Differential Fault Analysis on DES Middle Rounds

Matthieu Rivain

Oberthur Technologies & University of Luxembourg
m.rivain@oberthur.com

Abstract. Differential Fault Analysis (DFA) is a powerful cryptanalytic technique that disturbs cryptographic computations and exploits erroneous results to infer secret keys. Over the last decade, many works have described and improved DFA techniques against block ciphers thus showing an inherent need to protect their implementations. A simple and widely used solution is to perform the computation twice and to check that the same result is obtained. Since DFA against block ciphers usually targets the last few rounds, one does not need to protect the whole ciphering thus saving computation time. However the number of rounds to protect must be chosen very carefully in order to prevent security flaws. To determine this number, one must study DFA targeting middle rounds of the cipher. In this paper, we address this issue for the Data Encryption Standard (DES) algorithm. We describe an attack that breaks DES by introducing some faults at the end of round 9, 10, 11 or 12, more or less efficiently depending on the fault model and the round number.

1 Introduction

Fault analysis is a class of implementation attacks that consists in disturbing cryptographic computations to recover secret keys. Among these attacks, one merely identifies two families which differ in the information exploited to recover the key. *Differential Fault Analysis* (DFA) [3] exploits the difference between correct and faulty results while other attacks focus on the behavior of the corrupted computation, namely on whether the induced fault effectively provokes an erroneous result or not. Among them, one lists *safe-error attacks* [26] on exponentiation algorithms as well as *Ineffective Fault Analysis* (IFA) [9, 23] and *Collision Fault Analysis* (CFA) [16] against block ciphers implementations.

IFA and CFA consider an adversary that is able to set an intermediate variable of its choice to a known value (usually to 0). If the result is erroneous or if a fault is detected, the attacker knows that the intermediate variable was different from the induced value. Obtaining this information for several encryptions enables key-recovery. A simple way to thwart this kind of attack is to use data masking [15, 2] which is often applied to protect embedded implementation against *power analysis* [18]. Indeed, masking ensures that no single intermediate variable provides information on the secret key. However, masking does not ensure the result integrity and is hence ineffective against DFA [5].

DFA on block ciphers was first introduced by Biham and Shamir against DES [3]. Since then, several DFA were proposed on AES [11,4,13,22,7,25,17] as well as on other block ciphers such as IDEA [10] and CLEFIA [8,24]. These different works demonstrate the vulnerability of block ciphers towards DFA and the subsequent need of including countermeasures to embedded implementations. A straightforward way to protect any algorithm against DFA is to compute it twice and check whether the obtained results are equal or not. Another similar solution is to verify the integrity of an encryption by a decryption and *vice versa*. It is also possible to include redundancy and coherence checking at the operation level; the complexity-security ratio of such schemes is usually of the same order than the one of computation doubling [19]. An advantage of computation doubling is the scalability on the number of rounds to protect. In fact, most of DFA techniques target the last few rounds of the block cipher. To thwart these attacks, one only need to double the computation of these last few rounds thus saving computation time. However, a question remains: how many rounds should be protected to obtain a good security level towards DFA? To answer this question, we need to investigate DFA on middle rounds of the cipher.

This issue has been addressed in [21] by Phan and Yen for the AES block cipher. They apply block cipher cryptanalysis techniques to improve DFA on AES and exhibit some attacks against rounds 7, 6 and 5. Concerning DES, the original work by Biham and Shamir [3] described an attack that exploits a fault corrupting either round 16, 15 or 14 (and equivalently the end of round 15, 14 or 13). In his PhD thesis [1], Akkar investigates the application of differential cryptanalysis techniques to attack earlier rounds of DES. In a first place, the considered attacker is assumed to be able to induce a differential of its choice in the DES internal value at the end of some round. The last round key is recovered by guessing every 6-bit parts independently and by selecting, for each subkey, the candidate that produces the expected differential at the S-box output the more frequently. The obtained attacks are quite efficient but, as mentioned by the author, the fault model is not realistic. Akkar then applies this attack under two more realistic fault models: a single bit switch at a fixed position (in the left part of the DES internal state) and a single bit switch at a random position (in the right part of the DES internal state). For the fixed position bit error model, the attack needs a few hundred fault injections at the end of round 11 and it fails on round 9 (the attack on round 10 is not considered). For the random position bit error model, the attack needs a few dozen fault injections at the end of round 12 and it fails on round 11.

