A Comparative Cost/Security Analysis of Fault Attack Countermeasures

Tal G. Malkin[1], François-Xavier Standaert[1,2], and Moti Yung[1]

[1] Dept. of Computer Science, Columbia University
[2] UCL Crypto Group, Université Catholique de Louvain
{tal, moti}@cs.columbia.edu, fstandae@dice.ucl.ac.be

Abstract. Deliberate injection of faults into cryptographic devices is an effective cryptanalysis technique against symmetric and asymmetric encryption algorithms. To protect cryptographic implementations (*e.g.* of the recent AES which will be our running example) against these attacks, a number of innovative countermeasures have been proposed, usually based on the use of space and time redundancies (*e.g.* error detection/correction techniques, repeated computations). In this paper, we take the next natural step in engineering studies where alternative methods exist, namely, we take a comparative perspective. For this purpose, we use unified security and efficiency metrics to evaluate various recent protections against fault attacks. The comparative study reveals security weaknesses in some of the countermeasures (*e.g.* intentional malicious fault injection that are unrealistically modelled). The study also demonstrates that, if fair performance evaluations are performed, many countermeasures are not better than the naive solutions, namely duplication or repetition. We finally suggest certain design improvements for some countermeasures, and further discuss security/efficiency tradeoffs.

Keywords: Attacks and countermeasures in hardware and software.

1 Introduction

Fault attacks consist of forcing a cryptographic device to perform some erroneous operations, hoping that the result of that wrong behavior will leak information about the secret parameters involved. These techniques have been increasingly studied since the publication of Boneh, Demillo and Lipton in 1996 [9] in the context of public key cryptosystems, and its extension to the private key setting by Biham and Shamir [8]. They were improved thereafter by several different authors in various contexts (*e.g.* [7,17,27]). Two survey papers have recently described practical and algorithmic issues of these methods [3,13].

Countermeasures against fault attacks can be deployed in hardware or software and generally help circuits to avoid, detect and/or correct faults. Certain active protections use sensors and detectors to infer abnormal circuit behaviors. Passive protections such as randomization of the clock cycles or bus and memory encryption [10,14] may also be used to increase the difficulty of successfully attacking a device. However, in practice, most proposed schemes are

L. Breveglieri et al. (Eds.): FDTC 2006, LNCS 4236, pp. 159–172, 2006.

based on classical error-detecting techniques using space or time redundancies [5,6,16,20,19,21,22,23,32]. In this paper, we conduct a comparative study regarding these latest techniques, assessing their security and efficiency. We believe that while the original investigations are useful and inventive in many ways, the comparative perspective is valuable since it forces a more uniform and perhaps more realistic view of the effectiveness of the countermeasures, from both security and cost point of view. In particular, our findings underline that certain published countermeasures may not be sufficient to counteract fault attacks due to limited modelling (*e.g.* intentional malicious fault injection that are unrealistically modeled as random limited number of faults, more typical in non-malicious environments). We also point out that, if fair performance evaluations are conducted, many countermeasures are not better than the naive solutions, namely duplication or repetition. Finally, we discuss the resulting security *vs.* efficiency tradeoff in the general context of hardware implementations that our study implies.

The rest of this paper is structured as follows. Section 2 investigates error detection techniques based on the use of space redundancies, including parity checks and other codes. We discuss limitations of security models in certain countermeasure designs which lead to attacks and, when overcome, lead to efficiency overhead. Section 3 similarly discusses techniques based on repetition or duplication. We reveal certain design issues that need corrections and we essentially realize that these schemes tend to resemble the naive countermeasures. Our conclusions, outlining the usefulness of our comparative study are in Section 4.

2 Error Detection Techniques Using Space Redundancies

2.1 Description of a First Scheme

References [23,32] describe a solution for the low cost concurrent error detection in substitution-permutation networks. We briefly summarize the proposed schemes in this section. For clarity purposes, we target the AES Rijndael [11].

