2012 Workshop on Fault Diagnosis and Tolerance in Cryptography

A DFA on AES based on the entropy of error distributions

Ronan Lashermes[‡], Guillaume Reymond[‡], Jean-Max Dutertre[*], Jacques Fournier[‡], Bruno Robisson[‡], Assia Tria[‡]
[*][‡]*Département Systèmes et Architectures Sécurisés (SAS)*
[‡]*CEA-Leti*
Gardanne, France
Email: firstname.name@cea.fr
[*]*École Nationale Supérieure des Mines de Saint-Étienne (ENSMSE)*
Gardanne, France
Email: name@emse.fr

***Abstract*—Differential fault analysis (DFA) techniques have been widely studied during the past decade. To our best knowledge, most DFA techniques on the Advanced Encryption Standard (AES) either impose strong constraints on the fault injection process or require numerous faults in order to recover the secret key. This article presents a simple methodology based on information theory which allows to adapt the number of required faults for the analysis to the fault injection process. With this technique, the constraints on the fault model to recover the last round key are considerably lowered. Additionally, entropy is proposed as a tool to apprehend the most complex fault models in DFA. A practical realization and simulations are presented to illustrate our methodology.**

***Keywords*-Differential fault analysis; Shannon entropy; error distribution; Advanced Encryption Standard; information theory.**

I. INTRODUCTION

Differential fault analysis against secret key algorithms like the Advanced Encryption Standard (AES) has become a major concern linked to the implementation of such algorithms. This concern mainly originates from the complexity linked to fault attacks on the Integrated Circuits (ICs) used to run/implement cryptographic algorithms. The multiple aspects of this complexity are: the fault injection means used and their associated parameters; the analogue nature of the ICs under analysis and the effect induced by the different fault injection means on these ICs; the targeted cryptographic algorithm and the associated cryptanalysis methods. This complexity has led to the proposal of several fault analysis methods based on practical and/or theoretical fault models.

In this paper, we propose a generic methodology for performing differential fault analysis based on the entropy of the errors that are generated in an IC. We provide an example of its implementation against the AES. We demonstrate that, based on only three hypotheses on the kind of injected errors, the tenth round key of an AES encryption can be rapidly found. We first provide a short review of physical attack techniques. Then a quick overview of the AES algorithm and the practical attack set-up used are given. We next present the main ideas and motivations behind our approach before actually describing the proposed method. We then cover the issues linked to the practical implementation of our analysis methodology before proposing some enhancements. We also provide a discussion on the relevance of the error distribution model used, especially in the presence of countermeasures.

II. PHYSICAL ATTACKS

"Physical attacks" target the ICs running the cryptographic algorithms as opposed to cryptanalytic or mathematical techniques which search for vulnerabilities in the cryptographic algorithm itself. There are three main kinds of "physical attacks" on cryptographic devices. The first type, called "invasive attacks", covers all the techniques based on the analysis or modification of an IC's design by an invasive method such as micro-probing or the use of Focused Ion Beams. The second kind, called "side channel analysis" (SCA), exploits the fact that some physical values or "side channels" like the power consumption, the electromagnetic radiation or the duration of computation of an IC depend on its internal computations [1]–[3]. This threat can not be underestimated since these analyses can be quickly mounted with cheap equipment, without altering the physical integrity of the circuit. This dependency between the "side channels" and the internal computations can be analysed using mathematical tools like correlation [4], mutual information [5], variance [6] or entropy [7]. The third kind of attacks, called "fault analysis", consists in disturbing the circuit's behaviour in order to alter the correct progress of the algorithm. The faults are injected into the device by various means such as laser, clock glitches, spikes on the voltage supply or electromagnetic perturbations. There are three categories of fault attacks:

- "Algorithm modifications" consist either in reducing the ciphering complexity of the cryptographic algorithm [8] or in bypassing hardware or software protections.
- "Differential Fault Attack" (DFA), originally described in [9], [10], consists in retrieving the key by comparing the correct ciphertexts with faulty ones. A detailed comparison of DFA schemes against AES, for example, is given in [11]. They are usually distinguished by

the constraints on the fault models (timing and value of the faults, etc.) and the algorithmic complexity of the analysis. Most of those schemes are based on the hypothesis that the fault randomly modifies one byte [12]–[15] but a minority of DFA schemes consider on the contrary that the fault on the byte is not random [16], [17]. In [18] the notion of key recovery based on error distributions was firstly introduced. However our approach is different in various aspects: crypto-algorithm targeted, distinguishers used and we focus on the practical use of the analysis.

- "Safe-error" attacks consider fault models with near constant errors [19]–[21]. The attacker does not necessarily need the pair of correct and faulty ciphertexts but only the information about the behaviour of the chip (for example about the start of an alarm, the rising time of the alarm or a premature stop in computation) by a simple analysis of power consumption for example.

III. Clock glitch attack on AES

To validate the DFA's methodology proposed in section IV, practical fault injections were done on a hardware AES using clock glitches as the injection means.

A. AES presentation

The AES is a standard established by the NIST [22] for symmetric key cryptography. AES encryption is based on a few transformations (i.e. SubBytes, ShiftRows, MixColumns, AddRoundKey) used iteratively in "rounds" (Fig. 1). In this paper, we focus on the 128-bit key version of the AES. This version processes data blocks of 128 bits in ten rounds (after round 0). The round keys ($K1$ to $K10$) used during every round are calculated by a key expansion routine (not detailed in this paper). A data block is represented as a matrix of 4×4 bytes called the "State". We denote $M1$ to $M10$ the AES States at the end of each round.

