SCIENCE CHINA
Information Sciences

• LETTER •

May 2021, Vol. 64 159101:1–159101:3
https://doi.org/10.1007/s11432-018-9797-9

Do multiple infections lead to better security? A new study on CHES 2014 infective countermeasure

Jingyi FENG[1,2,3], Hua CHEN[1*], Weiqiong CAO[1,3], Limin FAN[1], Wei XI[4] & Dengguo FENG[1,2]

[1]*Trusted Computing and Information Assurance Laboratory (TCA), Institute of Software, Chinese Academy of Sciences, Beijing* 100190, *China;*
[2]*State Key Laboratory of Cryptology, Beijing* 100878, *China;*
[3]*University of Chinese Academy of Sciences, Beijing* 100049, *China;*
[4]*Electric Power Research Institute, China Southern Power Grid, Guangzhou* 510080, *China*

Received 21 November 2018/Accepted 20 February 2019/Published online 16 March 2021

Citation Feng J Y, Chen H, Cao W Q, et al. Do multiple infections lead to better security? A new study on CHES 2014 infective countermeasure. Sci China Inf Sci, 2021, 64(5): 159101, https://doi.org/10.1007/s11432-018-9797-9

Dear editor,

Implementation attacks such as fault attack (FA) [1] and side-channel analysis (SCA) [2] are significant threats to the cryptographic device security. Infective countermeasure is a promising strategy to resist FAs. Its principle is to scramble the data path of the encryption and make the faulty ciphertext useless in the key retrieval. The scrambling here is called infection. It can be executed either once at the end of the algorithm or multiple times between different encryption operations.

The CHES 2014 infective countermeasure [3] is proved secure against the intermediate-oriented single fault attack. It adopts multiple infections. However, whether its multiple infections could lead to better security or result in additional vulnerabilities in terms of other implementation attacks has not been studied.

In this study, we make a new analysis of CHES 2014 infective countermeasure and pay particular attention to its multiple infections triggered by the random fault injection. Based on the vulnerability that the countermeasure infects different intermediates with an identical parameter, encrypts the parameter with the secret keys and outputs the parameter as the final ciphertext, we develop two new attacks. The first one is a double-fault attack. The second one is an FA-SCA combined attack. The attacks promote the key retrieval efficiency and accuracy. They show that the FA-resistance of the countermeasure does not increase with the number of infections. Even worse, the improper setting of multiple infections could benefit other attacks.

CHES 2014 infective countermeasure. Algorithm 1 takes AES-128 as an example to explain how it works. The algorithm consists of 3 kinds of iterations: cipher, redundant, and dummy. In the beginning, the plaintext P is assigned to both the cipher and redundant state, R_0 and R_1. The secret parameter β is assigned to the dummy state R_2. Then, the cipher, redundant and dummy rounds are executed in a non-deterministic order according to a random string rstr. Note that the redundant round is always executed before its corresponding cipher round. The iteration contains 3 steps: AES round computation, consistency check, and fault infection. AES round computation (line 8) consists of ADK, SB, SR and MC, except in the last cipher/redundant round. k^ζ ($\zeta \neq 0$) used in the cipher/redundant rounds are derived from the secret master key. k^0 used in dummy rounds is selected according to (1), to make the round input equal to the round output.

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

In each cipher round, there is a consistency check between R_0 and R_1 to detect the fault in the cipher/redundant inter-

Algorithm 1 Infective countermeasure for AES [3]

Input: Plaintext P, round key k^j for $j \in \{1, \ldots, 10(11)\}$, dummy round parameters (β, k^0).
Output: Ciphertext $C = \text{AES-128}(P, K)$.
1: Cipher 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) = 20
4: **while** $q \leqslant t$ **do**
5: $\lambda \leftarrow \text{rstr}[q]$; // 0 implies a dummy round
6: $\kappa \leftarrow (i \wedge \lambda) \oplus 2(\neg\lambda)$;
7: $\zeta \leftarrow \lambda \cdot \lceil i/2 \rceil$; // ζ is actual round counter
8: $R_\kappa \leftarrow \textsc{RoundFunction}(R_\kappa, k^\zeta)$;
9: $\gamma \leftarrow \lambda(\neg(i \wedge 1)) \cdot \text{BLFN}(R_0 \oplus R_1)$;
10: $\delta \leftarrow (\neg\lambda) \cdot \text{BLFN}(R_2 \oplus \beta)$;
11: $R_0 \leftarrow (\neg(\gamma \vee \delta) \cdot R_0) \oplus ((\gamma \vee \delta) \cdot R_2)$;
12: $i \leftarrow i + \lambda$;
13: $q \leftarrow q + 1$;
14: **end while**
15: **return** R_0.