A round of an unprotected block cipher implementation is represented in Figure 1. **S** blocks, representing non-linear substitution boxes (*i.e.* SubBytes in Rijndael), are followed by a linear diffusion layer (*i.e.* ShiftRows and MixColumns in Rijndael) and a bitwise key addition. The basic purpose of the countermeasure is to add a parity bit to the scheme in order to track errors during the execution of the

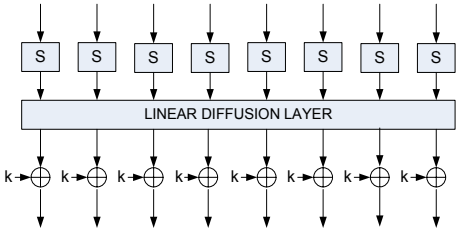

Fig. 1. Block cipher round without error check

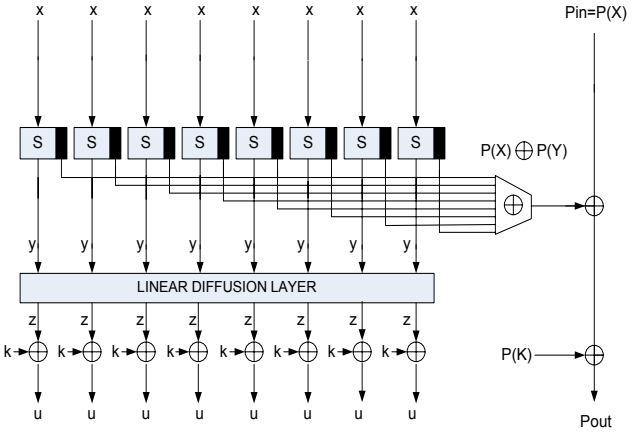

Fig. 2. Block cipher round with error check

algorithm. A single block cipher round with concurrent error check is represented in Figure 2 and the different steps of the error check are as follows.

1. **Computing the input parity.** The parity of the 128-bit input, denoted as Pin, is determined by a tree of XOR gates. This parity is computed once at the beginning of the algorithm.
2. **Parity modification according to the S-boxes.** An output bit is added to the S-boxes in order to implement the XOR of the parity of all S-boxes input bits with the parity of all S-boxes output bits, denoted as $P(X) \oplus P(Y)$. The value of this additional output bit can be determined from the truth table of the original S-box. It is represented as a black box in Figure 2.
3. **No parity modification according to the diffusion layer.** As detailed in [23,32], the linear substitution layer of Rijndael does not involve any modification of the previously defined parity. It is obvious for ShiftRows which only permutes the bytes of the state and does not affect their values. For MixColumn, it is observed that, due to the linearity of the transform, it does not alter the parity when the 32-bit columns are considered.
4. **Parity modification according to the key addition.** Since a 128-bit round key is bitwise XORed with the output of the diffusion layer, the input parity has to be modified by the parity $P(K)$.
5. **Output parity checking.** The parity of the actual outputs finally has to be compared with the modified input parity of the round.

According to the original paper, the proposed step-by-step parity modification overcomes the high diffusion of faults in block ciphers. Namely, a local fault detected within a processing step by parity checking of this processing step outputs will also be detected by comparing the modified parity of the round outputs. As an illustration of the technique, let us consider an input X with correct parity $P(X)$ and assume that a single bit fault occurs on this value of X, producing new intermediate values X^*, Y^*, Z^*, U^*.

First, the parity will be modified as follows:

$$Pout = P(X) \oplus P(X^*) \oplus P(Y^*) \oplus P(K)$$

Then, computing the output bits parity, we find:

$$P(U^*) = P(Z^*) \oplus P(K) = P(Y^*) \oplus P(K)$$

It is clear that the parities will only be equal if $P(X) = P(X^*)$, therefore allowing to detect the fault at the end of the round. Similarly, a single bit fault introduced after the S-boxes will cause:

$$Pout = P(X) \oplus P(X) \oplus P(Y) \oplus P(K) = P(Y) \oplus P(K)$$

This is because the parity $P(Y)$ is computed independently of the value of Y. Also, we have:

$$P(U^*) = P(Z^*) \oplus P(K) = P(Y^*) \oplus P(K)$$

Again the output parities will allow to detect the fault, and so will be for faults introduced after each processing unit of the block cipher. Although it is clear that multiple faults of even order will not be detected by such a scheme, the authors argue that, according to [26], the probability of 1-bit, 2-bit, 3-bit and 4-bit errors is respectively approximated by 85%, 10%, 3% and 1% in combinatorial logic circuits. It is therefore concluded that the error-correcting scheme allows to prevent most practical attackers, with a low hardware overhead.

