Fault Cryptanalysis of CHES 2014 Symmetric Infective Countermeasure

Alberto Battistello[1,2] and Christophe Giraud[1]

[1] Oberthur Technologies
Cryptography and Security Group
4, allée du Doyen Georges Brus, 33 600 Pessac, France
[2] Versailles Saint-Quentin-en-Yvelines University,
45 Avenue des Etats-Unis, 78035 Versailles Cedex, France
{a.battistello,c.giraud}@oberthur.com

Abstract. Fault injection has become over the years one of the most dangerous threats for embedded devices such as smartcards. It is thus mandatory for any embedded system to implement efficient protections against this hazard. Among the various countermeasures suggested so far, the idea of *infective computation* seems fascinating, probably due to its aggressive strategy. Originally conceived to protect asymmetric cryptosystems, infective computation has been recently adapted to symmetric systems. This paper investigates the security of a new symmetric infective countermeasure suggested at CHES 2014. By noticing that the number of executed rounds is not protected, we develop four different attacks allowing one to efficiently recover the secret key of the underlying cryptosystem by using any of the three most popular fault models used in literature.

Keywords: Fault Attack, Infective Countermeasure, AES.

1 Introduction

Over the last 20 years, the security of embedded devices has been challenged by several specific attacks. In particular, Boneh *et al.* showed in 1996 that a simple disturbance during the execution of an embedded algorithm may totally breaks its security [5]. They illustrated this new method by explaining how to break an CRT-RSA implementation by inducing only one error during the algorithm execution. By using so-called *fault attacks*, many signature schemes and symmetric cryptosystems have then been broken only a few months after the original Boneh *et al.* publication [1, 4]. A whole new research field thus appeared aiming at discovering new fault-based attacks and providing efficient countermeasures [11, 13, 14]. While researchers improved and discovered new fault attacks on each and every cryptosystem, the countermeasures were difficult to find and costly to implement. Among the ideas that emerged, the two most popular methods are the signature verification for asymmetric systems and the duplication method for symmetric ones. The first one simply consists in performing

a signature verification on the result. If a fault occurred then the signature is not consistent and the verification fails. The second method requires to execute the algorithm twice and to compare both results. If an attacker disturbs one of the two executions then the comparison detects the attack and no output is returned. A third approach called *infective* was suggested in 2001 by Yen *et al.* [18]. Their method consists in modifying and amplifying the injected error in such a way that the attacker cannot retrieve any information from the corresponding faulty output. Firstly applied to asymmetric cryptosystems [18] this method is tricky to conceive and all infective countermeasures for asymmetric algorithms published so far have been broken, see [3, 8, 16, 17] for instance. The infective method has been adapted only recently to the symmetric case. The first example of symmetric infection was proposed by Lomné *et al.* in 2012 to protect AES [12]. The authors suggest to execute the AES twice and to compute the infection by multiplicatively masking the differential of the two AES outputs. A second symmetric infection was suggested by Gierlichs *et al.* in [10] by using a random sequence of cipher and redundant rounds together with dummy rounds. If the outputs of the redundant and cipher rounds are not equal then the temporary result is infected. Unfortunately both methods have been broken by Battistello and Giraud in [2].

At CHES 2014, Tupsamudre *et al.* improved in [15] the attack of Battistello and Giraud and they also suggested an improved version of the infective countermeasure of Gierlichs *et al.*. While this new proposal is secure against the attacks found in [2, 15], one wonders if they are sufficient to make a symmetric implementation effectively secure, especially in the absence of a proof of security.

In this paper we study the proposition of [15] and show several flaws in its design. We suggest four different attack paths that allow to bypass the infective countermeasure by using three different fault models. This paper not only shows that the CHES 2014 infective countermeasure is not secure but it also shows that all known infective countermeasures are broken.

The rest of the paper is organized as follows. In Section 2 we recall the countermeasure suggested in [15]. Section 3 presents four different attacks on this countermeasure. In particular, we show that it is possible to recover the secret key by using any of the three most popular fault models used in the literature. Section 4 finally concludes this paper.

2 Description of CHES 2014 Infective Countermeasure

The infective countermeasure suggested at CHES 2014 by Tupsamudre *et al.* [15] is based on the work presented at LatinCrypt 2012 by Gierlichs *et al.* [10]. For the sake of simplicity we recall in Algorithm 1 the CHES 2014 countermeasure suggested in [15] applied to AES-128. For more information about AES, the reader can refer to [9].