* Corresponding author (email: chenhua@tca.iscas.ac.cn)

mediate (line 9). Similarly, there is a check in each dummy round to detect the dummy intermediate fault (line 10). If there exists any inconsistency, the fault infection will be triggered. The cipher state R_0 will be assigned the value of R_2 (line 11), which will result in a new inconsistency between R_0 and R_1. In this case, one fault injection can lead to multiple infections on all the subsequent cipher states. After t-round iterations, R_0 is output as the final ciphertext.

Vulnerability after a random fault injection in a cipher or redundant intermediate. Among all the existing fault models, the random fault is the weakest one. A totally random unknown fault can hardly threaten the unprotected cipher. However, it could trigger multiple infections in the CHES 2014 infective countermeasure.

Consider the case that a random fault is injected in an intermediate of the i-th cipher or redundant round. According to Algorithm 1, the dummy state R_2 remains β. As a result:

• R_0 will be assigned β over and over in the subsequent infections;

• β will be encrypted for $10-i$ times respectively by the secret round keys $k^{i+1}, k^{i+2}, \ldots, k^{10(11)}$ in the following cipher rounds;

• β will finally be output as the ciphertext.

The above encryption is vulnerable. On the one hand, the output of β makes the AES round computation inputs of all the infected cipher rounds open to the adversary. Compared to the uninfected cipher round encryption with unknown input, it offers more information for the key retrieval.

On the other hand, if $i \leqslant 8$, the AES round computation with the known input β and the secret round key will be executed more than once in different cipher rounds. More infections happen, more round keys can be recovered. Since the round keys all contribute to the retrieval of the master key, the information that should be obtained from each round key can be reduced.

Besides, the countermeasure itself cannot resist the flow sequence-oriented fault [4] and SCAs [5]. Based on these facts, we put forward two attacks.

Double-fault attack. The attack combines the intermediate-oriented random fault injection and the flow sequence-oriented injection. It is feasible when (β, k^0) are reused in the encryptions of different plaintexts. Since β is reused, the faults in the cipher or redundant intermediate will make the infected cipher round inputs equal in different encryptions. The input β can be obtained from any of these encryptions. If the last infection operation of one encryption can be further skipped by an additional injection, the uninfected cipher state $R_0 = \textsc{RoundFunction}(\beta, k^{10(11)})$ can be obtained. Then, the secret master key can be worked out from β and R_0. Given the above, the attack requires 2 encryptions and 3 effective fault injections. The procedures are given as follows.

Step 1. Get β. In the first encryption, we inject a random fault into a cipher or redundant intermediate and obtain β from the ciphertext.

Step 2. Get the uninfected R_0 of the last cipher round. In the second encryption, we first inject a random fault into a cipher or redundant intermediate. Then, we inject another fault into the last cipher round to skip the infection and get the uninfected R_0 from the ciphertext. According to the infection in Algorithm 1 (line 11), there are 4 potential targets for the second injection:

• Inject a random fault into the variable i after the selection of the round key (line 7) and before the computation of γ (line 9), to make i odd;

• Inject a bit-flipping fault into the variable λ after the selection of the round key (line 7) and before the computation of γ (line 9), to make the value of λ equal to 0;

• Inject an instruction-skipping fault in the computation of γ (line 9) or inject a bit-flipping fault into γ right after this computation, to make the value of γ remain 0;

• Inject an instruction-skipping fault in the assignment of the cipher state (line 11), to make the value of R_0 remain unchanged.

Step 3. Recover the round keys (k^{10}, k^{11}) by solving the key schedule function and the last cipher round function $R_0 = \mathrm{SR} \cdot \mathrm{SB}(k^{10} \oplus \beta) \oplus k^{11}$.

Step 4. Recover the master key. The attack feasibility is analyzed in Appendix A. In this attack, the plaintexts of different encryptions are not required to be equal or known. Besides, it requires weaker fault models than the majority of the existing double-fault attacks with two identical faults.