2.2 Security of the Presented Scheme

Before discussing the presented countermeasure, let us first emphasize that, from an algorithmic point of view, the number of faults necessary to mount a successful attack has been dramatically reduced during the last years. In particular, it has been shown in [27] that the AES Rijndael can be corrupted with only two faulty ciphertexts. As a very straightforward consequence, a protection detecting only 85% of the injected faults is clearly not enough. Moreover, considering single-bit faults only is certainly not a conservative approach, as multiple-bit faults start to be a concern in very deep submicron technologies. Recent experiments have notably shown that high-energy ions can energize two or more adjacent memory cells in a circuit [15,28].

Anyway, in practice, it is unlikely that the mentioned experiments (*i.e.* evaluations of fault occurrences due to radiations effects) correctly model the behavior of a malicious insider. In particular, there are at least two parameters missing in the previous analysis, namely time and space localization, that may enhance the attacker capabilities to much more precision than unintended radiation effects.

Starting with time localization, it is clear that being able to induce a single-bit fault twice during a round function will simply bypass the previous countermeasure. Choosing the time at which the fault occur can be done by using side-channel information to monitor the progress of the algorithm. As present

pulse generators allow to deal with high frequencies, it is virtually possible to insert a fault anytime during a cryptographic computation.

Similarly, being able to induce single faults in different nodes of an implementation also bypass a single-bit parity check. Choosing the location of the fault can be done if light [31] or electromagnetic [29] induction are considered. These techniques have been proven very efficient to force low cost faults in cryptographic devices. More expensive techniques are susceptible to be even more powerful.

As a consequence, the fault detection technique in Section 2.1 is practically insecure as soon as real attacker capabilities are considered. This discussion also suggests that resistance against faults attacks involve higher constraints than usually required for integrated circuits. In particular, multiple bit faults have to be taken into account, as well as space and time localization.

2.3 Description of Improved Schemes

From the previous descriptions, there are two basic reasons making the countermeasure in [23,32] susceptible to multiple-bit faults: (1) only one parity bit is used, (2) parity codes are linear. Both reasons involve simple extensions in order to improve the detection capabilities of the method. In this section, we discuss these improvements of the original scheme and their additional cost[1].

1. Using more parity bits is suggested and implemented in [5] in order to improve multiple-bit faults detection. Simple arguments allow to evaluate the effect of such a countermeasure if the faults are uniformly distributed. For example, let n be the number of parity bits used, the probability that a double fault affects twice the same parity bit is:

$$P = \frac{n}{\binom{n+1}{2}} = \frac{2}{n+1} \tag{1}$$

[5] proposes one parity bit per byte for Rijndael, which yields $P = 0.12$.

Again, from a simple probabilistic point of view, the proposed improvement is not sufficient to reject all attackers. Moreover, it is likely that multiple faults will not be uniformly distributed, as multiple-bit faults usually target adjacent memory cells. As a consequence, the probability of masked errors (*e.g.* double faults occurring in the same byte) will actually be higher than predicted.

Regarding the additional cost for AES implementations, the proposal involves more hardware overhead as there are more parity bits, but also because the parities are now affected by MixColumn, which involves the need of parity predictors for this transform as well. These overheads are summarized in Table 1.

[1] Note that making the parity checks only once a round does not affect the fault coverage. As suggested in Section 2.1, what is detectable inside the round is also detectable at its output. As a consequence, the use of more parity checkers only affects the detection latency and may not be considered as a relevant improvement.

Finally, let us remark that using pipelined implementations (*i.e.* dealing with multiple inputs in parallel) is another solution to decrease the probability of (1). Double masked errors then have to affect twice the same parity bit and text.

2. Using non linear robust codes is another solution proposed in [19,20,24] to obtain good resistance against single and multiple fault errors. For this purpose, the authors use a much more restricting fault model where faults are uniformly distributed throughout the circuit and the expected number of faults (*i.e.* fault multiplicities) is proportional to the number of gates in the circuit. Two proposals are actually considered.