Algorithm 1 uses three states denoted R_0, R_1 and R_2 for the cipher, the redundant and the dummy rounds respectively. The execution order of these rounds is given by a random bit string $rstr$ generated at the beginning of the

Algorithm 1: CHES 2014 Countermeasure applied on AES-128

Inputs : Plaintext P, round keys k^j for $j \in \{1, \ldots, 11\}$, pair (β, k^0), security level $t \geq 22$

Output: Ciphertext C = AES-128(P,K)

1 State $R_0 \leftarrow P$; Redundant state $R_1 \leftarrow P$; Dummy state $R_2 \leftarrow \beta$
2 $i \leftarrow 1$, $q \leftarrow 1$
3 $rstr \leftarrow \{0,1\}^t$ // $\#1(rstr) = 22, \#0(rstr) = t - 22$
4 **while** $q \leq t$ **do**
5 $\quad \lambda \leftarrow rstr[q]$ // $\lambda = 0$ implies a dummy round
6 $\quad \kappa \leftarrow (i \wedge \lambda) \oplus 2(\neg\lambda)$
7 $\quad \zeta \leftarrow \lambda \cdot \lceil i/2 \rceil$ // ζ is actual round counter, // $\zeta = 0$ is for dummy round
8 $\quad R_\kappa \leftarrow \texttt{RoundFunction}(R_\kappa, k^\zeta)$
9 $\quad \gamma \leftarrow \lambda(\neg(i \wedge 1)) \cdot \texttt{BLFN}(R_0 \oplus R_1)$
10 $\quad \delta \leftarrow (\neg\lambda) \cdot \texttt{BLFN}(R_2 \oplus \beta)$
11 $\quad R_0 \leftarrow (\neg(\gamma \vee \delta) \cdot R_0) \oplus ((\gamma \vee \delta) \cdot R_2)$
12 $\quad i \leftarrow i + \lambda$
13 $\quad q \leftarrow q + 1$
14 **end**
15 **return** (R_0)

algorithm. Each "0" on the string encodes a dummy round, while a "1" encodes a redundant or cipher round. Each time a "1" occurs, an index i is incremented and a redundant round (resp. a cipher round) is executed if i is odd (resp. even). Algorithm 1 thus executes a loop over the $rstr$ string bits and executes a cipher, redundant or dummy round accordingly. One may note that Algorithm 1 computes the redundant round before the cipher round all along the algorithm and dummy rounds can happen randomly at any time.

Dummy rounds are executed over a dummy state R_2 which is initialized to a random 128-bit value β and by using the round key k^0 which is computed such that:

$$\texttt{RoundFunction}(\beta, k^0) = \beta \ . \tag{1}$$

The number of dummy rounds is parameterized by a security level t chosen by the developer. More precisely, this parameter represents the whole number of cipher, redundant and dummy rounds performed during Algorithm 1 execution. For instance in the case of AES-128, $t - 22$ dummy rounds will be performed.

Regarding the infective part, a first infection is activated after each cipher round if its state R_0 is different from the redundant state R_1 (Steps 9 and 11). Moreover another infection occurs if $R_2 \neq \beta$ after the execution of a dummy round (Steps 10 and 11). These infections consist in replacing the cipher state R_0 with the random value R_2, leaving no chance to the attacker to obtain information on the secret key once the infection is applied. To do so, a Boolean function BLFN is used which maps non-zero 128-bit values to 1 and outputs 0 for a null input.

Compared to the original LatinCrypt 2012 proposal, the CHES 2014 infective countermeasure differs by the way of dealing with the sequence of cipher, redundant and dummy rounds which is now done by using a random string $rstr$ and by the way the infection is performed which is now fully random.

Despite the security analysis of Algorithm 1 presented in [15], we show in the next section that it contains many weaknesses. Furthermore we show that for each of the three most popular fault models used in literature an attacker can recover the full secret key.

3 Attacks

In this section we firstly present the principle of our attacks which are based on the fact that the variables dealing with the number of rounds to perform are not protected. We then exploit this remark to suggest four different attacks that use different fault models such as the instruction skip, the stuck-at and the random error fault model.

3.1 Principle of Our Attacks

Due to the improvements of Algorithm 1 compared with the original LatinCrypt 2012 countermeasure, it is impossible for the attacker to obtain any information on the secret key once the infection has occurred. In order to thwart this countermeasure, we thus investigate the possibility to disturb the number of executed rounds since the corresponding variable is not protected in integrity. Indeed, if the attacker succeeds in disturbing the number of rounds she may be able to retrieve the secret key from the corresponding faulty ciphertext [6, 7]. Let us show in the following of this section how such an attack works if the last round of an AES-128 has been skipped.

If the attacker knows a correct and a faulty ciphertext obtained by skipping the last AES round then it is equivalent to know the input S^{10} and the output S^{11} of the last round. Due to the lack of *MixColumns* transformation during the last AES round, the last round key k^{11} can be recovered byte per byte by XORing the corresponding bytes of S^{10} and S^{11}:

$$k_i^{11} = S_i^{11} \oplus \texttt{SBox}(S^{10}_{\mathrm{SR}^{-1}(i)}) \ , \ \forall i \in \{1, \ldots, 16\}, \tag{2}$$

where SR corresponds to the byte index permutation induced by the transformation *ShiftRows*. In such a case, the attacker can recover the full AES-128 key from only one pair of correct and faulty ciphertexts.