FA-SCA combined attack. The attack combines the intermediate-oriented random fault injection and SCA. It is feasible when (β, k^0) are not reused in different encryptions. The fault injection makes the inputs of all the infected cipher rounds equal to β. Since β is distinct in different encryptions and can be obtained, all the round keys used in the infected cipher rounds are retrievable through correlation power analysis (CPA) [6]. Therefore, we perform standard CPAs on multiple round keys and integrate their results for the more accurate retrieval of the master key. The follows are the detailed attack, in which the variable with subscript i denotes the i-th byte of it.

Step 1. Measure the power consumption of the fault injected encryption and get β. Inject a random fault into a cipher or redundant round intermediate of an encryption. Measure the power consumption of this encryption and store it as a trace. Save the ciphertext as β. Repeat this step until the trace set is big enough for CPA.

Step 2. Uncover the order of rounds. For each encryption in Step 1, firstly work out k^0 with Eq. (1) and build the power consumption template for the dummy round with (β, k^0). Then, divide the measured trace into t segments and calculate their correlations to the template. Finally, take the most template-like $t-20$ segments as the dummy rounds, and take the others alternately as the redundant and cipher rounds.

Step 3. Recover the round key k^i through the modified CPA. Select two adjacent cipher rounds executed after the fault injections, for instance, the $(i-1)$-th and i-th rounds $(2 < i \leqslant 10)$.

• Perform the standard CPAs on k^{i-1} and k^i with the known round input β. Let w denote a candidate of the key byte k_j^I, where $w = 0, \ldots, 255$, $I = i-1, i$ and $j = 0, 1, \ldots, 15$. We save the correlation coefficient between the measured power consumption and the hypothetical one derived from (β_j, w) as $\rho_{k_j^I, w}$.

• Refresh the correlation coefficients and distinguish the candidate of k^i. According to the AES key schedule algorithm, the key bytes satisfy $k_j^i = G(k_j^{i-1}, k_h^I)$, where $I = i-1$ or i. We refresh the correlation coefficient for k_j^i's candidate w as

$$\hat{\rho}_{k_j^i, w} = \max_{u,v} \left\{ \left| \rho_{k_j^i, w} \cdot \rho_{k_j^{i-1}, u} \cdot \rho_{k_h^I, v} \right|^{\frac{1}{3}} \middle| w = G(u, v) \right\}. \quad (2)$$

The w which results in the maximum $\hat{\rho}_{k_j^i, w}$ is distinguished

as the retrieval result of k_j^i. Repeat this sub-step for all the bytes in k^i.

Step 4. Recover the master key. We conduct FA-SCA attack experiments in Appendix B. The attack has a better master key retrieval accuracy in the case of high noise compared to the existing SCA. In Appendix C, we give suggestions to improve the countermeasure.

Conclusion. This study makes a new analysis of the CHES 2014 infective countermeasure. We find that even though the infection is proved secure against the single fault injection, the improper setting of multiple infections could result in unexpected vulnerabilities in terms of other attacks. This study emphasizes the importance of paying particular attention to each newly introduced defensive operation.

Acknowledgements This work was supported by National Key R&D Program of China (Grant Nos. 2018YFB0904900, 2018YFB0904901), National Cryptography Development Fund (Grant No. MMJJ20170214), and National Cryptography Development Fund (Grant No. MMJJ20170211).

Supporting information Appendixes A–C. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Boneh D, DeMillo R, Lipton R. On the importance of checking cryptographic protocols for faults. In: Proceedings of International Conference on the Theory and Application of Cryptographic Techniques, Konstanz, 1997. 37–51
2 Kocher P, Jaffe J, Jun B. Differential power analysis. In: Proceedings of the 19th Annual International Cryptology Conference, Santa Barbara, 1999. 388–397
3 Tupsamudre H, Bisht S, Mukhopadhyay D. Destroying fault invariant with randomization. In: Proceedings of the 16th International Workshop on Cryptographic Hardware and Embedded Systems, Busan, 2014. 93–111
4 Battistello A, Giraud C. A note on the security of CHES 2014 symmetric infective countermeasure. In: Proceedings of the 7th International Workshop on Constructive Side Channel Analysis and Secure Design, Graz, 2016. 144–159
5 Cojocar L, Papagiannopoulos K, Timmers N. Instruction duplication: leaky and not too fault-tolerant! In: Proceedings of International Conference on Smart Card Research and Advanced Applications, Lugano, 2017. 160–179
6 Brier E, Clavier C, Olivier F. Correlation power analysis with a leakage model. In: Proceedings of the 6th International Workshop on Cryptographic Hardware and Embedded Systems, Cambridge, 2004. 16–29