In this paper, we generalize and improve the attack described by Akkar in [1]. We consider various realistic fault models for an error induced in the left part of the DES internal state, including the bit error model and the byte error model with chosen error position or random error position. As we will argue, disturbing the left part leads to better attacks than disturbing the right part. Moreover, we use more accurate distinguishers than the one proposed in [1]. In the usual (chosen position) byte error model, our attack recovers the whole last round key with a 99% success rate using 9 faults on round 12, 210 faults on round 11 and

13400 faults on round 10. In the (chosen position) bit error model, these numbers are reduced to 7, 11 and 290, respectively.

2 Data Encryption Standard

The Data Encryption Standard (DES) [12] is a block cipher that was selected by the US National Bureau of Standards in 1976 as an official standard for data encryption. DES uses a 56-bit key (usually represented on 64 bits including 8 parity check bits) and it operates on 64-bit blocks. It has an iterative structure applying 16 times the same round transformation F which is preceded by a bit-permutation IP and followed by the inverse bit-permutation IP^{-1}. Every round transformation is parameterized by a 48-bit round key $\mathbf{k}_r$ that is derived from the secret key through a key schedule process. To summarize, a ciphertext C is computed from a plaintext P as follows:

$$C = \mathsf{IP}^{-1} \circ \left(\bigcirc_{r=1}^{16} \mathsf{F}_{\mathbf{k}_r}\right) \circ \mathsf{IP}(P) \ .$$

The round transformation follows a *Feistel scheme*, namely, the block is split into two 32-bit parts L (the left part) and R (the right part), and F is defined as:

$$\mathsf{F}_{\mathbf{k}_r}(L, R) = (R, L \oplus f_{\mathbf{k}_r}(R)) \ ,$$

where f is a function parameterized with a 48-bit key and operating on a 32-bit block. This structure is illustrated on Fig. 1. In the sequel, the output block of the r-th round shall be denoted as (L_r, R_r). Defining $(L_0, R_0) = \mathsf{IP}(P)$, we have $(L_r, R_r) = \mathsf{F}_{\mathbf{k}_r}(L_{r-1}, R_{r-1})$ for every $r \leq 16$ and $C = \mathsf{IP}^{-1}(L_{16}, R_{16})$.

The function f of the DES first applies an *expansion layer* E that expands the 32 input bits into 48 output bits by duplicating 16 of them. The round key is then introduced by bitwise addition afterward the block is split into eight 6-bit blocks, each entering into a different *substitution box* (S-box) S_i producing a 4-bit output. Finally, the 32 bits from the eight S-box outputs are permuted through a bit-permutation P which yields the 32-bit output block.

In the sequel, E_i and P_i^{-1} denote the i-th 6-bit coordinate of the expansion layer E and the i-th 4-bit coordinate of the bit-permutation P^{-1}, respectively.

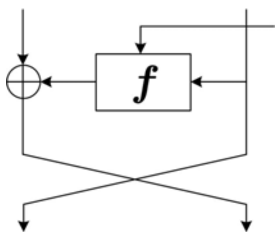


Fig. 1. Round transformation in the Feistel scheme

Similarly, $\mathbf{k}_{r,i}$ shall denote the i-th 6-bit part of a round key $\mathbf{k}_r$. We hence have the equality:

$$\mathsf{P}_i^{-1}(f_{\mathbf{k}_r}(\cdot)) = \mathsf{S}_i(\mathsf{E}_i(\cdot) \oplus \mathbf{k}_{r,i}) \ . \tag{1}$$

3 Fault Models

Our attack consists in corrupting some bits of the left part of the DES internal state at the end of the r-th round with $r \in \{9, 10, 11, 12\}$. We shall consider different fault models depending on the statistical distribution of the induced error. We first consider the *bit error model* : one and one single bit of the left part is switched. We also consider the *byte error model* : one byte of the left part is switched to a random and uniformly distributed value. Furthermore, the fault position may be either *chosen* by the attacker or *random* among the 32 bit-positions or the 4 byte-positions of the left part.