B. Clock glitch attacks on AES

In order to validate the assumptions made in this paper, experiments have been conducted to gather practical data. A clock glitch generator similar to the one presented in [23] and [24] has been used to inject faults into the calculations of an unprotected AES implemented on a FPGA. Faults were induced by the violation of the timing constraints related to the clock period of an IC. This phenomenon is analysed in [25].

Shown on Fig. 2, the architecture used completes an AES round in one clock cycle. It features a long datapath from the output of the register to its input. For every bit in the cipher block, the propagation delay varies depending on the data processed. Thus the critical path may change at each clock cycle. For different encryptions with the same plaintext and key, we reduce progressively the clock period during the 9^{th} round by steps of 25 ps until it becomes too short to comply

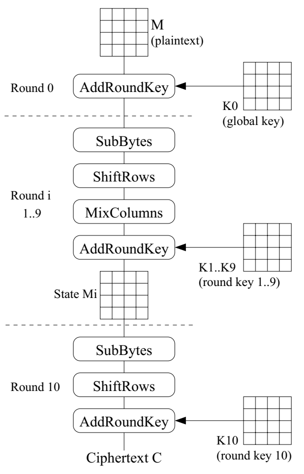

Figure 1. AES-128 encryption algorithm.

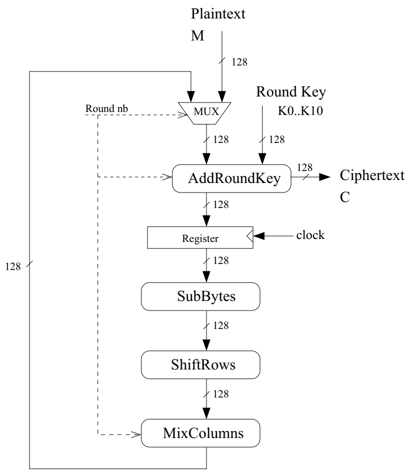

Figure 2. Hardware implementation used.

with the timing constraints. Hence, the value of $M9$ stored in the register is corrupted. This may be modelled with a bit-flip as shown in Fig.4. As a consequence, a progressive increase of the stress applied (i.e. the decrease, step by step, of the clock period) induces successively: (1) a first single bit fault located at the bit with the highest propagation delay ($\approx 90\%$ success rate); (2) a second and a third ($\approx 70\%$ and $\approx 50\%$ success rates respectively) fault appears; (3) more faults are induced at higher stress level. This provides the ability to tune the probabilities of occurrence of the injected faults. Since the propagation delays are data dependent, any change of the plaintext and/or of the key leads to a change in these propagation delays. As a consequence, the obtained faults for encryptions with different plaintexts but with the same stress are different in their location and number. The advantages hence provided by this attack scheme are: a fast fault injection technique ($\approx$ 100 faults per second); a good temporal resolution (100% faults are on $M9$); the possibility to tune the generated errors' distribution. Fig. 3

shows the example of a distribution of errors generated by the clock glitch attack with approximately 5,000,000 faults for 100,000 different plaintexts. In the figure, peaks match low Hamming weight errors (mainly single and double bits errors).

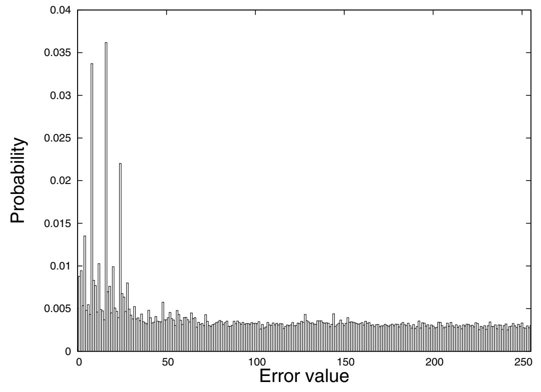

Figure 3. Distribution of injected errors using the clock glitch set-up

In the following, this distribution serves as an example to our methodology. However, one shall consider that the error distribution is unknow *a priori*.

IV. Proposed methodology

Our position of the problem is as follows: given the AES, given a fault injection means with a "characteristic" error distribution, by analysing a set of ciphertexts and their corresponding faulty ciphertexts we want to find a best candidate for the key (or round key) with some level of "confidence". Compared with existing DFA, our approach does not necessarily require fewer realizations. Yet by loosening the constraints on the fault model, analyses are more easily done in practice. We shall have three hypotheses below:

- **Hypothesis 1**: The faults generated are bit-flips. They can be modelled with an XOR operation with an error vector e (see Fig. 4). Set and reset faults would not work as efficiently as a bit-flip one in our scheme since it requires the use of a virtual model (cf Section VII).
- **Hypothesis 2**: The injected errors are not uniformly distributed (as shown for example on Fig.3). Supposing this condition, there are absolutely no constraints on the value of the injected error.
- **Hypothesis 3**: In the case of the AES, we shall suppose that such errors are injected on $M9$ as shown in Fig. 4.

For a matter of simplicity, the ShiftRows operation has been left out. Since there is no MixColumns operation on the last round, our analysis can focus individually on every byte of the State matrix.The generalisation to the whole State is straight-forward, so variables always represent bytes in the following. In practice all bytes can be faulted simultaneously.

