Fault Injection to Reverse Engineer DES-like Cryptosystems

Hélène Le Bouder, Sylvain Guilley, Bruno Robisson, Assia Tria

To cite this version:

Hélène Le Bouder, Sylvain Guilley, Bruno Robisson, Assia Tria. Fault Injection to Reverse Engineer DES-like Cryptosystems. Foundations and Practice of Security - 6th International Symposium, FPS 2013, , 2013, La Rochelle, France. hal-01818570

HAL Id: hal-01818570

https://hal-imt-atlantique.archives-ouvertes.fr/hal-01818570

Submitted on 19 Jun 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Fault Injection to Reverse Engineer DES-like Cryptosystems

Hélène Le Bouder[1,2], Sylvain Guilley[3], Bruno Robisson[1,4], and Assia Tria[1,4]

[1] Département Systèmes et Architectures Sécurisés (SAS), Gardanne, France
[2] Institut Mines-Télécom, École des Mines de Saint-Étienne (EMSE)
[3] Institut Mines-Télécom, TELECOM-ParisTech
[4] Commissariat à l'énergie atomique et aux énergies alternatives (CEA)

Abstract. This paper presents a fault injection attack in order to reverse engineer unknown s-boxes of a DES-like cryptosystem. It is a significant improvement of the FIRE attack presented by San Pedro *et al.* which uses differentials between s-boxes outputs. Since injecting faults on a cryptographic circuit may irreversibly damage the device, our aim has been to minimise the number of faults needed. We show that by considering faults in the penultimate round instead of last round, twice less faults are needed to reverse the s-boxes. Our attack requires no a priori knowledge on the s-boxes. However, if we assume that s-boxes satisfy some selected properties, then our attack can be made even more efficient, by a factor of two. Finally our attack needs four times less faults.

Keywords: physical attacks, fault injection attacks, reverse engineering, FIRE, DES-like cryptosystem, s-boxes.

1 Introduction

Although Kerckhoffs' principle [1] enunciates that a cryptosystem should be secure even if its algorithm is public knowledge, private algorithms are still used. For instance, Cryptomeria Cipher (C2) is a proprietary block cipher used to protect DVDs. An other example is the suite of A3 / A5 / A8 algorithms used for authentication / confidentiality / key agreement in GSM applications, that are designed to contain some secret customization. However, creating a strong new cryptosystem from scratch is not easy. It is better to use an algorithm which has gained one's spurs, therefore private algorithms are often derived from well-known algorithms. These algorithms respect some properties identical to their model.

A DES-like cryptosystem or pseudo-DES is an algorithm based on DES [2], but which slightly differs. For example in [3], DES was obfuscated by secret external encoding. In this article, we study a pseudo-DES whose s-boxes differ.

When the goal of an attacker is to retrieve information on a private algorithm, his attack is termed "reverse-engineering". One has to remark that the cipher key is often considered known in a reverse engineering attack.

Algorithms can be secured against algorithmic attacks for reverse engineering. In the case of DES, Luby and Rackoff demonstrated in [4] the following result: no efficient algorithm exists to determine the round function of a more than 3-rounds Feistel, with only observation of couples (plain-text, cipher-text). Despite this security, an algorithm may be vulnerable to physical cryptanalysis for reverse engineering. There are two family of physical cryptanalysis for reverse engineering: SCARE (Side Channels for Reverse Engineering) attacks as in [5,6,7,8] and [9]; and FIRE (Fault Injection for Reverse Engineering) attacks.

In this paper we have chosen to speak about FIRE. Fault injection attacks consist in disrupting the circuit's behaviour in order to alter the correct progress of the algorithm. Biham and Shamir studied fault injection attacks on DES, in [10] and [11] in order to recover the key. The first time that fault injection attacks were used to reverse engineer a DES-like cryptosystem was described by Clavier *et al.* in [3]. But this pseudo-DES was different than the one studied here. Our attack studies a DES-like where s-boxes are customised. Indeed, there is an extremely large choice of s-boxes, whereas the other parts of a Feistel network have less degrees of freedom to be varied. FIRE attacks against secret s-boxes were presented in [12] by San Pedro *et al.* on DES-like and AES-like cryptosystems, and in [13] by Ming *et al.* on substitution-permutation networks and Feistel structures. These state-of-the-art reverse-engineering attacks focus in a generic way on relations in the last round of the Feistel cipher. A feature of our paper is that results are obtained with differentials between s-boxes outputs as in [12] and [13]. But, our approach is different: we fully exploit the knowledge of the cipher structure to design an adapted attack. The main idea is injecting faults in register $R14$ instead of $R15$ as in [12]. The second idea is to consider some properties of DES s-boxes described in [14].