In the first one [19], the AES Rijndael is divided into two blocks: linear and non-linear, where the non-linear block only consists in the multiplicative inverse of the Rijndael S-box. The non-linear code is simply represented in Figure 3 and computes the product of two inverses X and Y. In order to reduce the area overheads, it is proposed to check only a few bits (typically 2) of the result. Then, for the linear-part, every column of the AES is associated with an 8-bit

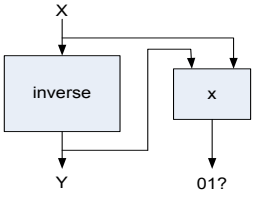

Fig. 3. Multiplicative inverse with error check

parity, namely the XOR between the 4 bytes of the column. It yields a 32-bit redundancy for the complete algorithm, which is computed independently, as the S-boxes parities in Section 2.1. The fault coverage of this scheme is contrasted. On the one hand, the non-linear part allows good detection of multiple faults, while low-order faults can clearly be masked because of the 2-bit comparison. On the other hand, the linear part suffers from the same problems as the previous linear schemes for the detection of higher-order faults. Globally, it is conjectured that the scheme only provides good error detection for faults with high multiplicities. The hardware overheads of the proposal are again summarized in Table 1. Note that [19] requires the S-box inverters and affine transforms to be implemented independently, while hardware implementations frequently combine both transforms in one single RAM block.

In the second proposal [20], a robust non-linear code is described, based on the addition of two cubic networks, computing $y(x) = x^3$ in $GF(2)^8$, to the previous linear scheme. The method allows to produce r-bit signatures to detect errors. It is shown that the fraction of undetectable errors is proportional to 2^{-2r}. Although the proposal offers a good fault coverage, its actual implementation is a real concern as the ratio throughput/area (a usual estimator of hardware efficiency) is decreased by a factor of two. As a consequence, the solution cost is somewhat comparable to duplication, which also has good non-linear properties

and therefore provides good fault coverage. Note finally that non-linear robust codes have been additionally discussed in [24] and the question to know if they can lead to more efficient implementations is open.

2.4 Summary of the Results

We have investigated 5 recent countermeasures against fault attacks, based on the use of space redundancies. Those are summarized in Table 1. The first two ones use an unrealistic fault model, considering single faults only, and may not be considered as sufficient to protect against a malicious attacker. [5] proposes to use more parity bits to improve their fault coverage, but faults of even order may still be masked with non-negligible probability.

Table 1. Space redundancy based techniques

Ref.	Method	Sin. fault detection	Mul. fault detection	Area overhead	Delay overhead	Thr. overhead	Thr./Area overhead
[23,32]	single parity bit	yes	no	+7.4%	+6.4%	-	-
[5]	multiple parity bits ($n = 16$)	yes	double faults masked with $P \propto \frac{2}{n+1}$	+20%	-	-	-
[19]	linear + non-linear codes	weak	good	+35%*	-	-	-
[20]	non-linear r-bit codes ($r = 28$)	good, missed with $P \propto 2^{-2r}$	good, missed with $P \propto 2^{-2r}$	+77%	+15%	-13%	-51%

The last two ones use a much more restrictive fault model, but only [20] provides good error detection properties against faults of all multiplicities. For this last scheme, the hardware overhead is comparable to duplication, as the ratio throughput/area has been divided by two. Remark that the objective of this table is only to summarize the results, not to provide fair comparisons between the different proposals. As a matter of fact, the area overhead is a function of the hardware cost of the unprotected primitive and, for example, [19,20] are low cost architectures compared to the ones used in the parity code papers. As a consequence, their overhead in % are higher.

3 Error Detection Using Repetition and Duplication

The previous section underlined that error-detection techniques based on space redundancies become as expensive as duplication if realistic attackers are considered. As a consequence, it is natural to investigate how codes based on repetition

or duplication can be used to improve the security of cryptographic devices. For this purpose, we start with some precisions about our model.

(1) We consider a n-bit block cipher, with q rounds independently implemented. (2) We assume that the error detection can be performed at three different levels: algorithm-level, round-level or operation level. Working at one level involves that the observed level is performed in at least one clock cycle, as its result has to be stored and compared. (3) In operation level detection schemes, we denote the number of operations considered per round as p. (4) The error detection latency only depends on the detection level. (5) Depending on the detection level, the codes have different non-linearity properties. However, as we perform n-bit comparisons, we assume that the error miss rate is 2^{-n} for all levels.

In general, the performance reduction in repetition or duplication schemes has two parts. One corresponds to the comparators required to check the validity of intermediate values. It is inversely proportional to the detection latency, as illustrated in Table 2, where τ denotes the timing function[2]. The other one corresponds to the repetition or duplication itself and directly affects the implementation throughput or area. Namely, repetition codes will cause a -50% reduction of the throughput while duplication will require +100% additional hardware. Regarding their detection properties, both solutions are not equivalent, as *repetition codes only allow to detect temporary (or soft) faults while duplication also allows to detect permanent (or hard) faults.*

Table 2. Latency *vs.* additional resources tradeoff

Latency	Additional 1-bit comparators
τ(Algorithm)	n
τ(Round)	nq
τ(Operation)	npq

While these solution may be straightforwardly implemented, the next sections show that certain particular contexts allow to obtain the effects of repetition or duplication for less than their usual cost.

