_p = \alpha_p m_p + \beta_p(1 + r) \bmod p'$
 4. **if** $(\hat{m}_p \neq m \bmod p)$ **then**
 5. **return** error
 6. $d'_p = d_p + R_1(p - 1)$
 7. $S_{pr} = \hat{m}_p^{d'_p} \bmod p'$
 8. **if** $(d'_p \neq d_p \bmod p - 1)$ **then**
 9. **return** error
 10. **if** $(\beta_p S_{pr} \neq \beta_p(1 + d'_p r) \bmod p')$ **then**
 11. **return** error
 12. $S'_p = S_{pr} - \beta_p(1 + d'_p r - R_3)$
 13. $q' = qr^2, m_q = m \bmod q'$
 14. $i_{qr} = q^{-1} \bmod r^2, \beta_q = qi_{qr}$ and $\alpha_q = 1 - \beta_q \bmod q'$
 15. $\hat{m}_q = \alpha_q m_q + \beta_q(1 + r) \bmod q'$
 16. **if** $(\hat{m}_q \neq m \bmod q)$ **then**
 17. **return** error
 18. **if** $(m_p \bmod r^2 \neq m_q \bmod r^2)$ **then**
 19. **return** error
 20. $d'_q = d_q + R_2(q - 1)$
 21. $S_{qr} = \hat{m}_q^{d'_q} \bmod q'$
 22. **if** $(d'_q \neq d_q \bmod q - 1)$ **then**
 23. **return** error
 24. **if** $(\beta_q S_{qr} \neq \beta_q(1 + d'_q r) \bmod q')$ **then**
 25. **return** error
 26. $S'_q = S_{qr} - \beta_q(1 + d'_q r - R_4)$
 27. $S = S'_q + q(i_q(S'_p - S'_q) \bmod p')$
 28. $N = pq$
 29. **if** $(N[S - R_4 - qi_q(R_3 - R_4)] \neq 0 \bmod Nr^2)$ **then**
 30. **return** error
 31. **if** $(qi_q \neq 1 \bmod p)$ **then**
 32. **return** error
 33. **return** $S \bmod N$
-

IV. DESCRIPTION OF TWO FAULT ATTACKS ON VIGILANT'S CRT-RSA

After recalling the fault model assumptions made in [21], we present two attacks on Algorithm 1. These attacks are very efficient since only one faulty signature allows us to recover the whole private key. Finally, we present practical results of the first attack.

A. Fault Model

In [21], it is claimed that Algorithm 1 is meant to resist fault attacks under the following fault model. The attacker can:

- modify a value in memory obtaining a totally random result uncorrelated to the original value (as known as permanent fault);
- modify a value when it is handled in local registers, without modifying the global value in memory. The value handled obtained is fully random looking to the attacker and uncorrelated to the original value (as known as transient fault);

but he cannot:

- modify the code execution. Processor instructions cannot be replaced or removed while executing code;
- inject a permanent fault in the input elements, the message m or in the key (p, q, d_p, d_q, i_q) ;
- change the Boolean result of a conditional check. An expression “if $a = b$ ” has a result *true* or *false* that cannot be modified.

In the following section, we show that Algorithm 1 is vulnerable to fault attacks in accordance with this fault model.

B. Disturbance of Modulus Computation

The attack presented in this section is based on the fact that the integrity of the modulus computation is never checked during Algorithm 1. Indeed, we consider the test of step 29 ; it is checked that:

$$N[S - R_4 - qi_q(R_3 - R_4)] \bmod Nr^2 = 0 \quad (1)$$

otherwise an error is returned. Our crucial observation is that this test is actually independent of N ; namely for any value of N it is equivalent to:

$$S - R_4 - qi_q(R_3 - R_4) = 0 \bmod r^2$$

Therefore if we can induce a fault when computing $N = pq$ just before this test, then equality (1) will still be valid with the faulty \tilde{N} and eventually a faulty signature $\tilde{S} = S \bmod \tilde{N}$ will be returned.