The paper is organised as follows. The section 2 describes the DES. The section 3 presents previous works. Our proposed improvement of FIRE is described in 4. The section 5 presents the algorithm for cryptanalysis and our results. The conclusion is drawn in section 6.

2 Description of DES

Data Encryption Standard (DES) was developed by the National Bureau of Standards [2]. Even if Advanced Encryption Standard (AES [15]) is the new cipher block standard, the DES is not outdated. DES and 3DES are still in use in many devices. Fig. 1 presents DES. DES is a symmetric cryptosystem, specifically a 16-round Feistel cipher. DES starts by IP, a permutation of 64 bits and finishes by its inverse IP^{-1}.

The round function F on 32 bits consists in 4 steps.

- Expansion E which maps 32 bits in 48 bits by duplicating half of the bits. One can note that two bits are used only once in each s-box. And an s-box has two bits in common with the next s-box and two bits in common with the previous s-box; we consider that s-box 1 follows s-box 8.
- Xor with the 48 bits of round key K_j, $j \in [\![1, 16]\!]$.
- S-boxes S_i, $i \in [\![1, 8]\!]$, which substitute a 6-bits input m_i for a 4-bits output y_i We note:

$$S_i(m_i) = y_i \tag{1}$$

In the standard, s-boxes are represented with a table of 4 lines and 16 columns. Let m_i be one input, the first and the last bit establish the line number. The bits in the middle establish the column number[5]. To sum up m_i defines the position in the s-box of a cell and y_i defines the value in the same cell.
For example, if $m_i = (25)_{10} = (011001)_2$, the number of line is $(01)_2 = (1)_{10}$ and that of column is $(1100)_2 = (12)_{10}$. The output y_i is equal to the value in the cell of the table associated with S_i, at line 1, column 12.

– Permutation P of 32 bits.

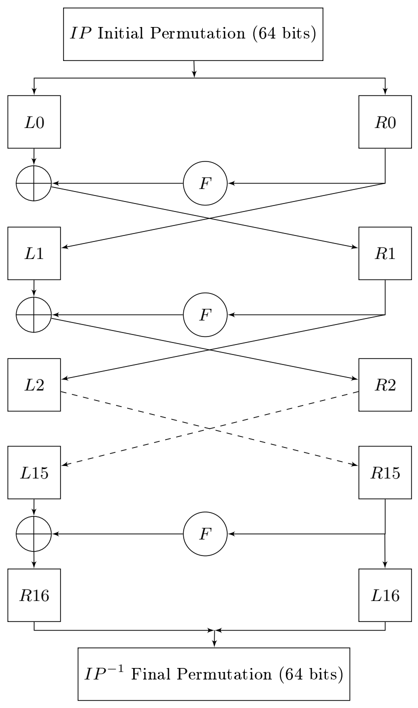
IP Initial Permutation (64 bits)
$L0$
$R0$
F
$L1$
$R1$
F
$L2$
$R2$
$L15$
$R15$
F
$R16$
$L16$
IP^{-1} Final Permutation (64 bits)

Fig. 1. Scheme of Data Encryption Standard

In this paper, we focus on a customised DES-like cryptosystem, whose s-boxes have been changed and are unknown, the rest of the algorithm being genuine, i.e. P, E, IP and the round keys K_j are known.

3 Previous work on FIRE

In this article, a variable marked by an asterisk (*) denotes a data modified by a fault injection; for example x^* is the faulty value of x.

[5] The columns and lines numbering starts at 0.

The attack described in [13] on Feistel structures is generic in the size of the s-boxes and of the linear parts. Because the studied data path is different, it cannot be applied to a pseudo-DES. They use two keys per round instead of one for DES. Their results are presented on a Twofish-like cryptosystem, and they need in average 234 faults.