3.1 Description of a First Scheme

Reference [16] describes a solution for the low cost concurrent error detection in involutional block ciphers, exploiting the involution property to check if the condition $f(f(x)) = x$ is respected through the cipher. The authors argue that the scheme achieves close to 0% time overhead. In this section, we show that:

1. The proposal can be improved by modifying the comparison scheme.
2. The proposal can be extended to non-involutional ciphers.
3. The proposal is actually a kind of repetition code.

[2] Remark that the registers needed to store intermediate values are not considered as hardware overhead. We show in the next section that, if well chosen, they can be combined with the original implementation registers.

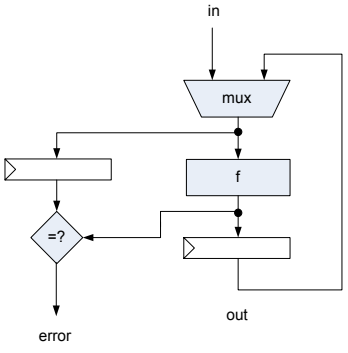

Fig. 4. Concurrent error detection for involutional functions [16]

The original error correction principle is represented in Figure 4. Reference [16] applies it to the Khazad block cipher [4], for which the non-linear and linear layer are involutional. First, let us observe that the area overhead can be straightforwardly reduced by changing the comparison scheme. Indeed, by comparing the function f's output with its following register output in place of with the multiplexor output, *we can avoid the comparison register*. It is represented in Figure 5, where we extend the scheme to a complete block cipher loop architecture.

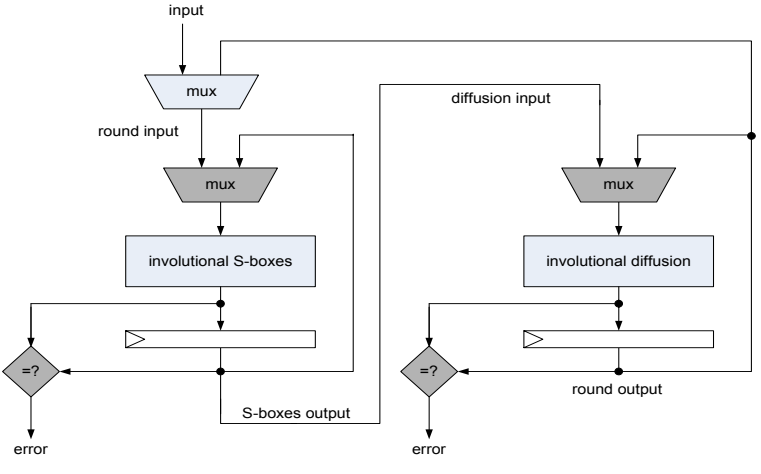

Fig. 5. Improved concurrent error detection for involutional rounds

Now, let us investigate the real time overhead of the countermeasure. For clarity purposes, we assumed that the work frequency was not affected by the comparison scheme. In Figure 5, we represented the original round operations in light grey and the overhead in dark gray. Removing the dark grey boxes, it is clear that the round can be performed in two clock cycles. It is basically a pipelined implementation dealing with two different plaintexts concurrently. Then, adding

the dark grey registers, the round operations (*i.e.* S-boxes and diffusion layer) will be used half the clock cycles for encrypting, the other half for checking the involution property. As a consequence, the proposed countermeasure will cause a -50% throughput overhead. We show that the proposed countermeasure is actually a repetition code, by extending it to non-involutional ciphers, as illustrated in Figure 6. Looking at the light grey boxes, the round is again divided

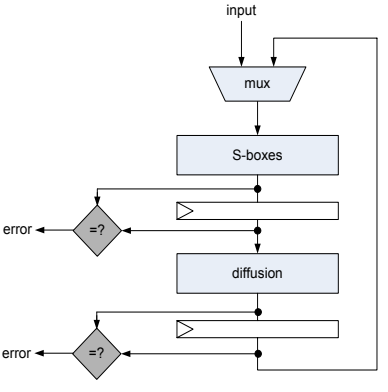

Fig. 6. Similar concurrent error detection for non involutional ciphers

into two operations and pipelined. Let us imagine an encryption mode where the same plaintext is encrypted twice and we add the comparison boxes. We can then detect errors as in Figure 5. The repetition is now obvious. The only differences between schemes 5 and 6 are:

1. The involutional scheme allows to detect permanent errors.
2. The involutional scheme needs two additional multiplexors.