More precisely, we induce a transient fault on p when $N = pq$ is computed at the previous step. This means that $\tilde{N} = \tilde{p}q$ is computed instead of $N = pq$, where \tilde{p} is a random value. Since this is a transient fault the global value of p is not modified in memory; this implies that the equality $qi_q = 1 \bmod p$ that is

checked later is still valid. As explained previously the equality:

$$\tilde{N}[S - R_4 - qi_q(R_3 - R_4)] = 0 \bmod \tilde{N}r^2$$

is still valid even for the faulty \tilde{N} . Therefore a faulty signature:

$$\tilde{S} = S \bmod \tilde{N} = S \bmod \tilde{p}q$$

is returned. Since the signature \tilde{S} is faulty modulo p but correct modulo q , with very high probability a single gcd computation enables to recover the factorization of N :

$$\gcd(\tilde{S}^e - m, N) = q$$

We note that a transient fault on q when computing $N = pq$ also gives the factorization of N .

At this step, one can wonder about the pertinence of this fault model in practice. To answer this question we present in Algorithm 2 the classical long integer multiplication algorithm. Typically, such a multiplication is performed on smart cards by a crypto-coprocessor based on a t -bit \times t -bit hardware multiplier, which provides dedicated operations for long integers. In the following, we denote by R_i 's the t -bit internal registers of the crypto-coprocessor and by $(x_{k-1}x_{k-2} \dots x_1x_0)_b$ the decomposition of the integer x in base $b = 2^t$.

Algorithm 2 Long Integer Multiplication

INPUTS: $x = (x_{k-1}x_{k-2} \dots x_1x_0)_b$,
 $y = (y_{k-1}y_{k-2} \dots y_1y_0)_b$
 OUTPUT: $z = x \times y$

1. **for** i **from** 0 to $2k - 1$ **do**
 2. $z_i \leftarrow 0$
 3. **for** i **from** 0 to $k - 1$ **do**
 4. $R_0 \leftarrow 0$
 5. $R_1 \leftarrow y_i$
 6. **for** j **from** 0 to $k - 1$ **do**
 7. $R_2 \leftarrow x_j$
 8. $R_3 \leftarrow z_{i+j}$
 9. $(R_5R_4)_b \leftarrow R_3 + R_2 \times R_1 + R_0$
 10. $z_{i+j} \leftarrow R_4$
 11. $R_0 \leftarrow R_5$
 12. $z_{i+k} \leftarrow R_5$
 13. **return** z
-

By analyzing Algorithm 2, one can observe that by disturbing step 5 which represents the loading

of the value y_i into a crypto-coprocessor register, an attacker obtains kx as output, k being equal to $(y_{k-1} \dots y_{i+1} k_i y_{i-1} \dots y_0)_b$ where k_i is a t -bit random value. This proves the practicality of our fault model.

C. Disturbance of $p - 1$ or $q - 1$ Computation

In this section, we present another transient fault attack on Algorithm 1. Our second attack is based on the fact that the integrity of the $p - 1$ computation is not verified throughout Algorithm 1. Therefore, if the computation of $p - 1$ is disturbed during step 6 then it leads to a faulty exponent \tilde{d}'_p which is used to compute a faulty signature \tilde{S} . Therefore this faulty signature \tilde{S} is not congruent to S_p modulo p and it is congruent to S_q modulo q . The secret parameter q can thus be recovered by computing the GCD of $\tilde{S}^e - m \bmod N$ and N . This attack works similarly if the subtraction $q - 1$ is disturbed during step 20.

For this attack to succeed, we need to make the weak assumptions that:

- the value $p - 1$ computed during step 6 is kept in memory up to step 8 (i.e. is not recomputed at step 8). This hypothesis is realistic since this is the more efficient way of implementing steps 6 to 8 and no particular information on this computation is given in [21].
- and that the value $p - 1$ (resp. $q - 1$) is not used for further steps after half exponentiations. Indeed, if the values p and q used for the final check $q^i q = 1 \bmod p$ are recomputed from the values $p - 1$ and $q - 1$ already present in memory, this attack would be detected.

Finally, the corresponding fault model can be seen as the injection of a transient fault in a crypto-coprocessor register during the subtraction, which complies with Vigilant's fault model.