Our attack is an improvement of the FIRE attack of San Pedro *et al.* [12] on a DES-like cryptosystem. This section summarizes this last one. Unlike in [13] which applies an 8-bits random fault model, the fault model of [12] is a single bit fault: the value of one bit is switched.
The fault occurs in $R15$ in [12]. There is a maximum of two different bits between m^* and m the s-boxes inputs of the last round. To ascertain whether the collected information is exploitable, San Pedro *et al.* focus on the last round as illustrated in Fig. 2.

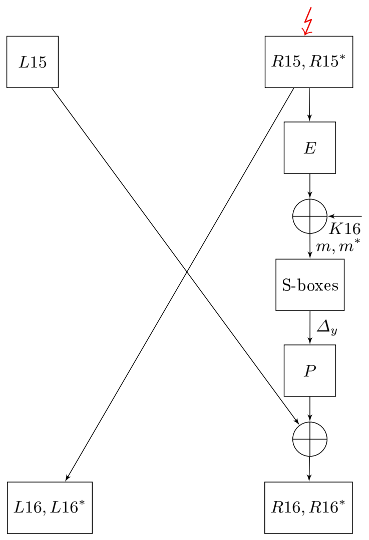

Fig. 2. Attack path of the FIRE attack [12]

Knowing the ciphertext C and the faulty ciphertext C^*, the values of $L16$, $R16$, $L16^*$ and $R16^*$ can be obtained with the permutation IP the inverse function of IP^{-1}. From the description of DES, $L16 = R15$. The round key $K16$ and the expansion E are known. Thus the s-boxes inputs m and m^* can be retrieved with equations (2):

$$\begin{aligned} m &= E(R15) \oplus K16 &= E(L16) \oplus K16 \\ m^* &= E(R15^*) \oplus K16 &= E(L16^*) \oplus K16 \end{aligned} \quad (2)$$

Because $L15$ is unknown, the s-boxes outputs y and y^* cannot be retrieved;

$$\begin{aligned} y &= P^{-1}(R16 \oplus L15) \\ y^* &= P^{-1}(R16^* \oplus L15) \end{aligned}$$

However the differential between the s-boxes outputs:
$\Delta_y = y \oplus y^*$ can be computed as follows:

$$\Delta_y = P^{-1}(R16 \oplus R16^* \oplus L15 \oplus L15)$$

Thus:

$$\Delta_y = P^{-1}(R16 \oplus R16^*) \tag{3}$$

Thereby in FIRE attack, only the inputs m, m^* and the differential Δ_y, can be computed and split. So for $i \in [\![1, 8]\!]$ we have: $S_i(m_i) \oplus S_i(m_i^*) = \Delta_{y_i}$.

From now on, for an s-box S_i we call a relation (4) the link between any two inputs m_{i_0}, m_{i_1} and a differential output Δ_{y_i}.

$$S_i(m_{i_0}) \oplus S_i(m_{i_1}) = \Delta_{y_i} \tag{4}$$

One has to remark that the transitive relation of the xor operator, is used to perform the FIRE attack. It can be applied for relations (4) in the same s-box to obtain more relations (4). Let m_{i_0}, m_{i_1} and m_{i_2} be any three inputs, we have:

$$\left.\begin{array}{l} S_i(m_{i_0}) \oplus S_i(m_{i_1}) = \Delta_{y_i} \\ S_i(m_{i_0}) \oplus S_i(m_{i_2}) = \Delta'_{y_i} \end{array}\right\} \Rightarrow S_i(m_{i_1}) \oplus S_i(m_{i_2}) = \Delta_{y_i} \oplus \Delta'_{y_i} \tag{5}$$

Because only relations (4) are known but not equalities (1), y_i and y_i^* are unknown and the s-boxes cannot be obtained directly. We say that each s-box is "defined up to a translation". Knowing all relations (4) of one s-box S_i, only one couple (m_i, y_i) which satisfies (1) is needed to fully define S_i. For this reason, FIRE attack must finish with an exhaustive search. The attacker must test the 16 values for a cell (an output y_i associated with an input m_i) of each s-box. There are $16^8 = 2^{32}$ guesses to fully define all 8 s-boxes.

In attack [12], 130 faults are needed in average to define one s-box up to a translation. In total, $130 \cdot 8 = 1040$ faults are needed to find all 8 s-boxes.

4 Proposed improvements

This section presents the two improvements of our FIRE attack.

4.1 The Attack Path

Fault model Unfortunately, in fault injection attacks the circuit can be damaged. Sakiyama *et al.* in [16] emphasizes the importance of limiting the number of fault injections. So, our attack is conducted with this goal in mind.