In the sequel, $\widetilde{L}_i$ and $\widetilde{R}_i$ will respectively denote the corrupted value of the left part L_i and the right part R_i at the end of round i and $\widetilde{C} = \mathsf{IP}^{-1}(\widetilde{L}_{16}, \widetilde{R}_{16})$ will denote the faulty ciphertext. We shall further denote by ε the induced error that is defined as $\varepsilon = L_r \oplus \widetilde{L}_r$.

4 Attack Description

4.1 General Principle

Let us denote by Δ the bitwise difference between the correct value and the corrupted value of the left part at the end of the fifteenth round: $\Delta = L_{15} \oplus \widetilde{L}_{15}$. Due to the Feistel scheme, we have the following relation:

$$R_{16} \oplus \widetilde{R}_{16} = f_{\mathbf{k}_{16}}(L_{16}) \oplus f_{\mathbf{k}_{16}}(\widetilde{L}_{16}) \oplus \Delta \ . \tag{2}$$

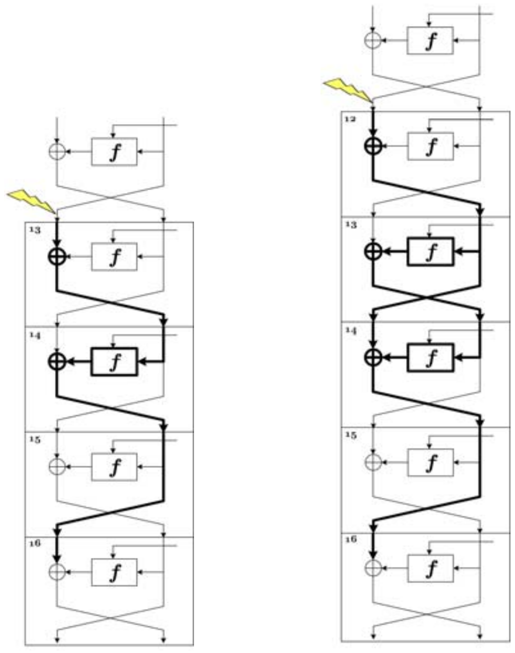
Based on (2), an adversary that knows Δ can mount a key recovery attack. The principle is to make a guess on the value of the round key $\mathbf{k}_{16}$. Then, given a pair of ciphertexts $(C, \widetilde{C})$, the attacker checks whether (2) is consistent for this guess. If not, the guess is discarded. In this way, $\mathbf{k}_{16}$ is uniquely determined using few pairs of ciphertexts. Due to the structure of f (see (1)), the attacker does not need to guess the whole round key $\mathbf{k}_{16}$ but he can guess and check each subkey $\mathbf{k}_{16,i}$ independently. When an error is induced in the final rounds, the differential Δ (or at least a part of it) can be predicted according to the pair $(C, \widetilde{C})$ which enables the attack [3]. This is no more the case for an error induced in a middle round; in that case the attack must be extended.

As noted in [1], if an error ε is induced in the left part at the end of the thirteenth round then Δ equals ε. Therefore, an attacker that is able to induce a chosen (or at least known) error in L_{13} can apply the previous attack. For a fault induced in the left part during an earlier round, the equality $\Delta = \varepsilon$ does not hold anymore. However the statistical distribution of Δ may be significantly biased (depending on the fault model and the round number). Indeed, as illustrated in Fig. 2, a fault injection in the left part skips one round before propagating through the function f. Besides, the error propagation path from L_r to L_{15} sticks through the function f only once for $r = 12$, twice for $r = 11$, etc. This is quite low considering the slow diffusion of the function f. As a result, a fault induced in L_r may produce a differential Δ with a distribution that is significantly biased. As described hereafter, this bias enables a key recovery attack based on a statistical distinguisher.

Remark 1. From Fig. 2, it can be noticed that the injection of an error ε in L_r is equivalent to the injection of ε in R_{r+1}. This demonstrates the relevance of attacking the left part rather than the right one. Besides, this explains why the attack on the right part described in [1] is inefficient compared to the one on the left part on the same round.