At this point, it is not clear how the proposal can achieve a 0% time overhead and actually, this assumption is not generally true. However, considering the context of feedback encryption modes, the countermeasure of [16] becomes particularly interesting, as the pipeline cannot be used to deal with different plaintexts[3] but still allows to ensure error-proofness. Compared to a non-pipeline loop architecture, as usually required in feedback modes, we still require twice more clock cycles for one encryption, but it is likely that the clock frequency will be improved proportionally, so that the throughput will only slightly be affected. Note that this latter point is not a particular quality of the proposed technique, but a general rule in hardware design. A fair comparison of architectures for feedback encryption modes is represented in Figure 7, where we can clearly observe the tradeoff between the number of cycles increase for one encryption and the expected increase of clock frequency (because the critical path is reduced).

[3] It is mandatory to complete one plaintext encryption before starting the next one.

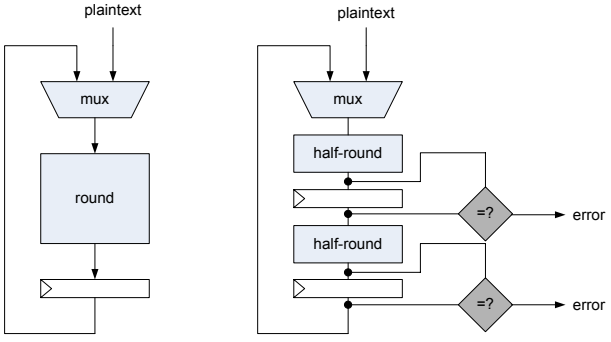

Fig. 7. Encryption with feedback, without and with error detection

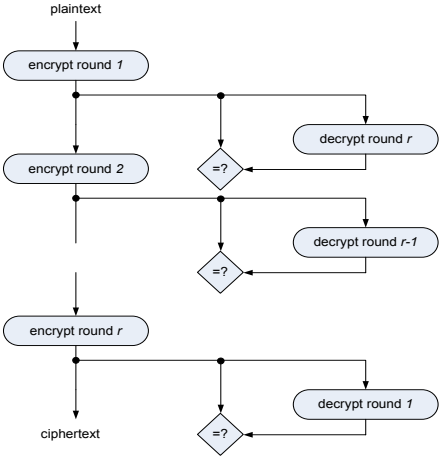

Fig. 8. Concurrent error detection using encryption/decryption designs

3.2 Another Proposal Equivalent to Repetition/Duplication

A very similar scheme has been presented in [21] for the concurrent error detection in symmetric block ciphers. It is based on exactly the same ideas as [16], in the more general context of non-involutional ciphers.

Basically, as the involutional property is not available, it is replaced by a design allowing to perform encryption and decryption. The error-detection principle is illustrated in Figure 8 and can be viewed as (1) duplication if the encryption and decryption blocks are independently implemented, or (2) repetition if the same hardware resources are used for encryption and decryption[4]. However, as in the previous section, the proposal gain particular interest in certain specific contexts. For example, if the cost of a decryption design is less than the one for

[4] For most algorithms, only a part of the resources can be shared between encryption and decryption. A perfect repetition scheme is only possible for involutional ciphers.

encryption[5], the solution has a lower cost than duplication. Also, in applications where encryption and decryption are necessary, but not concurrently, the actual performances will not be harmed by using the (otherwise unused) reverse operation for error detection.

4 Discussion and Conclusions

In this paper, we reviewed a certain number of countermeasures against fault attacks based on the use of space or time redundancies. It is shown that most of these countermeasures are either insecure, due to an unrealistic fault model, or their cost is close to duplication or repetition, excepted in certain particular implementation contexts (*e.g.* encryption with feedback, encryption/decryption designs). From an information theoretic point of view, this conclusion is close to the one in [25], stating that most of efficient concurrent error detection schemes exceed the cost of duplication. In general, improvements of these protections are possible in two different directions.