In our attack, the fault attack randomly changes one and only one bit of register $R14$, at the end of the fourteenth round. The fault model is a single bit fault as in [12], but the location of the injection differs.

Biham and Shamir showed the relevance of using injected faults in the penultimate rounds of DES to discover the cipher key. They studied the reverse engineering of s-boxes on DES-like. However they did not use injected faults in the penultimate rounds to discover DES-like s-boxes. Based on these seminal works [10,11], we improved their s-boxes recovery attack, by applying faults in the last but one round. A fault in $R14$ is more propagated than a fault in $R15$,

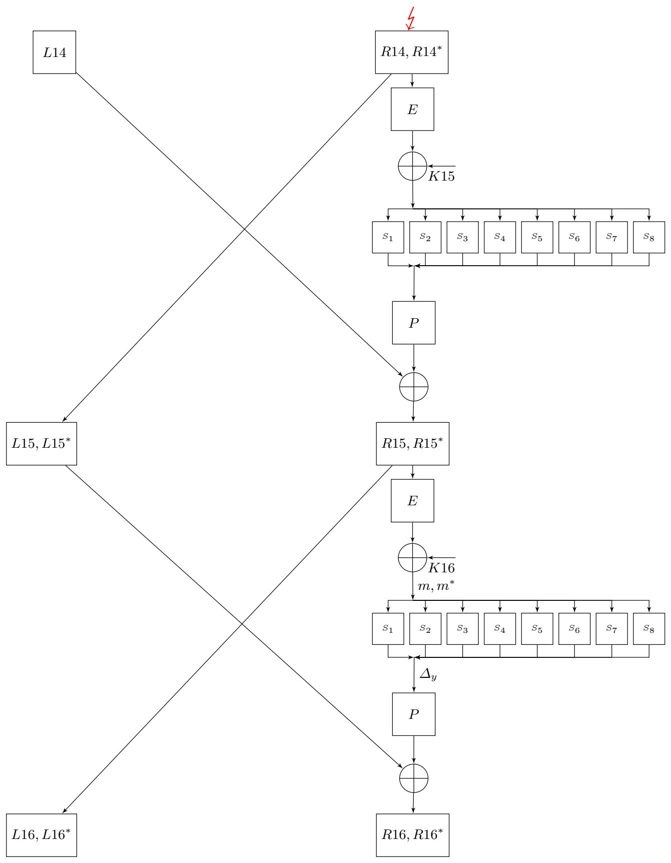

Fig. 3. Propagation of the fault on $R14$ in the round 15 and 16.

i.e. more bits in the s-boxes inputs of the last round are faulted. We note e the fault on $R14^*$, so $e = R14 \oplus R14^*$.

Expansion E duplicates half of the bits of the previous R ($R14$ or $R15$). A faulty bit at an s-box input can change the 4 bits at the output. Thus the combination of expansion and s-boxes can fault 8 bits in $R15^*$. With the expansion on last round, 16 bits of m^* can differ with m. Thanks to this propagation which is illustrated in the Fig. 3, many s-boxes can be attacked at the same time. The injection of one faulty bit is rarely uniform, thus with a single-bit fault injected in $R15$ as in [12], some s-boxes may be difficult to reach. This problem does not exist in our attack.

One might argue that targeting precisely $R14$ might seem tough; but in practice, it is possible to know if a fault occurred in the last or the last but one round. If $L16$ has more than 8 faulty bits, then the fault occurred earlier than $R14$, and shall be discarded since not usable by our attack. Otherwise if $L16$ contains less than 9 faulty bits, the fault occurs in $R14$. One has to remark that if $L16$ contains only one faulty bit the fault can have occurred in $R15$. Although such a fault conveys less information than one in $R14$, it can nonetheless be exploited.

As explained in section 3, we focus on the last round as explained in section 3, knowing only $R16$, $L16$, $R16^*$ and $L16^*$. The inputs m and m^* can be computed with equation (2). Since the fault is injected in $R14$ instead of $R15$:
$R14^* = L15^* \neq L15$. So :

$$\Delta_y = P^{-1}(R16 \oplus R16^* \oplus L15 \oplus L15^*) = P^{-1}(R16 \oplus R16^* \oplus e)$$

Thus, Δ_y cannot be computed as in [12]. In our attack we compute instead:

$$\Delta_z = P^{-1}(R16 \oplus R16^*) \tag{6}$$

One has to remark that $\Delta_z \neq \Delta_y$. It comes from the fact that, $L15^* = L15 \oplus e$, so:

$$\Delta_y = \Delta_z \oplus P^{-1}(e) \tag{7}$$

Δ_z and Δ_y differs only by one bit. We must find the value of e in order to get Δ_y. The steps to retrieve e are described in the next paragraph 4.1.