Let i be the index associated with every plaintext. Using the same unknown key K, random plain texts M_i are

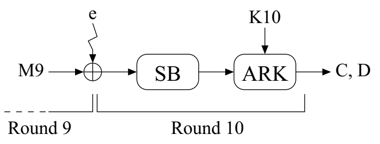

Figure 4. Attack scheme with fault injection on $M9$.

encrypted - with and without fault injection - to get the corresponding correct ciphertexts C_i and faulty ciphertexts D_i. For every i, the resulting pair of correct and faulted ciphertexts is here called a *realization*. For each encryption, two equations describe the realization:

$$C_i = K10 \oplus SB(M9_i) \quad (1)$$
$$D_i = K10 \oplus SB(M9_i \oplus e_i) \quad (2)$$

where $K10$ is the value of the 10^{th} round key and e_i is the injected error. For each realization, as $K10$ is unknown, all byte hypotheses s on the round key have to be tried. It is then possible to compute the value $e_{i,s}$ such that:

$$M9_i = SB^{-1}(C_i \oplus s) \quad (3)$$
$$e_{i,s} = M9_i \oplus SB^{-1}(D_i \oplus s) \quad (4)$$

For every s, an error value $e_{i,s}$ is obtained from every realization i. Then, a Realization/Key hypothesis table (or RK-table) is built (cf Table I).

Table I
Realizations / Key hypotheses (RK) table

	Key hypothesis s				
Realization i	0	1	2	$\cdots$	255
0	$e_{0,0}$	$e_{0,1}$	$e_{0,2}$	$\cdots$	$e_{0,255}$
1	$e_{1,0}$	$e_{1,1}$	$e_{1,2}$	$\cdots$	$e_{1,255}$
2	$e_{2,0}$	$e_{2,1}$	$e_{2,2}$	$\cdots$	$e_{2,255}$
...	...	...	...	...	...

The core of our methodology relies on the following properties (valid with our hypotheses):

- **Property 1**: Only one column in the *RK-table* corresponds to the correct key byte ($K10$) and the values $e_{i,s}$ in this column correspond to the errors that have been actually injected.
- **Property 2**: Given the properties of the AES SubByte operation, for every wrong key guess, the corresponding set of $e_{i,s}$ is quasi-random.

A discussion about *Property 2* is given in [11]. A condition for *Property 2* states that errors are independent of $M9$. In our experiment, we neglect the slight dependence between errors and $M9$ and we consider that the SBox perfectly randomizes errors in the case of a wrong key guess.

Based on those two properties, finding the correct key guess comes down to distinguishing between the columns of values $e_{i,s}$ that correspond to the wrong key guesses, which exhibit a random distribution, and the "single" column of errors that corresponds to the correct key guess - it has a non random, i.e. biased, distribution [18].

The "distinguisher" proposed here is the *Shannon entropy* $H(\mathcal{S}) = H(p_s)$ computed from a distribution p_s constructed from a vector $\mathcal{S}$ of size i_{max} for one key hypothesis s given by

$$H(p_s) = -\sum_{e=0}^{255} p_s(e) * \log_2 p_s(e) \tag{5}$$

where $p_s(e)$ is the probability of occurrence of value e (value between 0 and 255) for a given key hypothesis s. For a uniform distribution of errors, with a large number of realizations, the corresponding *Shannon entropy* shall tend to 8; whereas for a biased distribution, the *Shannon entropy* shall tend to $H_{inj} < 8$, the asymptotic entropy of the injection means. For a detailed discussion about Shannon Entropy and probability distributions, see Appendix A.

Applied to the *RK-table*, we calculate the *Shannon entropy* corresponding to each column and we shall have the following decision criterion:

$$\lim_{i_{max} \to \infty} H(\mathcal{S}) = 8 \Longleftrightarrow \text{Wrong key guesses} \tag{6}$$

$$\lim_{i_{max} \to \infty} H(\mathcal{S}) = H_{inj} < 8 \Longleftrightarrow \text{Correct key guess} \tag{7}$$

Note that in our approach, equation 7 holds because of *Hypothesis 2*.

V. Practical implementation issues

In this section, we address two practical issues concerning the methodology proposed in Section IV: how to define a quantitative method (i.e. a decision criterion) to distinguish between the two cases corresponding to equations 6 and 7; and how to define the minimum value of i_{max} for which the decision criterion holds. We shall then illustrate the implementation of our approach based on data collected from the experimental set-up described in Section III-B.

A. Quantitative decision criterion for sorting out key guesses

In practice, the number of realizations is limited, which means that the Shannon entropy of a random distribution of errors shall be strictly lower than 8. The decision criterion is defined to take this limiting factor into account. We thus have to study the entropy of a limited set of (error) bytes generated by a pseudo-random generator and establish a model. For a fixed number of realizations i_{max}, the data sets resulting from a pseudo-random data source do not all have the same entropy. We represent this entropy distribution which gives the probability of occurrence of each entropy according to a Gaussian law. This allows to quantify the level of *confidence*, for a fixed number of realizations, that a value of entropy comes from a pseudo-random generator or not.

As detailed in Appendix B, we precompute $\mu^{rand}_{i_{max}}$, the mean of a random distribution from a data set of size i_{max}, and $\sigma^{rand}_{i_{max}}$, the standard deviation of the same random data set. We can then evaluate the confidence cf that an entropy of value H is not random using equation 8.

$$cf_{i_{max}}(H) = \frac{\mu^{rand}_{i_{max}} - H}{\sigma^{rand}_{i_{max}}} \tag{8}$$

From there we define that, for a limited number of realizations i_{max}, a key is detected if its confidence is higher than a 6-sigma confidence, i.e. $K10$ is the only value for key hypothesis s that satisfies the equation 9.

$$cf_{i_{max}}(H(p_{K10})) > 6 \tag{9}$$

This 6-sigma value is the result of a trade-off between confidence and analysis efficiency. A lower confidence would make our analysis provide a key candidate sooner but with a higher false-positive probability. 6-sigma allows us to reach the same analysis efficiency as Giraud's one in Table II.