D. The First Attack in Practice

We have implemented Algorithm 1 on a smart card equipped with an 8-bit CPU and a 32-bit crypto-coprocessor. By observing the power consumption of this device during the execution of Algorithm 1, we obtain Figure 1 where the top curve corresponds to the I/O and the bottom curve correspond to the power consumption of the card.

One can observe two similar patterns corresponding to the computation of S'_p and of S'_q . These patterns

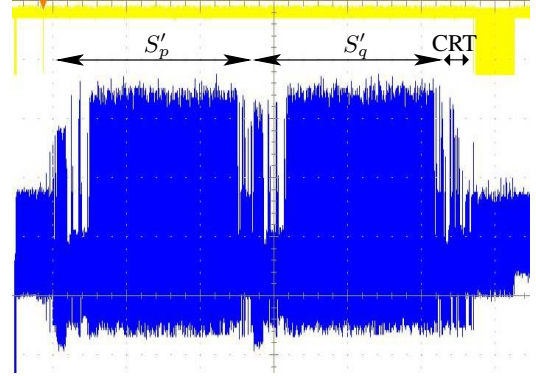


Figure 1. Power consumption observed during Algorithm 1 execution.

are followed by the signal corresponding to the CRT-recombination execution. When focusing on the latter we obtain the signals depicted in Figure 2 where the modulus computation of step 28 is identified by the arrow.

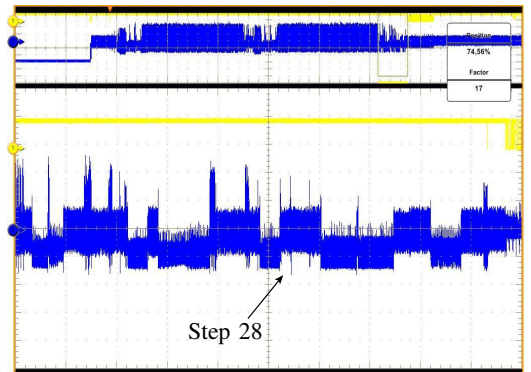


Figure 2. Power consumption observed during the CRT-recombination.

By observing Figure 2, one can note three main levels of power consumption. The lowest one corresponds to CPU operations and the other ones correspond to crypto-processor operations.

By using a laser, we succeeded in disturbing the multiplication between p and q such that the corresponding faulty signature allows us to obtain the secret parameter p by using the method described in Section IV-B.

V. OUR COUNTERMEASURES

In this section, we suggest some modifications to Algorithm 1 to resist the attacks previously described.

Regarding the attack presented in Section IV-B, one has to verify the integrity of the modulus computation

during step 28. This integrity check can be performed in many ways:

- The verification $N[S - R_4 - qi_q(R_3 - R_4)] \bmod Nr^2 = 0$ can be replaced by $pq[S - R_4 - qi_q(R_3 - R_4)] \bmod Nr^2 = 0$. The modification of N into a faulty \tilde{N} would then be detected by this verification with very high probability. Namely for a faulty \tilde{N} we have $p \cdot q \neq 0 \bmod \tilde{N}$ with very high probability.
- We pick a random integer R and we compute $R_p = p \bmod R$, $R_q = q \bmod R$ and $R_N = R_p \cdot R_q \bmod R$. Before returning the result modulo N , it can be also verified that $N \bmod R = R_N$. R is for example a 32-bit integer. The detection probability of a fault injection in N would be detected with a probability about $1 - \frac{1}{|R|}$.
- A test $N \cdot i_{qr} = p \bmod r^2$ could also be added before returning the result.

To prevent the attack described in Section IV-C, the developer can recompute the value $p - 1$ (resp. $q - 1$) in step 8 (resp. in step 22). Since permanent faults on the private key parameters are not considered in Vigilant's fault model, such double computations make Algorithm 1 resistant against the attack described in Section IV-C.

We describe in Algorithm 3 our secure version of Vigilant's scheme. One may note that the cost of our countermeasures is negligible, since only one modular multiplication and two subtractions have been added and no memory extra buffer is required. Therefore even with this extra countermeasure, the Vigilant's scheme remains efficient and suited to embedded devices.