Let us define, for every $i \in \{1, \cdots, 8\}$, the function g_i as the prediction of the i-th 4-bit coordinate of $\mathsf{P}^{-1}(\Delta)$ according to a pair $(C, \widetilde{C})$ and to a guess k on the value of $\mathbf{k}_{16,i}$:

$$g_i(C,\widetilde{C},k) = \mathsf{S}_i\big(\mathsf{E}_i(L_{16}) \oplus k\big) \oplus \mathsf{S}_i\big(\mathsf{E}_i(\widetilde{L}_{16}) \oplus k\big) \oplus \mathsf{P}_i^{-1}\big(R_{16} \oplus \widetilde{R}_{16}\big)\ .$$



(a) From L_{12} to L_{15}. (b) From L_{11} to L_{15}.

Fig. 2. Error propagation paths

From (1) and (2), it can be checked that, for the correct key guess, $g_i(C, \widetilde{C}, k)$ equals $\mathsf{P}_i^{-1}(\Delta)$. On the other hand, for a wrong key guess, $g_i(C, \widetilde{C}, k)$ can be assumed to have a uniform distribution. This is a classical assumption in block cipher cryptanalysis known as the *wrong-key assumption.*

Let us define, for every $i \in \{1, \cdots, 8\}$ and for every $\delta \in \{0, \cdots, 15\}$, the probability $p_i(\delta)$ as:

$$p_i(\delta) = \Pr\left[\mathsf{P}_i^{-1}(\Delta) = \delta\right] .$$

To summarize, according to the wrong-key assumption, we have:

$$\Pr\left[g_i(C, \widetilde{C}, k) = \delta\right] = \begin{cases} p_i(\delta) & \text{if } k = \mathbf{k}_{16,i} \\ \frac{1}{16} & \text{otherwise} \end{cases} \tag{3}$$

Provided that the distribution $p_i(\cdot)$ is significantly biased, (3) clearly exhibits a wrong-key distinguisher for $\mathbf{k}_{16,i}$.

4.2 Wrong-Key Distinguishers

We define hereafter two possible distinguishers $d(k)$ for a key candidate k which are expected to be maximal for the correct key candidate $k = \mathbf{k}_{16,i}$. These distinguishers take as input a set of N pairs $(C_n, \widetilde{C}_n)$, $1 \leq n \leq N$. The choice of the distinguisher to use depends on the attacker's knowledge of the fault model.

Likelihood distinguisher. The attacker is assumed to have an exact knowledge of the fault model, namely he knows the distribution of ε. In that case, he can compute (or at least estimate) the distribution $p_i(\cdot)$ in order to use a maximum likelihood approach. The likelihood of a key candidate k is defined as the product of the probabilities $p_i\big(g_i(C_n, \widetilde{C}_n, k)\big)$ for $n = 1, \cdots, N$. For practical reasons, we make the classical choice to use the logarithm of the likelihood, namely $d(k)$ is defined as:

$$d(k) = \sum_{n=1}^{N} \log\left(p_i\big(g_i(C_n, \widetilde{C}_n, k)\big)\right) .$$

Squared Euclidean Imbalance (SEI) distinguisher. The attacker does not have a precise knowledge of the fault model and is hence not able to estimate the distribution $p_i(\cdot)$. In that case, an alternative strategy is to look for the strongest bias in the distribution of $g_i(C_n, \widetilde{C}_n, k)$. This is done by computing the squared Euclidean distance to the uniform distribution (known as *squared Euclidean imbalance*), namely $d(k)$ is defined as:

$$d(k) = \sum_{\delta=0}^{15} \left(\frac{\#\{n; g_i(C_n, \widetilde{C}_n, k) = \delta\}}{N} - \frac{1}{16}\right)^2 .$$

4.3 Chosen Error Position Strategies

In a chosen error position fault model scenario, we further have to define a strategy to choose the positions where to induce the errors.

Table 1. Destination S-boxes for the input bits of f

bits	1,32	2,3	4,5	6,7	8,9	10,11	12,13	14,15
S-boxes	1,8	1	1,2	2	2,3	3	3,4	4
bits	16,17	18,19	20,21	22,23	24,25	26,27	28,29	30,31
S-boxes	4,5	5	5,6	6	6,7	7	7,8	8

Bit error model. In the bit error model, ε has a single bit to 1 which implies that the function f in round $r+2$ has one or two *active S-boxes*. That is, the correct output and the corrupted output of f only differ for one or two S-boxes. Indeed, as shown in Table 1, the expansion layer sends every input bit of f in one or two S-boxes. In order to maximize the bias in the distribution of Δ, the bit-positions should be chosen among the ones entering in a single S-box hence slowing the error propagation. Our strategy is simply to first choose a bit-position entering in S-box 1 only, then in S-box 2 only, and so on until S-box 8 and start over with S-box 1, etc.