An unknown differential We are now interested in finding e. For that purpose we study Δ_m. Let Δ_m be the differential between s-boxes inputs in the last round.

$$\Delta_m = m \oplus m^*$$

Δ_m can be computed and then split in eight Δ_{m_i}, one for each s-box.

Case where e is discovered $\Delta_{m_i} = 0 \Rightarrow \Delta_{y_i} = 0$. Indeed for a given s-box S_i if inputs m_i and m_i^* are equal, output y_i equals y_i^*. If $\Delta_{m_i} = 0$ and $\Delta_{z_i} \neq 0$ the fault is found; the faulty bit is the bit of Δ_{z_i} which equals 1 up to the permutation P^{-1}. We have found e thus Δ_y can be computed with (7). For all s-boxes we have relations (4). In practice for 10000 faults, this case occurs with a probability equals to 0.56.

Case where e stays unknown In this paragraph we list the different values of e which may explain our experiment. We note e_b the guessed fault for e such that all bits of e_b are equal to zero except the bits b.

If there are no $\Delta_{m_i} = 0$ with $\Delta_{z_i} \neq 0$, input differentials Δ_{m_i} must be examined in more details. The differentials $\Delta_{m_i} \neq 0$ indicate s-boxes which have a faulty output in round 15. Table 1 indicates the s-box in round 15 from which each bit of m_i comes from.

Table 1. Origin of the 6 bits of Δ_{m_i} in the last round from the s-boxes in round 15.

round 16 \ bits	1	2	3	4	5	6
Δ_{m_1}	S_7	S_4	S_2	S_5	S_6	S_8
Δ_{m_2}	S_6	S_8	S_3	S_7	S_5	S_1
Δ_{m_3}	S_5	S_1	S_4	S_6	S_7	S_2
Δ_{m_4}	S_7	S_2	S_5	S_8	S_3	S_1
Δ_{m_5}	S_3	S_1	S_2	S_6	S_4	S_8
Δ_{m_6}	S_4	S_8	S_7	S_1	S_3	S_5
Δ_{m_7}	S_3	S_5	S_4	S_8	S_2	S_6
Δ_{m_8}	S_2	S_6	S_3	S_1	S_7	S_4

An observed faulty output implies a faulty input. In round 15, either two s-boxes have a faulty output or only one s-box has a faulty output.

- *Two s-boxes have a faulty output in round* 15.

If two s-boxes have a faulty output, there are always only two possible faulty bits in $R14^*$, noted b_1 and b_2 which may explain Δ_m. The table of expansion is used to retrieve them. There are two possible values e_{b_1} and e_{b_2}. One has to remark that there are 6 s-boxes S_i for which $(P^{-1}(e_{b_1})))_i = (P^{-1}(e_{b_2})))_i = 0$; thus Δ_{y_i} is known and equals to Δ_{z_i}. We have 6 relations (4).

Suppose that the attacker observes that bits 6 of Δ_{m_1}, 4 of Δ_{m_2}, 1 and 4 of Δ_{m_4} are equal to one. Table 1 indicates that in round 15 s-boxes S_7 and S_8 have faulty outputs. In the table of expansion, bits 28 and 29 of $R14$ are shared on S_7 and S_8. There are two possible values of e: e_{28} and e_{29}. $P^{-1}(\{28, 29\}) = \{6, 22\}$. The bit 6 is at the output of S_2, and the bit 22 is at the output of S_6. For s-boxes S_i, $i \in \{1, 3, 4, 5, 7, 8\}$, $\Delta_{y_i} = \Delta_{z_i}$.

- *Only one s-box has a faulty output in round* 15.