B. Discussion on the number of realizations i_{max}

In order to quantify the influence of the number of realizations on the calculated entropies associated to the columns of the *RK-table*, we ran simulations using the pseudo-random number generator of the Mono framework (www.mono-project.com) on an Opteron 1222 PC processor to generate 5000 data sets of every size ranging from 1 to 2999. From these sets, entropy distributions, mean values and standard deviations have been calculated for every data set size. In Fig. 5, the results of this simulation can be seen with the mean and the lower 6-sigma deviation from the mean. From the latter figure, we foresee that there must be a minimum number of realizations needed for a given injection entropy. For each possible entropy value (between 0 and 7.95), several random distributions of this given entropy were generated. They were used to create data sets of various sizes. With these sets we measured the average minimum size such that the entropy of the data set is lower than the 6-sigma boundary of the random model. The result is shown on Fig. 6. We have the average minimum data set size as a function of the injection entropy. We show with this Figure a direct link between the analysis efficiency and the entropy of the injection means. Hence we provide a metric to assess how efficient an injection means is to perform our attack.

In practice, in order to implement our entropy-based differential fault analysis, an attacker has to progressively increase the number of realizations while computing the entropies until one entropy reaches the 6-sigma boundary. This way, the attacker needs in average only the minimum data set size shown on Fig. 6 to find the key.

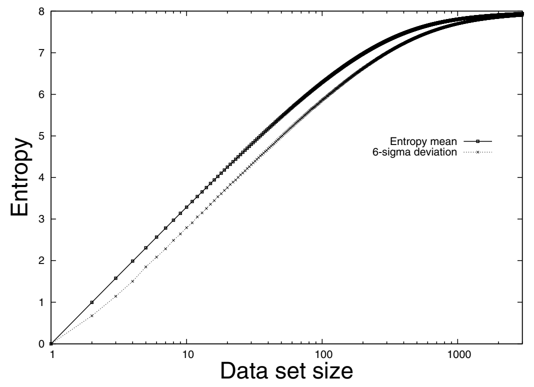

Figure 5. Entropies for a pseudo-random process along with the 6-sigma deviations for various sizes of data sets.

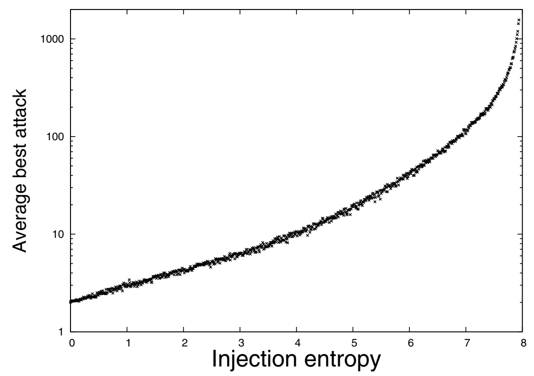

Figure 6. Average minimum number of faults needed to find the key for various entropies of the injection means.

C. Practical implementation using clock glitches on AES

Using the experiment presented in Section III-B, we used faults created when progressively reducing the clock period by 150 steps of 25 ps. We determined experimentally that the entropy H_{inj} of our physical attack is $H_{inj} \approx 7.76$.

On Fig. 7 are shown the entropies as a function of the key hypothesis for a data set of size 506. With a similar method, we created a data set of increasing size from 100 to 1000. Fig. 8 illustrates all entropies for all key hypotheses. We can see that with an injection process of entropy ≈ 7.76, 165 faults were needed in this particular example to find the key.

The presented analysis has interesting properties: no prior fault model is necessary but a bit-flip model. In particular, we do not need to know which faults are more likely to occur but only that such faults exist. An injection process with a high entropy can still allow to find the key. Moreover this method also allows the attacker to learn about the error distribution generated by the fault injection means. The main drawback of our approach described so far is that a larger amount of faults is needed to extract the key than existing DFA. For example, in a single-bit fault injection, we would need, on average, a minimum of 6.4 faults to find the correct key against only 2.24 with Giraud's DFA [16] (cf Table II). In the next section, we shall discuss how to improve this.

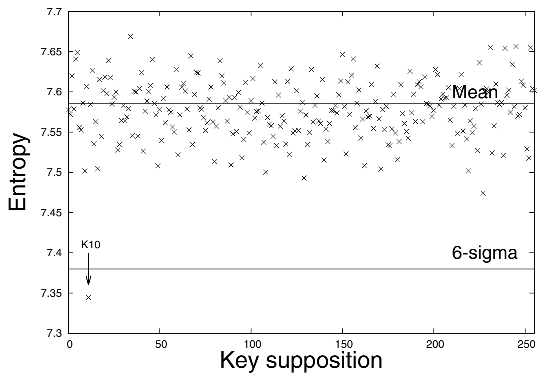

Figure 7. Entropies (with corresponding mean & 6-sigma deviation) for a data set of size 506.