Remark 2. Relation (3) implicitly assumes that the i-th S-box in the sixteenth round is active, otherwise $g_i(C, \widetilde{C}, k)$ equals 0 for every k. For a chosen position bit error attack on round 12, each selected bit-position implies that two S-boxes are *inactive* in round 16. However, the pair of inactive S-boxes differs for each bit-position which ensures the soundness of the attack.

Byte error model. Concerning the byte error model, every input byte of f is spread over four S-boxes. This can be checked from Table 2 that gives the destination S-boxes of every input byte of f. As a result, a byte error in L_r always implies four active S-boxes in the output differential of f in round $r+2$. For the attacks in the chosen position byte error model, the four byte-positions are hence equivalently chosen since they all induce the corruption of exactly four S-boxes in round $r+2$.

Remark 3. In a chosen error position attack, several fault models are involved hence, for a given i, different distributions $p_i(\cdot)$ are induced. Consequently, the SEI distinguisher shall not be directly applied but the SEI of $p_i(\cdot)$ shall be estimated for every error position independently. The SEI distinguisher is then defined as the sum of the SEIs for the different error positions.

Remark 4. In our attack simulations, we tried more specific strategies taking into account the bias in the $(p_i(\cdot))_i$ distributions resulting from the different bit-error positions. These strategies did not yield substantial improvements.

Table 2. Destination S-boxes for the input bytes of f

bytes	1	2	3	4
S-boxes	8,1,2,3	2,3,4,5	4,5,6,7	6,7,8,1

5 Attack Simulations

This section presents some experimental results. We performed attack simulations for each of the fault models introduced in Sect. 3 with a fault induced at the end of round 12, 11, 10 or 9. For every round number and every fault model, we applied the likelihood distinguisher and the SEI distinguisher (see Sect. 4.2). For the likelihood distinguisher, we empirically computed the distributions $(p_i(\cdot))_i$ based on several[1] ciphertexts pairs, each obtained from the correct and faulty encryptions of a random plaintext[2].

In what follows, we consider an attack successful when the whole last round key is determined with a 99% success rate. This strong requirement is motivated by the fact that, for a triple DES, too many key bits remain to perform an exhaustive search once the last round key has been recovered. Therefore, one shall fully determine the sixteenth round key before reiterating the attack on the fifteenth and so on. Every subsequent attack on a previous round key can be performed by using the same set of ciphertexts pairs and is expected to be substantially more efficient since the error propagates on fewer rounds. This way, if the last round key is recovered with a 99% success rate then the cipher can be considered fully broken.

Fig. 3 shows the success rate (over 1000 simulations) of the different attacks (chosen/random position bit/byte error, likelihood/SEI distinguishers) on round 12, 11 and 10. Fig. 4 shows the success rate (over 10 to 100 simulations) for the attacks on round 9 in the bit error model. Attacks on round 9 in the byte error model all required more than 10^8 faults. The numbers of faults required for a 99% success rate are summarized in Table 3.

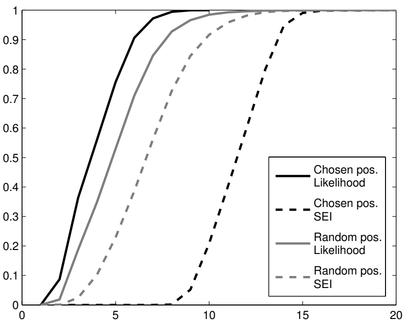
Attack efficiency vs. round number. The attacks on rounds 11 and 12 are very efficient: less than 25 faults are sufficient on round 12 while, on round 11, less than 100 faults are sufficient in a bit error model and less than 1000 faults are sufficient in a byte error model. On round 10, the attacks are still fairly efficient: the best attack (chosen position bit error model, likelihood distinguisher) requires 290 faults whereas the least efficient attack (chosen position byte error model, SEI distinguisher) requires 26400 faults. It is on round 9 that the attacks become quite costly since the most efficient attack in the bit error model (chosen position, likelihood distinguisher) requires around $3.4 \cdot 10^5$ faults and all the attack in the byte error model require more than 10^8 faults[3].