One may note that this attack works similarly if the attacker knows the input and the output of the first round. For further details on the first round attack the reader can refer to [6].

In the following, we describe different ways of disturbing Algorithm 1 by using several fault models such that it does not perform the AES with the correct number of rounds whereas no infection is performed. In our description we make use of AES-128 as a reference, however our attacks can straightforwardly apply to other key sizes.

3.2 Attack 1 by Using Instruction Skip Fault Model

The attack presented in this section exploits the fact that whenever the variable i is odd and $\lambda = 1$ then a redundant round is executed and that this kind of round does not involve any infection. To exploit this remark, we assume that the attacker is able to skip an instruction of its choice by means of a fault injection.

Description If the attacker skips Step 12 of Algorithm 1 after the last redundant round then the increment of i is not performed. Therefore i stays odd so the last cipher round is replaced by another redundant round. As no infection is involved for redundant rounds, the algorithm returns the output of the penultimate round. The attacker can thus take advantage of such an output to recover the secret key as explained in Section 3.1.

Efficiency As explained in Appendix A, the probability of skipping the last cipher round and thus to recover the AES key after repeating the fault injection (i.e. skipping Step 12) r times during the q-th round is given by:

$$\mathcal{P}_r = 1 - \left(1 - \frac{\binom{q-1}{20}\binom{t-q}{1}}{\binom{t}{22}}\right)^r \quad . \tag{3}$$

where t is the total number of rounds performed during Algorithm 1.

Some numerical values of (3) are given in Table 1 for t equal to 30, 40 and 50, $q = t-3, \cdots, t-1$ and $r = 1, \cdots, 4$. One can notice that if the fault is injected when q equals t then the attack does not work because all the rounds have already been executed.

t	q	Number r of faults			
		1	2	3	4
30	27	11.80%	22.21%	31.39%	39.49%
	28	30.34%	51.48%	66.20%	76.46%
	29	53.10%	78.01%	89.69%	95.16%
40	37	19.34%	34.93%	47.52%	57.66%
	38	28.06%	48.24%	62.76%	73.21%
	39	29.62%	50.46%	65.13%	75.46%
50	47	18.96%	34.32%	46.77%	56.86%
	48	22.00%	39.16%	52.54%	62.98%
	49	18.86%	34.16%	46.57%	56.65%

Table 1. Probability of obtaining a useful faulty ciphertext by skipping Step 12 during the q-th loop of Algorithm 1.

By analyzing Table 1, one can deduce the best strategy for the attacker. For example if $t = 30$ then the attacker should target the 29-th round in order to obtain the best chances of retrieving the key with the minimal number of fault injections.

Experiments The attack described in this section has been simulated for $t = 30$ and for each q between 25 and 29. The experience has been repeated 3 000 times for each configuration. The results of our tests are depicted in Figure 1.

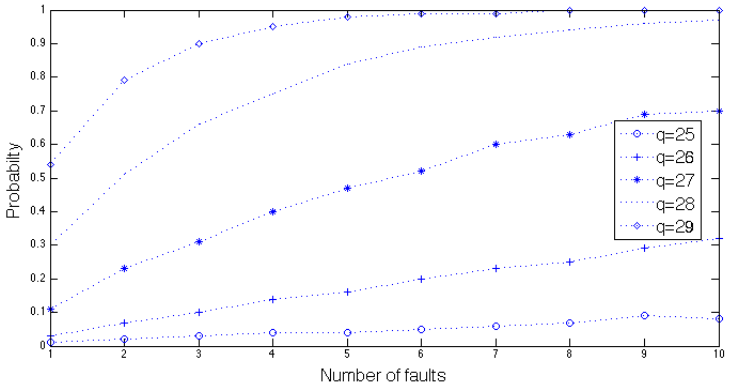

Fig. 1. Experimental probability of obtaining a useful faulty ciphertext by skipping Step 12 during the q-th loop of Algorithm 1 for $t = 30$.

By comparing Figure 1 and the row $t = 30$ of Table 1, one can notice that the experiments perfectly match with the theoretical results.

3.3 Attack 2 by Using Stuck-at 0 Fault Model

In this section we use the *stuck-at 0* fault model where we assume that the attacker can set to zero a variable of her choice. As for the attack presented in Section 3.2, the goal of the attacker is to skip the execution of the last cipher round.

Description To avoid the execution of the last cipher round by using a stuck-at 0 fault model without activating an infection, the attacker can set to zero the variable λ right after Step 5 during the loop involving the last "1" of $rstr$, i.e. during the loop dealing with the last cipher round. The computation of the last cipher round is thus skipped since $\lambda = 0$ implies a dummy round. The attacker thus retrieves an exploitable faulty ciphertext that can be used to retrieve the secret key as described in Section 3.1. As no consistency check is performed on λ, $rstr$ nor on the number of cipher rounds executed, Algorithm 1 does not detect the fault.