VI. CONCLUSION

This paper discusses the fault-resistance security of Vigilant's scheme proposed at CHES 2008. We have underlined two weaknesses which can be exploited by an attacker to recover the whole private key by using only one faulty signature. We have also suggested simple countermeasures to strengthen Vigilant's scheme. Since our countermeasures have a negligible cost, the scheme remains well-suited to embedded device constraints.

This paper shows that the security proof of Algorithm 1 is incomplete in the original paper. Formal proof of the FA-resistance of Vigilant's scheme including our countermeasures is still an open (and challenging) issue.

Algorithm 3 Improved Vigilant's CRT-RSA scheme

INPUTS: The message to sign m , the private key (p, q, d_p, d_q, i_q) , a 32-bit random integer r , four 64-bit random integers R_1, R_2, R_3 and R_4

OUTPUT: The signature $S = m^d \bmod N$

1. $p' = pr^2$, $m_p = m \bmod p'$
 2. $i_{pr} = p^{-1} \bmod r^2$, $\beta_p = pi_{pr}$ and $\alpha_p = 1 - \beta_p \bmod p'$
 3. $\hat{m}_p = \alpha_p m_p + \beta_p(1 + r) \bmod p'$
 4. **if** ($\hat{m}_p \neq m \bmod p$) **then**
 5. **return** error
 6. $p_{minusone} = p - 1$
 7. $d'_p = d_p + R_1 p_{minusone}$
 8. $S_{pr} = \hat{m}_p^{d'_p} \bmod p'$
 9. $p_{minusone} = p - 1$
 10. **if** ($d'_p \neq d_p \bmod p_{minusone}$) **then**
 11. **return** error
 12. **if** ($\beta_p S_{pr} \neq \beta_p(1 + d'_p r) \bmod p'$) **then**
 13. **return** error
 14. $S'_p = S_{pr} - \beta_p(1 + d'_p r - R_3)$
 15. $q' = qr^2$, $m_q = m \bmod q'$
 16. $i_{qr} = q^{-1} \bmod r^2$, $\beta_q = qi_{qr}$ and $\alpha_q = 1 - \beta_q \bmod q'$
 17. $\hat{m}_q = \alpha_q m_q + \beta_q(1 + r) \bmod q'$
 18. **if** ($\hat{m}_q \neq m \bmod q$) **then**
 19. **return** error
 20. **if** ($m_p \bmod r^2 \neq m_q \bmod r^2$) **then**
 21. **return** error
 22. $q_{minusone} = q - 1$
 23. $d'_q = d_q + R_2 q_{minusone}$
 24. $S_{qr} = \hat{m}_q^{d'_q} \bmod q'$
 25. $q_{minusone} = q - 1$
 26. **if** ($d'_q \neq d_q \bmod q_{minusone}$) **then**
 27. **return** error
 28. **if** ($\beta_q S_{qr} \neq \beta_q(1 + d'_q r) \bmod q'$) **then**
 29. **return** error
 30. $S'_q = S_{qr} - \beta_q(1 + d'_q r - R_4)$
 31. $S = S'_q + q(i_q(S'_p - S'_q) \bmod p')$
 32. $N = pq$
 33. **if** ($pq[S - R_4 - qi_q(R_3 - R_4)] \neq 0 \bmod Nr^2$) **then**
 34. **return** error
 35. **if** ($qi_q \neq 1 \bmod p$) **then**
 36. **return** error
 37. **return** $S \bmod N$
-