Attack efficiency vs. fault model. As expected, we observe that, for a given setting (random/chosen position, likelihood/SEI distinguisher), a bit error model always leads to more efficient attacks than a byte error model. Similarly, a chosen position usually leads to more efficient attacks than a random position. Some exceptions are observed for the SEI distinguisher for which a random position sometimes leads to more efficient attacks than a chosen position. The reason of

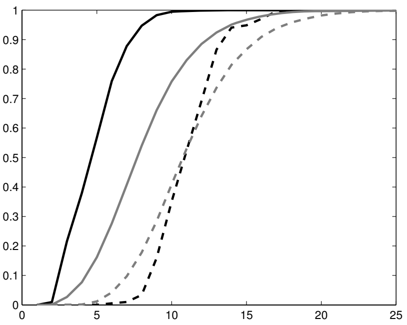
[1] 10^7 for bit errors models and 10^8 for byte errors models.

[2] Note that the value of the key does not change the $(p_i(\cdot))_i$ distributions.

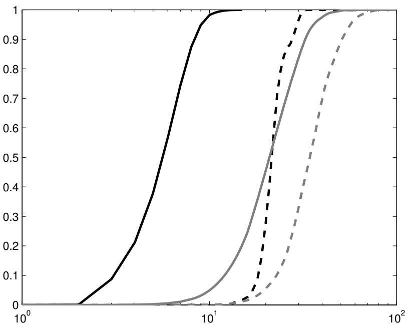
[3] The most efficient one (chosen position byte error model, likelihood distinguisher) yielded a 0% success rate (over 10 attack simulations) for 10^8 faults.



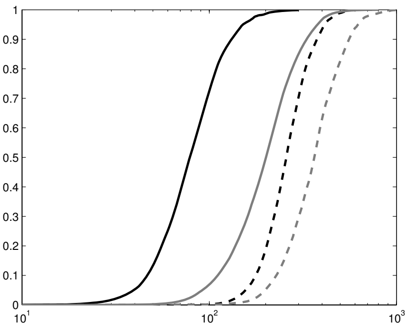
Attack on round 12 – Bit error model.



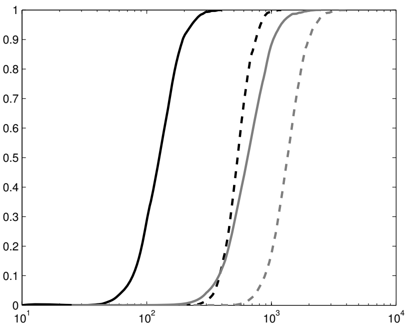
Attack on round 12 – Byte error model.



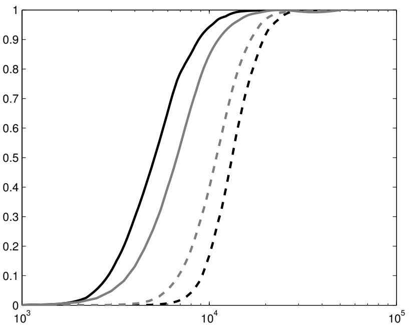
Attack on round 11 – Bit error model.



Attack on round 11 – Byte error model.



Attack on round 10 – Bit error model.



Attack on round 10 – Byte error model.

Fig. 3. Attacks on rounds 10, 11 and 12: success rate w.r.t. number of faults

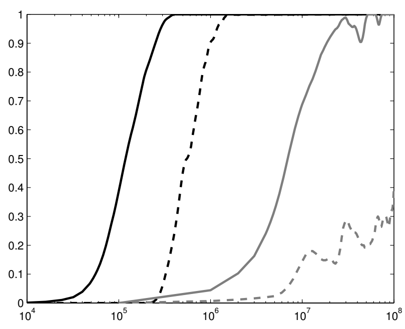


Fig. 4. Attacks on rounds 9, bit error model: success rate w.r.t. number of faults