Efficiency We detail in Appendix B the reasoning to compute the probability of obtaining at least one useful faulty ciphertext after r fault injections. Such a probability is given by:

$$\mathcal{P}_r = 1 - \left(1 - \frac{\binom{q-1}{21}}{\binom{t}{22}}\right)^r \quad . \tag{4}$$

Some numerical values of (4) are given in Table 2 for t equal to 30, 40 and 50, $q = t-2, \cdots, t$ and $r = 1, \cdots, 4$.

t	q	Number r of faults			
		1	2	3	4
30	28	5.06%	9.86%	14.42%	18.75%
	29	20.23%	36.37%	49.24%	59.51%
	30	73.33%	92.89%	98.10%	99.49%
40	38	11.36%	21.42%	30.35%	38.26%
	39	25.38%	44.33%	58.46%	69.00%
	40	55.00%	79.75%	90.89%	95.90%
50	48	14.14%	26.29%	36.71%	45.66%
	49	25.14%	43.96%	58.05%	68.60%
	50	44.00%	68.64%	82.44%	90.17%

Table 2. Probability of obtaining a useful faulty ciphertext by sticking λ at 0 during the q-th loop of Algorithm 1.

By comparing Table 2 with Table 1, one may note that the attack presented in this section is more efficient than the one presented in Section 3.2, especially when the attacker targets the last loop execution.

Experiments We simulated the attack for $t = 30$ and for q from 27 to 30. For each value of q we performed 3 000 tests with random $rstr$. The results of such experiments are depicted in Figure 2.

3.4 Attack 3 by Using Random Error Fault Model

We show in this section how the attacker can use the random error fault model to obtain a useful faulty ciphertext. In this fault model, we assume that the attacker can change the value of a chosen internal variable into a random value.

Description Due to its central role in the infection and scheduling, string $rstr$ is very sensitive. However, the authors of [15] do not suggest any mean of ensuring its integrity. We thus investigated this path and we noticed that an attacker can disturb the generation of $rstr$ at Step 3 of Algorithm 1 such that it does not contain 22 "1" anymore. If the fault disturbs the string $rstr$ such that it contains only 21 (resp. 20) "1" then Algorithm 1 does not execute the last cipher

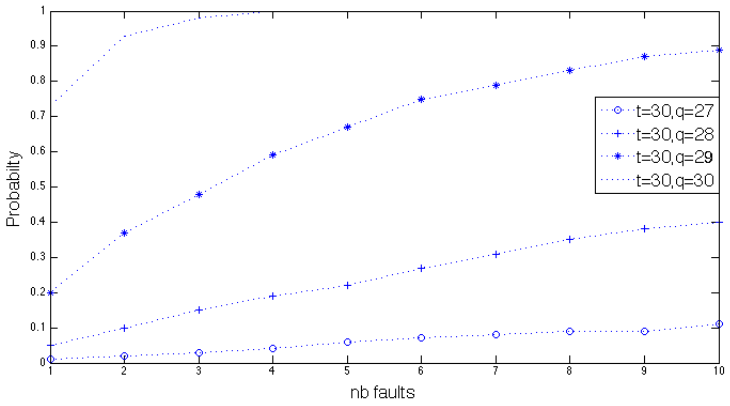

Fig. 2. Experimental probability of obtaining a useful faulty ciphertext by sticking λ at 0 during the q-th loop of Algorithm 1 for $t = 30$.

round (resp. the last redundant and cipher rounds). In both cases no infection is performed allowing the attacker to exploit the corresponding faulty ciphertext to recover the secret key as explained in Section 3.1.

Efficiency The probability to obtain at least one useful faulty ciphertext after injecting r faults during the q-th round is given by:

$$\mathcal{P}_r = 1 - \left(1 - \left(\sum_{i=1}^{8} \frac{\binom{t-8}{22-i}\binom{8}{i}}{\binom{t}{22}} \sum_{j=1}^{i} \frac{\binom{i}{j}\binom{8-i}{j-1}}{255} + \sum_{i=2}^{8} \frac{\binom{t-8}{22-i}\binom{8}{i}}{\binom{t}{22}} \sum_{j=2}^{i} \frac{\binom{i}{j}\binom{8-i}{j-2}}{255}\right)\right)^r . \quad (5)$$

For more details about the computation of this probability, the reader can refer to Appendix C.

Table 3 gives the probability to obtain a useful faulty ciphertext for t equal to 30, 40 and 50.

t	Number r of faults			
	1	2	3	4
30	41.63%	65.93%	80.11%	88.39%
40	34.72%	57.39%	72.18%	81.84%
50	24.60%	43.15%	57.13%	67.67%

Table 3. Probability of obtaining a useful faulty ciphertext by disturbing Step 3 of Algorithm 1.