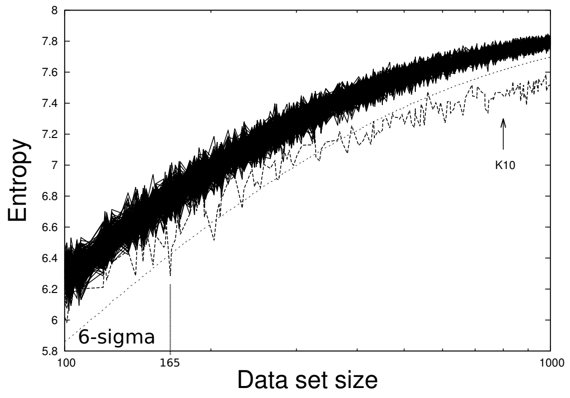

Figure 8. Entropies for all key guesses for data set size from 100 to 1000 (with the $K10$ & the 6-sigma boundary).

VI. Enhancement: model based approach

In order to increase the efficiency of our approach, we can use a predetermined model of the fault injection means and compare it with the measured data sets for each key hypothesis. The proposed model is based on the error distribution. If no prior model is known, it is possible to use the previously described attack using the Shannon entropy to recover the key and to construct the model which can in-turn be used in future differential analyses made on the same device or on devices of the same family.

A. Comparing two error distributions

Let t be the probability distribution of a given *model* which represents what the expected distribution should be. Let p_s be the measured error distribution for a given key hypothesis s. In order to include information about what fault is more or less probable than some other one, we shall use the concept of *Relative Entropy*, noted RH (also called Kullback-Leibler divergence), as defined by:

$$RH(p_s, t) = \sum_{e=0}^{255} p_s(e) \log_2 \left(\frac{p_s(e)}{t(e)} \right) \qquad (10)$$

A condition on $t(e)$ is that $\forall e, t(e) \neq 0$ in order to have the Relative Entropy properly defined. With respect to Shannon entropy, the template function $t(e)$ allows to introduce a weighting coefficient among the probabilities.

B. Illustration

As an example, we simulated the case where the injection means generate single-bit errors. We can then compare our analyses with Giraud's single-bit analysis [16] and the RLK (Roche-Lomné -Khalfallah) fault analysis [17]. We slightly modified the latter one to take into account a fault on $M9$ rather than on $K9$ and $K10$. The RLK analysis is particularly relevant as a comparison with our owns since the underlying principle which makes it works (detection of a non-random probability of error) is the same as in our approach.

We simulated a perfectly single-bit fault injection source in order to generate our faults. The template used for the Relative Entropy (RH) analysis in Equation 10 has been arbitrarily defined as follows: $t(e) = \frac{31}{256}$ for $e \in \{1, 2, 4, 8, 16, 32, 64, 128\}$, for the other e values, $t(e) = \frac{1}{7936}$. This is an approximation of the probability distribution where all single-bit faults have the same probability of occurrence, but respecting the condition $\forall e, t(e) \neq 0$. We simulated 4 different analyses: Giraud's, our's based on Shannon Entropy, our's using the Relative Entropy, and the RLK analysis. For all analyses but Giraud's[1], a 6-sigma condition is used to detect the key. With this condition, we found out the minimum number of faults necessary to detect the key. The results are displayed in Table II.

Table II
MINIMUM (AV.) NUMBER OF FAULTS FOR A SINGLE-BIT FAULT INJECTION.

Analysis name	Average minimum amount
Giraud's	2.24
Shannon entropy	6.41
Relative entropy	2.24
RLK's	8.95

We then performed the same analyses (but Giraud's) with the same RH template (a single-bit one) on data collected during the clock glitch attack described in Section III-B. The results are shown in the Table III. Please note that, in this experiment, the RLK analysis, similarly to the Shannon entropy approach, does not include *a priori* information on the fault injection process.

Table III
MINIMUM (AV.) NUMBER OF FAULTS FOR A CLOCK GLITCH FAULT INJECTION.

Analysis name	Average minimum amount
Shannon entropy	445
Relative entropy	359
RLK's	137

[1]since it is not a statistical analysis.

It is interesting to note that the Relative Entropy, a statistical distinguisher, is able to match Giraud's performances on extremely small data set (of size 2 or 3). The RLK is extremely efficient with our clock glitch attack. This is due to the highest peak present in the distribution (Fig. 3). If the peak were lower, more texts would be necessary since the performance of RLK mainly depends on the value of the highest peak.

C. Constructing the relevant error distribution template

In order to generate a relevant fault model, one can, as illustrated previously, build a theoretical one. A second alternative is to build one based on our Shannon entropy approach. A third alternative would be to profile the behaviour of the device under test while injecting faults during the tenth AES round: for example by injecting faults after the SubByte of round 10, the injected errors can be qualified by comparing the correct ciphertexts with the erroneous ones. This can work only at the condition that the injected errors do not solely depends on the timing of the injection. However this alternative has not been tested in practice.

VII. COUNTERMEASURES AND WRONGLY TIMED INJECTIONS

Up to now, our discussion has been based on measurements made on an unprotected AES using a fault injection means that provided precise timing. In this section, we discuss about how our approach can be extended when any of those two conditions is not present. In this case a larger number of realizations will be needed to extract the correct key.

From now on, we shall suppose that the targeted IC always gives a result for each encryption: when an error is detected either a correct ciphertext (the error has been corrected) or an erroneous one is returned. Moreover we suppose that there is always a certain proportion, strictly greater than zero, of errors passing through the protection undetected. In order to use our approach, we shall define a "virtual model" based on the fact that every fault on the ciphertext can be associated with an error value injected on $M9$. For example, suppose each time the countermeasure detects a fault, the chip outputs a random ciphertext.Then the difference between the random faulty ciphertext and the correct one can be associated with an error on $M9$ using equations 3 and 4.