Table 3. Number of faults to recover the 16-th round key with a 99% success rate

		bit error		byte error	
round	distinguisher	chosen pos.	random pos.	chosen pos.	random pos.
12	Likelihood	7	11	9	17
	SEI	14	12	17	21
11	Likelihood	11	44	210	460
	SEI	30	71	500	820
10	Likelihood	290	1500	13400	18500
	SEI	940	2700	26400	23400
9	Likelihood	$3.4 \cdot 10^5$	$2.2 \cdot 10^7$	$> 10^8$	$> 10^8$
	SEI	$1.4 \cdot 10^6$	$> 10^8$	$> 10^8$	$> 10^8$

this phenomenon may be that in a chosen position bit (resp. byte) error model, 8 (resp. 4) different SEIs are estimated based on 8 (resp. 4) times less faults than in the random position model where a single SEI is estimated (see Remark 3). As a result, these estimations are less precise which may render the attack less efficient than in the random position model. In these cases, the attacker can compute a single SEI based on all the faults, which amounts to perform the attack in the random position model.

To summarize, we naturally have that a bit error is better than a byte error and a chosen position is better than a random position. What was not *a priori* straightforward is the superiority of the random position bit error model compared to the chosen position byte error model. Except on round 12 where both cases are almost equivalent, our results show that the attacks in the random position bit error model are significantly more efficient than the ones in the chosen position byte error model.

Another interesting observation is that, in the bit error model, the ability to choose the error position is more advantageous than in the byte error model.

This phenomenon results from the strategy for the choice of the bit-positions (see Sect. 4.3) which selects 8 positions over 32 leading to more important bias in the distributions $p_i(\cdot)$ than the average case, whereas, in the chosen position byte error model, the 4 byte-positions are equivalently used.

Likelihood vs. SEI. As expected, the likelihood distinguisher always leads to more efficient attacks than the SEI distinguisher. It is interesting to note that this difference of efficiency is always greater in a chosen position model than in a random position model. Once again, this phenomenon results from the fact that, for a chosen position model, several different SEIs are estimated based on 4 or 8 times less faults compared to the random position model where a single SEI is estimated.

6 How Many Rounds To Protect ?

The question of the number of rounds to protect does not have a unique answer. Indeed, the answer to this question depends on the ability of an attacker to induce faults and on the number of correct and faulty ciphertexts pairs that he can collect. Besides, more efficient attacks that those described in this paper may exist.

What provide our paper are some lower bounds on the number of rounds to protect. We have shown that in a realistic fault model, efficient DFA attacks can be performed by inducing some faults until round 10. It seems therefore reasonable to protect at least the last seven rounds of the cipher. However, this may not suffice while considering a strong adversary model. We have shown that in a chosen position bit error model, $3.4 \cdot 10^5$ faults induced at the end of round 9 are sufficient to recover the last round key with a 99% confidence. Consequently, in order to thwart an adversary able to induce a single bit fault at a chosen position and to gather about 10^5 ciphertexts pairs, one shall at least protect the last eight rounds.

Attacks on initial rounds. As noted in [14], if an attacker has access to a decryption oracle then any DFA attack can be transposed on the initial rounds of the cipher. In fact, the attacker may obtain a faulty ciphertext $\widetilde{C}$ from a plaintext P by inducing a fault at the end of the first round. The plaintext P can then be viewed as the faulty result of a decryption of $\widetilde{C}$ for which a fault has been induced at the beginning of the last round. The attacker then asks for the decryption of $\widetilde{C}$ which provides him with a plaintext $\widetilde{P}$. The pair $(\widetilde{P}, P)$ thus constitutes a pair of correct and faulty results of the decryption algorithm with respect to an error induced at the beginning of the last round. According to this principle, any fault attack on an initial round of an encryption can be transposed to a fault attack on a final round of a decryption, provided that the attacker has access to a decryption oracle. In that case, the same number of rounds should be protected at the beginning and at the end of the cipher in order to obtain an homogenous security level. For a simple DES, based on our study, we recommend to protect the whole cipher. For a triple DES, one can only protect some rounds at the beginning of the first DES computation and some rounds at the end of the last DES computation; the number of protected rounds being at least seven according to our study.

7 Conclusion

In this paper, we have investigated differential fault analysis on DES middle rounds. We have described a generic attack and we have demonstrated its efficiency under various realistic fault models. We have shown that DES can be broken by inducing some faults at the end of rounds 12, 11, 10 and 9, more or less efficiently depending on the round number and the fault model. Although we focused on DES, our attack could be applied on any Feistel scheme.