Experiments Figure 3 shows the results obtained by simulating the attack described above. The simulations have been performed by generating a random string $rstr$ and disturbing it with an 8-bit random error. The test has been performed 3 000 times for each t equal to 30, 40 and 50.

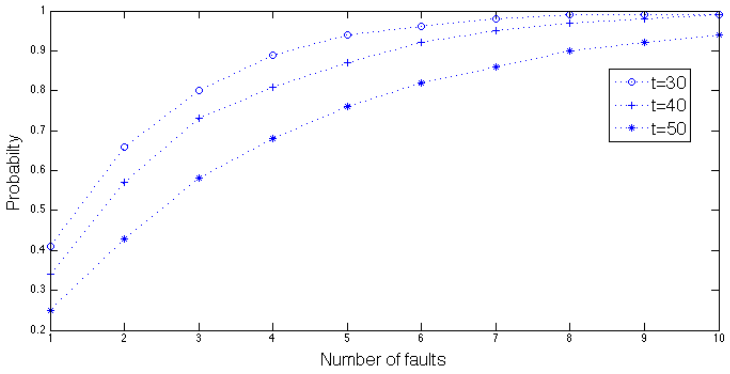

Fig. 3. Experimental probability of obtaining a useful faulty ciphertext by disturbing Step 3 of Algorithm 1.

3.5 Attack 4 by Using Random Error Fault Model

This section describes a second attack that can be mounted by using the random error fault model.

Description The idea of the attack is to disturb the increment of index q at Step 13 of Algorithm 1 during the execution of the first cipher round. We noticed that if the disturbance produces an error e such that $q \oplus e > t$ then the evaluation at Step 4 is false and the algorithm returns. If the algorithm computes only one cipher round then the attacker can use such an output to retrieve the first round key, cf. [6]. It is important to notice that in order to retrieve a useful output the attacker needs to disturb the execution during the first cipher round and not after a redundant or dummy round.

Efficiency As detailed in Appendix D, the probability to obtain at least one useful faulty ciphertext after injecting r faults during the q-th loop is given by:

$$\mathcal{P}_r = 1 - \left(1 - \frac{2^8 - t}{2^8} \sum_{i=2}^{3} \frac{\binom{q}{i}\binom{t-q}{22-i}}{\binom{t}{22}}\right)^r . \tag{6}$$

We give in Table 4 the probability that the attacker retrieves a useful faulty ciphertext for t equal to $30, 40$ and 50 and for q from 2 to 4.

t	q	Number of faults			
		1	2	3	4
30	2	46.88%	71.78%	85.01%	92.04%
	3	73.67%	93.07%	98.17%	99.52%
	4	60.52%	84.42%	93.85%	97.57%
40	2	24.99%	43.73%	57.79%	68.34%
	3	48.66%	73.64%	86.47%	93.05%
	4	58.22%	82.55%	92.71%	96.95%
50	2	15.17%	28.05%	38.96%	48.23%
	3	32.88%	54.95%	69.76%	79.70%
	4	45.58%	70.38%	83.88%	91.23%

Table 4. Probability of obtaining a useful faulty ciphertext by injecting a random error fault on Step 13 of Algorithm 1.

The attacker can use Table 4 to choose the best strategy for her attack. For example for $t = 30$, one obtains the best chances to retrieve a useful faulty ciphertext by attacking the third loop. Furthermore when comparing the efficiency of our four attacks, the attack presented in this section is the most efficient one.

Experiments We mounted several simulations where we disturbed the Step 13 of the q-th round with a random byte error e. We mounted the experiments for $t = 30$ and for q from 2 to 6. For each different q we repeated the experiment 3 000 times. The results of such experiments are shown in Figure 4.

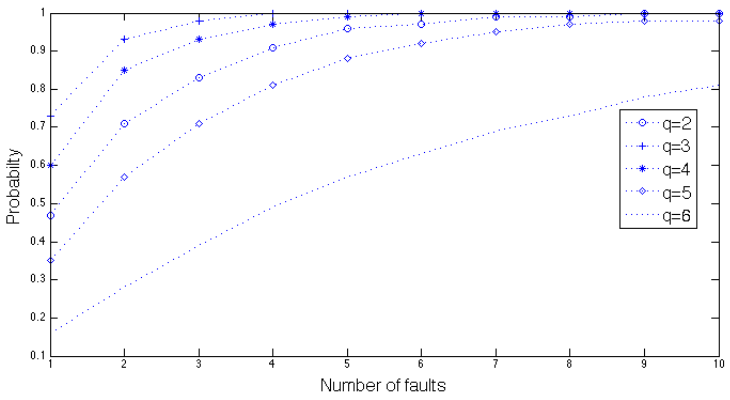

Fig. 4. Experimental probability of obtaining a useful faulty ciphertext by a injecting random error fault on Step 13 of Algorithm 1 for $t = 30$.