A. Virtual model with result discrimination

Result discrimination is the ability for the attacker to know if the countermeasure has detected the fault (or if the injection was not correctly timed). If we have result discrimination, the virtual model only includes modifications due to the variations of the detection rate since we can filter all detected faults. Indeed, let $d(e)$ be the average detection rate of the error of value e ($0 \leq d(e) \leq 1$) and $p_{K10}(e)$ the probability that error e is injected by our physical injection means. We can define the virtual error distribution v as:

$$v(e) = \frac{p_{K10}(e)(1-d(e))}{\sum_{n=0}^{255} p_{K10}(n)(1-d(n))} = \frac{p_{K10}(e)(1-d(e))}{1-D} \tag{11}$$

where D is the global error detection rate:

$$D = \sum_{n=0}^{255} p_{K10}(n)d(n) \tag{12}$$

The collected erroneous ciphertexts correspond to errors which have not been detected, which means that we have to take into account the fact that there are errors that have been detected and filtered by the countermeasure. Hence in practice, we need more faults in the presence of the countermeasure ($\frac{1}{D}$ times).

If the distribution of the errors injected physically is uniform but not their associated detection rates, the countermeasure can allow a key recovery with the creation of a non-uniform virtual error distribution. In practice, such a situation can occur when we have countermeasures like one proposed by Bertoni *et al.* in [26] where an error detection code is used by adding one parity bit per byte of the State. This protection is able to detect all errors of odd parity and none of even parity. If we have result discrimination, such a scheme artificially filters some errors and not others, thus decreasing the entropy of the virtual model as shown in Table IV. Hence, for countermeasures to be efficient against our scheme, they should have uniform detection rates. Please note that Bertoni's countermeasure has been presented as an illustration even if it does not constitute an efficient protection against state-of-the-art DFA techniques.

B. Virtual model and random noise

If the attacker is not able to know if the countermeasure has detected a fault or if a wrongly timed injection occurs (e.g. before the SubByte in a round other than round 9) the error can be modelled as a random error at round 9. This is similar to a random noise added to the previous virtual error distribution. In the presence of a countermeasure, the new virtual error distribution w becomes:

$$w(e) = \frac{1}{256}D + p_{K10}(e)(1-d(e)) = \frac{1}{256}D + (1-D)v(e) \tag{13}$$

This is what we call the virtual error distribution without result discrimination, for which an example of error entropy is given in Table IV. An example of the resulting entropy values is shown on Table IV, with the corresponding distribution in Appendix C. One should note the entropy decrease when we have result discrimination.

The result discrimination feature can be achieved in practice with different methods such as the probing of the alarm signal, the measurement of characteristic data of the error detection circuit, etc. It is highly unlikely that the attacker can achieve total result discrimination in practice. In this case, the new virtual model is obtained as a combination of the two previous ones.

Table IV
ENTROPIES OF GENERATED ERRORS USING CLOCK GLITCHES FOR DIFFERENT AES IMPLEMENTATIONS WITH BERTONI'S SCHEME.

	Entropy
Unprotected AES	7.76
Virtual injection *with* discrimination (v)	6.86
Virtual injection w/o discrimination (w)	7.78

VIII. CONCLUSION

We presented a way to perform differential fault analysis based on the study of the entropy of the injected errors having a non uniform distribution. As an illustration, we focused on faults injected on the intermediate state matrix $M9$ of the AES. We used the Shannon entropy of the error distribution as the distinguisher and the *6-sigma confidence* as the decision criterion. We also showed how to find the minimum number of realizations needed to efficiently reach such a decision. We further proposed a refinement based on the knowledge of the model of the error distributions and how this can be quantitatively achieved using the notion of *relative entropy*. We finally discussed about how our approach is still relevant even in the presence of some countermeasures or in the presence of imprecisions on the injection tools used. To our best knowledge, this is the first DFA on AES which adapts its efficiency to the constraints on the fault injection process based on error distributions. It is no longer necessary to constraint faults to subsets such as single-bit errors or constant errors.

This work can be extended in different directions: applying the same methodology when the fault occurs at some other location in the AES algorithm, with other AES implementations, with other injections means (optical, EM, etc.). Finally the effects of existing masking schemes shall be investigated.

ACKNOWLEDGMENT

This work was partially funded by the French DGCIS (Direction Générale de la Compétitivité de l'Industrie et des Services) through the CALISSON 2 project. We would like to express our gratitude to H. Le Bouder, N. Moro and J.B. Rigaud for proofreading this paper.

APPENDIX A. SHANNON ENTROPY & ERROR DISTRIBUTION

Error distributions are used in order to extract information from the columns of the *RK-table*. Creating an error distribution from an error data set is very simple. If the function expressing the distribution is p_s ($p_s(e)$ is the probability of occurrence of error value e for the key hypothesis s) and the data set $\mathcal{S}$ is expressed as $(e_{i,s})_{0 \leq i < i_{max}}$ for a given hypothesis s, we can define:

$$p_s(e) = \frac{\#\{e_{i,s} | e_{i,s} = e, 0 \leq i < i_{max}\}}{i_{max}} \qquad (14)$$

with e ranging between 0 and 255 (all possible errors for one byte) and $\#$ denotes the cardinality of the set.