First, restricting the fault model could allow to design more efficient solutions, but it requires to consider the behavior of a malicious insider. Presently, only a few works have been published about actual methods for fault injections and more practical experiments are a preliminary step for such improvements. In particular, it is not clear that attacker capabilities could reasonably be reduced in terms of fault multiplicities or any other parameter. A conservative approach therefore requires to provide an equal security for faults of any multiplicity, with possible space and time localization.

Second, considering probabilistic fault detection is another usual alternative to design schemes less expensive than duplication. However, regarding the requirements of present attacks (*e.g.* in [27], Rijndael is corrupted with only two faulty ciphertexts), fault detection in cryptographic devices has particularly strong constraints. Therefore, this proposal has to be taken with care as faults have to be detected with high probability.

More specifically, this work:

1. Points out the unrealistic fault model used in certain recently proposed countermeasures [23,32].
2. Suggests that the actual cost of other countermeasures [19,20] are close to duplication if fair comparisons are performed.
3. Improves the comparison scheme of [16] and generalizes it from involutional block ciphers to all block ciphers.
4. Observes that countermeasures proposed in [16,21] are actual repetition codes used in a specific context.

As a consequence of these observations, theoretical solutions to the problem of fault attacks, as suggested in [12], no more appear as completely unpractical. Also, due to their good detection properties, non-linear robust codes, such as the ones in [19,20,24], would deserve further analysis to improve their hardware cost and see how better they can compare with duplication.

[5] This is very rarely the case in practice.

References

1. R. Anderson, M. Kuhn, *Tamper Resistance - a Cautionary Note*, in the proceedings of the USENIX Workshop on Electronic Commerce, pp 1-11, Oakland, CA, USA, November 1996.
2. R. Anderson, M. Kuhn, *Low Cost Attacks on Tamper Resistant Devices*, in the proceedings of the 5th International Workshop on Security Protocols, Lecture Notes in Computer Science, vol 1361, pp 125-136, Paris, France, April 1997, Springer-Verlag.
3. H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, C. Whelan, *The Sorcerer's Apprentice Guide to Fault Attacks*, IACR e-print archive 2004/100, http://eprint.iacr.org, 2004.
4. P.Barreto, V.Rijmen, *The KHAZAD Legacy-Level Block Cipher*, Submission to NESSIE project, available from http://www.cosic.esat.kuleuven.ac.be/nessie/
5. G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, V. Piuri, *Error Analysis And Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard*, IEEE Transactions on Computers, vol 52, num 4, pp 492-505, April 2003.
6. G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, *An Efficient Hardware-Based Fault Diagnosis Scheme for AES: Performance and Cost*, in the proceedings of DFT 2004, 9 pp, Cannes, France, October 2004.
7. I. Biehl, B. Meyer, V. Müller, *Differential Fault Analysis on Elliptic Curve Cryptosystems*, in the proceedings of Crypto 2000, Lecture Notes in Computer Science, vol 1880, pp 131-146, Santa Barbara, California, USA, August 2000.
8. E. Biham, A. Shamir, *Differential Fault Analysis of Secret Key Cryptosystems*, in the proceedings of Crypto 1997, Lecture Notes in Computer Science, vol 1294, pp 513-525, Santa Barbara, CA, USA, August 1997, Springer-Verlag.
9. D. Boneh, R. DeMillo, R. Lipton, *On the Importance of Checking Cryptographic Protocols for Faults*, in the proceedings of Eurocrypt 1997, Lecture Notes in Computer Science, vol 1233, pp 37-51, Konstanz, Germany, May 1997, Springer-Verlag.
10. E. Brier, H. Handschuh, C. Tymen, *Fast Primitives for Internal Data Scrambling in Tamper Resistant Hardware*, in the proceedings of CHES 2001, Lecture Notes in Computer Science, vol 2162, pp 16-27, Paris, France, May 2001.
11. J. Daemen, V. Rijmen, *"The Design of Rijndael. AES – The Advanced Encryption Standard,"* Springer-Verlag, 2001.
12. R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, T. Rabin, *Algorithmic Tamper-Proof Security: Theoretical Foundations for Security Against Hardware Tampering*, in the proceedings of TCC 2004, Lecture Notes in Computer Science, vol 2951, pp 258-277, Cambridge, MA, USA, February 2004, Springer-Verlag.
13. C. Giraud, H; Thiebauld, *A Survey on Fault Attacks*, in the proceedings of CARDIS 2004, Toulouse, France, August 2004.
14. J.D. Golic, *DeKaRT: A New Paradigm for Key-Dependent Reversible Circuits*, in the proceedings of CHES 2003, Lecture Notes in Computer Science, vol 2779, pp 98-112, Cologne, Germany, September 2003.
15. K. Johansson, M. Ohlsson, N. Blomgren, P. Renberg, *Neutron Induced Single-Word Multiple-Bit Upset in SRAM*, in IEEE Transactions on Nuclear Science, vol 46, num 7, pp 1427-1433, December 1999.
16. N. Joshi, K. Wu, R. Karry, *Concurrent Error Detection Schemes for Involution Ciphers*, in the proceedings of CHES 2004, Lecture Notes in Computer Science, vol 3156, pp 400-412, Cambridge, Massachusset, USA, August 2004.