The simulations shows that this attack has a remarkable success rate. For example for $t = 30$, an attacker that reiterates the fault injection only twice

during the third loop has a probability of retrieving a useful faulty ciphertext greater than 90%.

3.6 A Remark on Possible Countermeasures

Whereas it seems obvious to counteract the attacks described above by ensuring the integrity of i, q, λ and $rstr$ for instance, we do not suggest an improvement of Algorithm 1. Indeed while searching for a suitable infective patch we didn't succeed in producing a satisfactory security proof and we think that such a proof is mandatory to give confidence in the fault resistance of any infective proposal.

4 Conclusion

In this article we showed that the infective countermeasure of CHES 2014 is not secure. Our attack targets variables and instructions that have been explicitly added to the original countermeasure in order to thwart fault attacks. While the new infective countermeasure gives no information to the attacker once the infection is applied, we discovered that it does not protect the number of cipher rounds effectively executed. By using this remark we applied the three most popular fault models and found four different attack paths that allow an attacker to recover the secret key of the underlying cryptosystem. For each attack we studied the success probability and performed simulations that validated our theoretical results.

Our work thus breaks the last infective countermeasure of symmetric cryptosystems claimed to be secure. With this work we also remark that the lack of formal security proofs in this field is clearly an issue. We hope that new ideas may pave the way to formally prove the security of cryptosystems against fault-based cryptanalysis.

References

1. F. Bao, R. Deng, Y. Han, A. Jeng, A. D. Narasimhalu, and T.-H. Ngair. Breaking Public Key Cryptosystems and Tamper Resistance Devices in the Presence of Transient Fault. In *5th Security Protocols Workshop*, volume 1361 of *LNCS*, pages 115–124. Springer, 1997.
2. A. Battistello and C. Giraud. Fault Analysis of Infective AES computations. In W. Fischer and J.-M. Schmidt, editors, *FDTC*, pages 101–107. IEEE, 2013.
3. A. Berzati, C. Canovas, and L. Goubin. (In)security Against Fault Injection Attacks for CRT-RSA Implementations. In L. Breveglieri, S. Gueron, I. Koren, D. Naccache, and J.-P. Seifert, editors, *Fault Diagnosis and Tolerance in Cryptography – FDTC 2008*, pages 101–107. IEEE Computer Society, 2008.
4. E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystem. In B. Kalisky Jr., editor, *Advances in Cryptology – CRYPTO '97*, volume 1294 of *LNCS*, pages 513–525. Springer, 1997.
5. D. Boneh, R. DeMillo, and R. Lipton. On the Importance of Checking Cryptographic Protocols for Faults. In W. Fumy, editor, *Advances in Cryptology – EUROCRYPT '97*, volume 1233 of *LNCS*, pages 37–51. Springer, 1997.

6. H. Choukri and M. Tunstall. Round Reduction Using Faults. In L. Breveglieri and I. Koren, editors, *Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC 2005*, 2005.
7. J.-M. Dutertre, A.-P. Mirbaha, D. Naccache, A.-L. Ribotta, A. Tria, and T. Vaschalde. Fault Round Modification Analysis of the Advanced Encryption Standard. In *IEEE International Symposium on Hardware-Oriented Security and Trust – HOST 2012*, pages 28–39. IEEE, 2012.
8. B. Feix and A. Venelli. Defeating with Fault Injection a Combined Attack Resistant Exponentiation. In W. Schindler and S. Huss, editors, *Third International Workshop on Constructive Side-Channel Analysis and Secure Design – COSADE 2013*, LNCS. Springer, 2013.
9. FIPS PUB 197. *Advanced Encryption Standard*. National Institute of Standards and Technology, Nov. 2001.
10. B. Gierlichs, J.-M. Schmidt, and M. Tunstall. Infective computation and dummy rounds: Fault protection for block ciphers without check-before-output. In A. Hevia and G. Neven, editors, *LATINCRYPT 2012*, volume 7533 of *LNCS*, pages 305–321. Springer, 2012.
11. C. Giraud. DFA on AES. In H. Dobbertin, V. Rijmen, and A. Sowa, editors, *Fourth Conference on the Advanced Encryption Standard (AES) – AES 4*, volume 3373 of *LNCS*, pages 27–41. Springer, 2004.
12. V. Lomné, T. Roche, and A. Thillard. On the Need of Randomness in Fault Attack Countermeasures – Application to AES. In G. Bertoni and B. Gierlichs, editors, *Fault Diagnosis and Tolerance in Cryptography – FDTC 2012*, pages 85–94. IEEE Computer Society, 2012.
13. D. Mukhopadhyay. An Improved Fault Based Attack of the Advanced Encryption Standard. In B. Preneel, editor, *AFRICACRYPT 2009*, volume 5580 of *LNCS*, pages 421–434. Springer, 2009.
14. G. Piret and J.-J. Quisquater. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad. In C. Walter, Ç. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *LNCS*, pages 77–88. Springer, 2003.
15. H. Tupsamudre, S. Bisht, and D. Mukhopadhyay. Destroying Fault Invariant with Randomization – A countermeasure for AES Against Differential Fault Attacks. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *LNCS*, pages 93–111. Springer, 2014.
16. D. Wagner. Cryptanalysis of a Provable Secure CRT-RSA Algorithm. In B. Pfitzmann and P. Liu, editors, *ACM Conference on Computer and Communications Security – CCS'04*, pages 82–91. ACM Press, 2004.
17. 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 *LNCS*, pages 53–61. Springer, 2006.
18. 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 K. Kim, editor, *Information Security and Cryptology – ICISC 2001*, volume 2288 of *LNCS*, pages 397–413. Springer, 2001.