Shannon Entropy (denoted H) is used on the columns of the *RK-table* as a signal characterization tool. When applied on a byte distribution p_s, the Shannon entropy is defined as:

$$H(p_s) = -\sum_{e=0}^{255} p_s(e) * \log_2 p_s(e) \qquad (15)$$

As a notation trick, for a data set $\mathcal{S}$, we write $H(\mathcal{S})$ the Shannon entropy of the data set as the Shannon entropy of the byte distribution constructed from this set. The Shannon entropy represents the minimum average amount of information bits per symbol.

APPENDIX B. GAUSSIAN DISTRIBUTIONS

When using an injection source of pseudo-random errors, the measured entropy of a data set of limited size i_{max} is strictly lower than 8. Moreover this entropy is not constant when modifying the data set due statistical variations in the random draw. As an example on Fig. 9 are shown the various entropies measured for data sets of size $i_{max} = 1000$ as well as their probability of appearance.

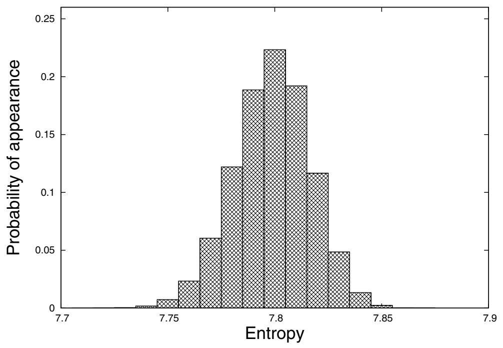

Figure 9. Distribution of the entropies for pseudo-random data sets of size $i_{max} = 1000$.

Because of the Gaussian shape of this distribution, it has been decided to model entropy variations of a given data set size with a Gaussian law. This law defines a mean $\mu^{rand}_{i_{max}}$ computed with the equation 16 on a set $\mathcal{D}$ of pseudo-random data sets of the same size i_{max}.

$$\mu^{rand}_{i_{max}} = \frac{\sum_{\mathcal{S}\in\mathcal{D}} H(\mathcal{S})}{\#\mathcal{D}} \qquad (16)$$

The standard deviation is defined by equation 17.

$$\sigma^{rand}_{i_{max}} = \sqrt{\frac{\sum_{\mathcal{S}\in\mathcal{D}} \left(H(\mathcal{S}) - \mu^{rand}_{i_{max}}\right)^2}{\#\mathcal{D}}} \qquad (17)$$

A classical result of Gaussian analysis is that the proportion of values (if following a Gaussian law) in the range $[\mu^{rand}_{i_{max}} - z * \sigma^{rand}_{i_{max}}, \mu^{rand}_{i_{max}} + z * \sigma^{rand}_{i_{max}}]$ is:

$$erf\left(\frac{z}{\sqrt{2}}\right) \qquad (18)$$

where erf is the so-called error function. We can hence define a confidence that an entropy value is a result of a random process. Indeed it is sufficient to compare the value with the Gaussian distribution by determining the likelihood that it belongs to that distribution. For example, a value at a 6-sigma distance from $\mu^{rand}_{i_{max}}$ has a very low probability to be the result of a random process.

The confidence we have in the fact that a particular entropy is not random can be directly expressed by this z value.
Hence the confidence cf in any entropy H of data size i_{max} can be computed from H, $\mu^{rand}_{i_{max}}$ and $\sigma^{rand}_{i_{max}}$:

$$cf_{i_{max}}(H) = \frac{\mu^{rand}_{i_{max}} - H}{\sigma^{rand}_{i_{max}}} \qquad (19)$$

In our case, for a given data set size, we have 255 random entropies corresponding to the 255 wrong key guesses. For each data size, random entropies are in average comprised in the range $[\mu^{rand}_{i_{max}} - z^{rand} * \sigma^{rand}_{i_{max}}, \mu^{rand}_{i_{max}} + z^{rand} * \sigma^{rand}_{i_{max}}]$. where $z^{rand} = \sqrt{2} erf^{-1}\left(\frac{255}{256}\right) \approx 2.9$. We need a criterion to determine if a key hypothesis is correct or not, larger than a 3-sigma confidence. We chose a 6-sigma one, but one could chose another value, depending on the desired level of confidence.

APPENDIX C. ERROR DISTRIBUTIONS WITH BERTONI'S COUNTERMEASURE

As an example of the effects of a countermeasure, we show on Fig. 10 and 11 the resulting virtual distributions respectively with and without result discrimination. In this particular example, the global detection rate is $D \approx 0.52$.

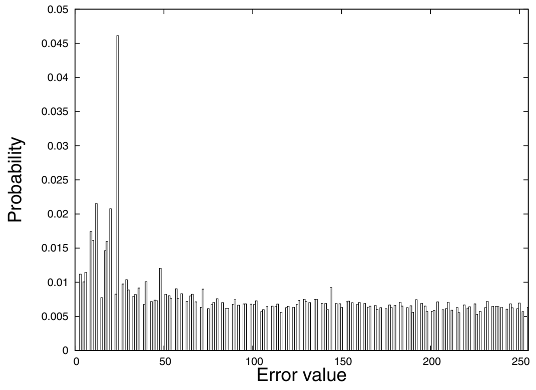

Figure 10. Distribution of the virtual injection with result discrimination.

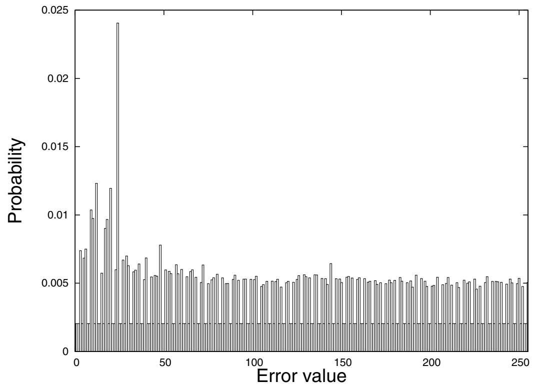

Figure 11. Distribution of the virtual injection without result discrimination.