References

1. Akkar, M.-L.: Attaques et méthodes de protections de systèmes cryptographiques embarqués. PhD thesis, Université de Versailles Saint-Quentin (October 1, 2004)
2. Akkar, M.-L., Giraud, C.: An Implementation of DES and AES, Secure against Some Attacks. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 309–318. Springer, Heidelberg (2001)
3. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystem. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
4. Blömer, J., Seifert, J.-P.: Fault Based Cryptanalysis of the Advanced Encryption Standard. In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003)
5. Boscher, A., Handschuh, H.: Masking Does Not Protect Against Differential Fault Attacks. In: Breveglieri et al. [6], pp. 35–40
6. Breveglieri, L., Gueron, S., Koren, I., Naccache, D., Seifert, J.-P. (eds.): Fault Diagnosis and Tolerance in Cryptography – FDTC 2008. IEEE Computer Society Press, Los Alamitos (2008)
7. Chen, C.-N., Yen, S.-M.: Differential Fault Analysis on AES Key Schedule and Some Countermeasures. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 118–129. Springer, Heidelberg (2003)
8. Chen, H., Wu, W., Feng, D.: Differential Fault Analysis on CLEFIA. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 284–295. Springer, Heidelberg (2007)
9. Clavier, C.: Secret External Encodings Do Not Prevent Transient Fault Analysis. In: Paillier, Verbauwhede [20], pp. 181–194
10. Clavier, C., Gierlichs, B., Verbauwhede, I.: Fault Analysis Study of IDEA. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 274–287. Springer, Heidelberg (2008)
11. Dusart, P., Letourneux, G., Vivolo, O.: Differential Fault Analysis on A.E.S. In: Zhou, J., Yung, M., Han, Y. (eds.) ACNS 2003. LNCS, vol. 2846, pp. 293–306. Springer, Heidelberg (2003)
12. FIPS PUB 46-3. Data Encryption Standard (DES). National Institute of Standards and Technology, October 25 (1999)
13. Giraud, C.: DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 27–41. Springer, Heidelberg (2005)

14. Giraud, C.: Attaques de cryptosystémes embarqués et contre-mesures associées. Thése de doctorat, Université de Versailles, Oberthur Card Systems (October 2007)
15. Goubin, L., Patarin, J.: DES and Differential Power Analysis – The Duplication Method. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 158–172. Springer, Heidelberg (1999)
16. Hemme, L.: A Differential Fault Attack against Early Rounds of (Triple-)DES. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 254–267. Springer, Heidelberg (2004)
17. Kim, C.H., Quisquater, J.-J.: New Differential Fault Analysis on AES Key Schedule: Two Faults Are Enough. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 48–60. Springer, Heidelberg (2008)
18. Kocher, P., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
19. Malkin, T., Standaert, F.-X., Yung, M.: A Comparative Cost/Security Analysis of Fault Attack Countermeasures. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 159–172. Springer, Heidelberg (2006)
20. Paillier, P., Verbauwhede, I. (eds.): CHES 2007. LNCS, vol. 4727. Springer, Heidelberg (2007)
21. Phan, R., Yen, S.-M.: Amplifying Side-Channel Attacks with Techniques from Block Cipher Cryptanalysis. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 135–150. Springer, Heidelberg (2006)
22. Piret, G., Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)
23. Robisson, B., Manet, P.: Differential Behavioral Analysis. In: Paillier, Verbauwhede [20], pp. 413–426
24. Takahashi, J., Fukunaga, T.: Improved Differential Fault Analysis on CLEFIA. In: Breveglieri et al. [6], pp. 25–34
25. Takahashi, J., Fukunaga, T., Yamakoshi, K.: DFA Mechanism on the AES Key Schedule. In: Breveglieri, L., Gueron, S., Koren, I., Naccache, D., Seifert, J.-P. (eds.) Fault Diagnosis and Tolerance in Cryptography – FDTC 2007, pp. 62–74. IEEE Computer Society Press, Los Alamitos (2007)
26. Yen, S.-M., Joye, M.: Checking Before Output Not Be Enough Against Fault-Based Cryptanalysis. IEEE Transactions on Computers 49(9), 967–970 (2000)