A Probability of Success of Attack 1

The success of Attack 1 depends on the chances for the attacker to fault the increment of i in the loop corresponding to the last redundant round execution. Let us denote by e_1 the event of faulting the last redundant round during the q-th loop. The probability $\mathcal{P}(e_1)$ is thus the probability of having a bit-string $rstr$ that contains 20 "1" on the first $q-1$ positions, one bit set on the q-th position and a last sub-string with only one bit set on the last $t-q$ positions. The corresponding number of such sub-strings being equal to $\binom{q-1}{20}$, $\binom{1}{1}$ and $\binom{t-q}{1}$ respectively, this leads us to $\binom{q-1}{20}\binom{t-q}{1}$ exploitable $rstr$ strings.

By dividing this value by the number of possible $rstr$ strings, we obtain the probability $\mathcal{P}(e_1)$:

$$\mathcal{P}(e_1) = \frac{\binom{q-1}{20}\binom{t-q}{1}}{\binom{t}{22}} \ . \tag{7}$$

As described in Appendix E, we then compute by using Equation (20) the probability to obtain at least one useful faulty ciphertext by repeating the fault injection r times.

B Probability of Success of Attack 2

Let us evaluate the probability that the event e_2 of obtaining a useful faulty ciphertext by setting to zero the variable λ at Step 5 of Algorithm 1 happens. The probability $\mathcal{P}(e_2)$ corresponds to the probability of obtaining a string $rstr$ that has 21 bits set on the first $q-1$ positions, a "1" on the q-th position and only "0"'s on the last $t-q$ positions. As we have done in Appendix A, we compute this probability as the number of such strings divided by the total number of possible $rstr$ strings. As there is only one possibility that the last $t-(q-1)$ bits of $rstr$ are exactly "$1\,0\cdots0$", we thus obtain:

$$\mathcal{P}(e_2) = \frac{\binom{q-1}{21}}{\binom{t}{22}} \ , \tag{8}$$

As described in Appendix E, we then compute by using Equation (20) the probability to obtain at least one useful faulty ciphertext by repeating the fault injection r times.

C Probability of Success of Attack 3

Let us denote by e_3 the event that a random byte error disturbs the string $rstr$ such that it contains only 21 or 20 "1". To evaluate the probability $\mathcal{P}(e_3)$ that the event e_3 occurs, let us assume for the sake of simplicity that the attacker disturbs the last byte B of $rstr$ which corresponds to a random byte fault model. By firstly evaluating the case 21, we observe that the probability that a bit-string

has exactly 21 bits set on the first $t-8$ positions and the remaining "1" in one of the last 8 positions is:

$$\mathcal{P}(HW(B)=1) = \frac{\binom{t-8}{21}\binom{8}{1}}{\binom{t}{22}} \ , \tag{9}$$

where we denote by $HW(B)$ the Hamming weight of the byte B. Equation (9) corresponds to the probability that the last byte of $rstr$ has an Hamming weight equal to 1. By summing the corresponding probabilities for all the Hamming weights between 1 and 8 we obtain the probability that the last byte of $rstr$ has an Hamming weight greater than zero:

$$\mathcal{P}(HW(B)>0) = \sum_{i=1}^{8} \frac{\binom{t-8}{22-i}\binom{8}{i}}{\binom{t}{22}} \ . \tag{10}$$

Now, let us compute the probability of injecting a random error on a byte of Hamming weight i such that the byte contains only $i-1$ "1" after the disturbance. We thus count for each possible value of B how many 8-bit values e exist such that $HW(B \oplus e) = HW(B) - 1$. This corresponds to the number of possible errors setting to "0" j bits "1" while setting to "1" $j-1$ bits "0". Afterwards we divide the result by the number of possible values for the error e:

$$\begin{aligned} \mathcal{P}(HW(B \oplus e) &= HW(B)-1 | B) \\ &= \frac{\sum_{j=1}^{HW(B)} \binom{HW(B)}{j}\binom{8-HW(B)}{j-1}}{255} \ . \end{aligned} \tag{11}$$

This corresponds to the probability that $HW(B \oplus e) = HW(B) - 1$ by injecting a random error e on a random 8-bit value B.