References

[1] P. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss and other systems," in CRYPTO. New-York: Springer-Verlag, 1996, pp. 104–113.

[2] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in CRYPTO, 1999, pp. 388–397.

[3] E. O. Stefan Mangard and T. Popp, Power Analysis Attacks - Revealing the Secrets of Smart Cards. Springer Verlag, 2007.

[4] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in CHES, 2004, pp. 16–29.

[5] B. Gierlichs, L. Batina, and P. Tuyls, "Mutual information analysis – a universal differential side-channel attack," Cryptology ePrint Archive, Report 2007/198, 2007. [Online]. Available: http://eprint.iacr.org/

[6] H. Maghrebi, J. Danger, F. Flament, S. Guilley, and L. Sauvage, "Evaluation of countermeasure implementations based on boolean masking to thwart side-channel attacks," in Signals, Circuits and Systems (SCS). IEEE, 2009, pp. 1–6.

[7] H. Maghrebi, S. Guilley, J. Danger, and F. Flament, "Entropy-based power attack," in Hardware-Oriented Security and Trust (HOST). IEEE, 2010, pp. 1–6.

[8] H. Choukri and M. Tunstall, "Round reduction using faults," in FDTC, 2005, pp. 13–24.

[9] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in EUROCRYPT, ser. Lecture Notes in Computer Science, W. Fumy, Ed., vol. 1233. Springer, 1997, pp. 37–51.

[10] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," in CRYPTO, ser. Lecture Notes in Computer Science, B. Kaliski Jr., Ed., vol. 1294. Springer, 1997, pp. 513–525.

[11] K. Sakiyama, Y. Li, M. Iwamoto, and K. Ohta, "Information-theoretic approach to optimal differential fault analysis," Information Forensics and Security, vol. 7, pp. 109–120, 2012.

[12] G. Piret and J.-J. Quisquater, "A differential fault attack technique against SPN structures, with application to the AES and Khazad," in CHES, ser. Lecture Notes in Computer Science, C. Walter, Ed., vol. 2779. Springer, 2003, pp. 77–88.

[13] A. Moradi, M. T. M. Shalmani, and M. Salmasizadeh, "A generalized method of differential fault attack against aes cryptosystem." in CHES, 2006, pp. 91–100.

[14] D. Mukhopadhyay, "An improved fault based attack of the advanced encryption standard," in AFRICACRYPT, ser. Lecture Notes in Computer Science, B. Preneel, Ed. Springer Berlin / Heidelberg, 2009, vol. 5580, pp. 421–434, 10.1007/978-3-642-02384-2_26. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02384-2_26

[15] M. Tunstall and D. Mukhopadhyay, "Differential fault analysis of the advanced encryption standard using a single fault," Cryptology ePrint Archive, Report 2009/575, 2009. [Online]. Available: http://eprint.iacr.org/2009/575.pdf

[16] C. Giraud, "Dfa on aes," in Advanced Encryption Standard - AES, ser. Lecture Notes in Computer Science, H. Dobbertin, V. Rijmen, and A. Sowa, Eds., vol. 3373. Springer Berlin / Heidelberg, 2005, pp. 27–41.

[17] T. Roche, V. Lomné, and K. Khalfallah, "Combined fault and side-channel attack on protected implementations of aes," Smart Card Research and Advanced Applications, pp. 65–83, 2011.

[18] M. Rivain, "Differential fault analysis on des middle rounds," CHES, pp. 457–469, 2009.

[19] S.-M. Yen and M. Joye, "Checking before output may not be enough against fault-based cryptanalysis," IEEE Transactions on Computers, vol. 49, no. 9, pp. 967–970, 2000.

[20] B. Robisson and P. Manet, "Differential behavioral analysis," in CHES, 2007, pp. 413–426.

[21] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis," in CHES, ser. Lecture Notes in Computer Science, S. Mangard and F.-X. Standaert, Eds. Springer Berlin / Heidelberg, 2010, vol. 6225, pp. 320–334, 10.1007/978-3-642-15031-9_22. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15031-9_22

[22] NIST, “Specification for the Advanced Encryption Standard,” FIPS PUB, vol. 197, November 2001.

[23] M. Agoyan, J. Dutertre, D. Naccache, B. Robisson, and A. Tria, “When clocks fail: On critical paths and clock faults,” Smart Card Research and Advanced Application, pp. 182–193, 2010.

[24] S. Endo, T. Sugawara, N. Homma, T. Aoki, and A. Satoh, “An on-chip glitchy-clock generator for testing fault injection attacks,” J. Cryptographic Engineering, vol. 1, no. 4, pp. 265–270, 2011.

[25] S. Guilley, L. Sauvage, J.-L. Danger, N. Selmane, and R. Pacalet, “Silicon-level Solutions to Counteract Passive and Active Attacks,” in FDTC, IEEE-CS, Ed. Washington, DC, États-Unis: IEEE-CS, Aug. 2008, pp. 3–17.

[26] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, “A parity code based fault detection for an implementation of the advanced encryption standard,” in Defect and Fault-Tolerance in VLSI Systems (DFT). Washington, DC, USA: IEEE Computer Society, 2002, pp. 51–59.