17. M. Joye, A.K. Lenstra, J.-J. Quisquater, *Chinese Remaindering Based Cryptosystems in the Presence of Faults*, Journal of Cryptology, vol 12, num 4, pp 241-246, 1999, Springer-Verlag.
18. T. Karnik, P. Hazucha, J. Patel, *Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes*, IEEE Transactions on Secure and Dependable Computing, vol 1, num 2, April 2004.
19. M. Karpovsky, K.J. Kulikowski, A. Taubin, *Differential Fault Analysis Attack Resistant Architectures For The Advanced Encryption Standard*, in the proceedings of CARDIS 2004, Toulouse, France, August 2004.
20. M. Karpovsky, K.J. Kulikowski, A. Taubin, *Robust Protection against Fault Injection Attacks on Smart Cards Implementing the Advanced Encryption Standard*, in the proceedings of DSN 2004, 9pp, Florence, Italy, June 2004.
21. R. Karri, K. Wu, P. Mishra, Y. Kim, *Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers*, in IEEE Transactions on Computer-Aided Design, vol 21, num 12, pp 1509-1517, December 2002.
22. R. Karri, M. Gössel, *Parity-Based Concurrent Error Detection in Symmetric Block Ciphers*, in the proceedings of ITC 2003, pp 919-926, Charlotte, USA, September 2003/
23. R. Karri, G. Kuznetsov, M. Gössel, *Parity-Based Concurrent Error Detection of Substitution-Permutation Network Block Ciphers*, in the proceedings of CHES 2003, Lecture Notes in Computer Science, vol 2779, pp 113-124, Cologne, Germany, September 2003.
24. K.J. Kulikowski, M.Karpovsky, A. Taubin, *Robust Codes for Fault Attack Resistant Cryptographic Hardware*, in the proceedings of FDTC 2005, pp 2-12, Edinburgh, Scotland, September 2005.
25. S. Mitra, E.J. McCluskey, *Which Concurrent Error Detection Scheme ro Choose*, in the proceedings of the International Test Conference 2000, pp 985-994, October 2000, Atlantic City, NJ, USA.
26. V. Moshanin, V. Otscheretnij, A. Dmitriev, *The Impact of Logic Optimization on Concurrent Error Detection*, in the proceedings of the 4th IEEE International On-Line Testing Workshop, pp 81-84, July 1998.
27. G. Piret, J.-J. Quisquater, *A Differential Fault Attack Technique Against SPN Structures, With Applications to the AES and Khazad*, in the proceedings of CHES 2003, Lecture Notes in Computer Science, vol 2779, pp 77-88, Cologn, Germany, September 2003.
28. R. Reed, *Heavy Ion and Proton Induced Single Event Multiple Upsets*, in the proceedings of the IEEE Nuclear and Space Radiation Effects Conference, July 1997.
29. D. Samyde, S. Skorobogatov, R. Anderson, J.-J. Quisquater, *On a New Way to Read Data from Memory*, in the proceedings of the IEEE Security in Storage Workshop 2002, pp 65-69, Greenbelt, Maryland, USA, December 2002.
30. P. Shirvani, *Fault Tolerant Computing for Radiation Environments*, PhD Thesis, Center for Reliable Computing, Stanford University, June 2001.
31. S. Skorobogatov, R. Anderson, *Optical Fault Induction Attacks*, in the proceedings of CHES 2002, Lecture Notes in Computer Science, vol 2523, pp 2-12, Redwood City, CA, USA, August 2002, Springer-Verlag.
32. K. Wu, R. Karri, G. Kuznetsov, M. Goessel, *Low Cost Error Detection for the Advanced Encryption Standard*, in the proceedings of ITC 2004, Oct 2004.