By combining the two probabilities above, we obtain the probability that $rstr$ contains 21 "1" after a random error injection on the last byte of $rstr$:

$$\mathcal{P}(HW(B \oplus e) = 21) = \sum_{i=1}^{8} \frac{\binom{t-8}{22-i}\binom{8}{i}}{\binom{t}{22}} \sum_{j=1}^{i} \frac{\binom{i}{j}\binom{8-i}{j-1}}{255} \ . \tag{12}$$

For the case where $rstr$ contains only 20 "1", we use the same reasoning and we obtain:

$$\mathcal{P}(HW(B \oplus e) = 20) = \sum_{i=2}^{8} \frac{\binom{t-8}{22-i}\binom{8}{i}}{\binom{t}{22}} \sum_{j=2}^{i} \frac{\binom{i}{j}\binom{8-i}{j-2}}{255} \ . \tag{13}$$

Thus the total probability of disturbing the generation of one byte of $rstr$ such that it contains a total of 21 or 20 "1" is:

$$\mathcal{P}(e_3) = \sum_{i=1}^{8} \frac{\binom{t-8}{22-i}\binom{8}{i}}{\binom{t}{22}} \sum_{j=1}^{i} \frac{\binom{i}{j}\binom{8-i}{j-1}}{255} + \sum_{i=2}^{8} \frac{\binom{t-8}{22-i}\binom{8}{i}}{\binom{t}{22}} \sum_{j=2}^{i} \frac{\binom{i}{j}\binom{8-i}{j-2}}{255} \ . \tag{14}$$

As described in Appendix E, we then compute by using Equation (20) the probability to obtain at least one useful faulty ciphertext by repeating the fault injection r times.

D Probability of Success of Attack 4

In the following we denote by e_4 the event that the error e is injected after a cipher round and is such that $q \oplus e > t$. In order to evaluate the probability $\mathcal{P}(e_4)$ we need to compute:

- the probability that the error e leads to $q \oplus e > t$,
- the probability that the attacker disturbs the algorithm after a cipher round and not after a redundant or dummy round.

For the first probability, without loss of generality, we assume that q is coded over one byte which should be the case in practice. We thus obtain that the probability of injecting an 8-bit error e such that $q \oplus e > t$ depends only on t and is given by:

$$\mathcal{P}(q \oplus e > t) = \frac{2^8 - t}{2^8} \ . \tag{15}$$

In order to evaluate the second probability we remark that it is equivalent to the probability that the string $rstr$ contains two or three "1" in the first q positions. We recall that $rstr$ is a string with 22 "1" at most. Thus the number of possible strings $rstr$ with only two "1" in the first q positions is:

$$\binom{q}{2}\binom{t-q}{20} \ . \tag{16}$$

Summing Equation (16) to the number of possible strings $rstr$ with only three "1" in the first q positions we obtain the number of favorable cases for the attacker:

$$\binom{q}{2}\binom{t-q}{20} + \binom{q}{3}\binom{t-q}{22-3} \ . \tag{17}$$

By dividing by the total number of possible $rstr$ strings we thus obtain the probability that the algorithm has executed only one cipher round after q rounds:

$$\mathcal{P}(HW(rstr[1,\ldots,q]) \in [2,3]) = \frac{\binom{q}{2}\binom{t-q}{20} + \binom{q}{3}\binom{t-q}{19}}{\binom{t}{22}} \ , \tag{18}$$

where $rstr[1,\ldots,q]$ denotes the sub-string of $rstr$ between the first and the q-th position. By combining the two probabilities we obtain:

$$\mathcal{P}(e_4) = \frac{2^8 - t}{2^8} \sum_{i=2}^{3} \frac{\binom{q}{i}\binom{t-q}{22-i}}{\binom{t}{22}} \ , \tag{19}$$

which corresponds to the probability that the algorithm returns an exploitable faulty ciphertext by injecting a random error after q rounds.

As described in Appendix E, we then compute by using Equation (20) the probability to obtain at least one useful faulty ciphertext by repeating the fault injection r times.

E Attack Repetition Probability

For each attack, we denote by $\mathcal{P}(e_i)$ the probability that event e_i occurs. By assuming that $\mathcal{P}(e_i)$ is independent for each execution we can compute the probability of getting at least one useful faulty ciphertext by repeating the fault injection r times as:

$$\mathcal{P}_r = 1 - (1 - \mathcal{P}(e_i))^r \ . \tag{20}$$