REFERENCES

- [1] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter, and J.-P. Seifert. Fault Attacks on RSA with CRT: Concrete Results and Practical Countermeasures. In CHES 2002, vol. 2523 of *LNCS*, pp. 260–275. Springer, 2002.
- [2] A. Berzati, C. Canovas, and L. Goubin. (In)security Against Fault Injection Attacks for CRT-RSA Implementations. In FDTC 2008, pp. 101–107. IEEE Computer Society, 2008.
- [3] J. Blömer, M. Otto, and J.-P. Seifert. A New RSA-CRT Algorithm Secure against Bellcore Attacks. In CCS'03, pp. 311–320. ACM Press, 2003.
- [4] D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In EUROCRYPT '97, vol. 1233 of *LNCS*, pp. 37–51. Springer, 1997.
- [5] A. Boscher, R. Naciri, and E. Prouff. CRT RSA Algorithm Protected against Fault Attacks. In WISTP 2007, vol. 4462 of *LNCS*, pp. 229–243. Springer, 2007.
- [6] M. Ciet and M. Joye. Practical Fault Countermeasures for Chinese Remaindering Based RSA. In FDTC'05, pp. 124–132, 2005.
- [7] J.-S. Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In CHES '99, vol. 1717 of *LNCS*, pp. 292–302. Springer, 1999.
- [8] J.-S. Coron, D. Naccache, and M. Tibouchi. Fault Attacks Against EMV Signatures. In CT-RSA 2010, vol. 5985 of *LNCS*, pp. 208–220. Springer, 2010.
- [9] E. Dottax, C. Giraud, M. Rivain, and Y. Sierra. On Second-Order Fault Analysis Resistance for CRT-RSA Implementations. In WISTP 2009, vol. 5746 of *LNCS*, pp. 68–83. Springer, 2009.
- [10] H. Garner. The Residue Number System. *IRE Transactions on Electronic Computers*, 8(6):140–147, June 1959.
- [11] C. Giraud. An RSA Implementation Resistant to Fault Attacks and to Simple Power Analysis. *IEEE Transactions on Computers*, 55(9):1116–1120, Sept. 2006.
- [12] M. Joye and S.-M. Yen. The Montgomery Powering Ladder. In CHES 2002, vol. 2523 of *LNCS*, pp. 291–302. Springer, 2002.
- [13] C. H. Kim and J.-J. Quisquater. Fault Attacks for CRT Based RSA: New Attacks, New Results, and New Countermeasures. In WISTP 2007, vol. 4462 of *LNCS*, pp. 215–228. Springer, 2007.
- [14] C. H. Kim and J.-J. Quisquater. How Can We Overcome Both Side Channel Analysis and Fault Attack on RSA-CRT? In FDTC 2007, pp. 21–29. IEEE Computer Society, 2007.
- [15] P. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In CRYPTO '99, vol. 1666 of *LNCS*, pp. 388–397. Springer, 1999.
- [16] A. Lenstra. Memo on RSA Signature Generation in the Presence of Faults. Manuscript, 1996.
- [17] PKCS #1. *RSA Cryptography Specifications Version 2.1*. RSA Laboratories, 2003.
- [18] M. Rivain. Securing RSA against Fault Analysis by Double Addition Chain Exponentiation. In CT-RSA 2009, vol. 5473 of *LNCS*, pp. 459–480. Springer, 2009.
- [19] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
- [20] A. Shamir. Improved Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attacks. Patent number: WO9852319, Nov. 1998. Also presented at the rump session of Eurocrypt'97.
- [21] D. Vigilant. RSA with CRT: A New Cost-Effective Solution to Thwart Fault Attacks. In CHES 2008, vol. 5154 of *LNCS*, pp. 130–145. Springer, 2008.
- [22] D. Wagner. Cryptanalysis of a Provable Secure CRT-RSA Algorithm. In CCS'04, pp. 82–91. ACM Press, 2004.
- [23] S.-M. Yen, D. Kim, and S. Moon. Cryptanalysis of Two Protocols for RSA with CRT Based on Fault Infection. In FDTC 2006, vol. 4236 of *LNCS*, pp. 53–61. Springer, 2006.
- [24] S.-M. Yen, S.-J. Kim, S.-G. Lim, and S.-J. Moon. RSA Speedup with Residue Number System Immune against Hardware Fault Cryptanalysis. In ICISC 2001, vol. 2288 of *LNCS*, pp. 397–413. Springer, 2001.

- [25] S.-M. Yen, S. Moon, and J.-C. Ha. Hardware Fault Attack on RSA with CRT Revisited. In ICISC 2002, vol. 2587 of *LNCS*, pp. 374–388. Springer, 2002.