The attacker observes that only one s-box S_i output is modified in round 15 and the faulty bit does not affect necessarily only one s-box S_i input. It may affect the inputs of two s-boxes, either S_{i-1} and S_i or S_i and S_{i+1}. In s-boxes two inputs may have the same output. Thus, if the single bit fault modifies the input, the output could stay unchanged. Note that the case where only one s-box is affected is more likely. As a consequence, six single bit faults $R14^*$ are possible. They can be retrieved with the expansion E. There are six possible values e_{b_1}, e_{b_2}, e_{b_3}, e_{b_4}, e_{b_5} and e_{b_6}. One has to remark that there are 2 s-boxes S_i for which $(P^{-1}(e_{b_j})))_i = 0$; thus Δ_{y_i} is known and equals to Δ_{z_i}. We have 2 relations (4).

For example, the attacker observes that bits 6 of Δ_{m_4}, 2 of Δ_{m_5} and 3 of Δ_{m_7} are not equal to zero. Table 1 indicates that in round 15 only S_1 has faulty outputs. The set of bits $\{32, 1, 2, 3, 4, 5\}$ of $R14$ determines the input of S_1. There are six possible values of e: e_{32}, e_1, e_2, e_3, e_4 and e_5 $P^{-1}(\{32, 1, 2, 3, 4, 5\}) = \{25, 16, 7, 20, 21, 29\}$. The bit 25 respectively $16, 7, 20, 21$ and 29 is at the output of S_7, S_4, S_2, S_5, S_6 and S_8. For s-boxes S_i, $i \in \{1, 3\}$, $\Delta_{y_i} = \Delta_{z_i}$.

In conclusion for any Δ_m we have a set of possible fault values in $R14^*$. More precisely there are one, two or six possible fault values, noted e_{b_j}. Each e_{b_j} gives an equation (8).

$$S(m) \oplus S(m^*) = \Delta_z \oplus P^{-1}(e_{b_j}) \tag{8}$$

We call I this set of equations, where only one of these equations is correct.

This set can be split in eight subsets I_i, one for each s-box. If $|I| = 1$, we have Δ_y thus we have 8 relations (4). If $|I| = 2$, we have 6 relations (4). And if $|I| = 6$, we have 2 relations (4).

It may be noted that many simplifications appear in the sub-sets I_i. The b_j are always next to each other. When going through P^{-1} these bits are dispatched to different s-boxes. Thus an s-box S_i is concerned by only one b_j. As a consequence, for $l \neq j$, $(P^{-1}(e_{b_l}))_i = 0$ and:

$$\Delta_{y_i} = \Delta_{z_i} \text{ or } \Delta_{z_i} \oplus (P^{-1}(e_{b_j}))_i$$

Thus a fault e gives, for each s-box such that $m_i \neq m_i^*$, a maximum of two possible differentials Δ_{y_i}.

Even if the fault value e is unknown when it is injected, the set of possibilities is very small. We have a lot of information on Δ_y, which in the end lead us to define the s-boxes up to a translation.

4.2 S-box properties

S-boxes of a DES-like cryptosystem can be a priori any $\{0,1\}^6 \to \{0,1\}^4$ function. Obviously such functions are not invertible, but this is anyway not a requirement since the DES-like decipherment also uses the direct s-boxes. However, to resist cryptanalytic attacks, it is highly recommended that the s-boxes respect some properties. Some desired properties of s-boxes are described and justified by Brickell in [14].

We decided to take into account only the two following properties, which are used in many pseudo-DES ciphers, as for example in DESL [17][6]

Of course, different or additional constraints could be assumed as well, without fundamentally changing our conclusions.

Property P1: *Changing 1 input bit of an s-box results in changing at least 2 output bits.*

P1 implies that a faulty bit at the input of one s-box changes its output. Thus the case where the output of one s-box affected by a single bit fault e is unchanged is excluded. So, P1 reduces the number of possible single bit faults b_j from six to two. With this property and the paragraph 4.1, there are a maximum of only two possible faults in $R14^*$, noted e_{b_1} and e_{b_2}.

[6] Let us notice that DESL is *not* primarily intended to resist reverse-engineering attacks, but to be more lightweight [18] than the genuine DES. Still, it is an illustration that trading s-boxes for others can done safely.

Let's use the example of the paragraph 4.1. Only the bits 2 and 3 are only at the input of S_1, the other bits are also at the input of S_8 and S_2. There are two possible error values e_2 and e_3.

Property P2: *Each line of an s-box is a permutation of the integers 0 to 15.*
Property P2 implies that a line contains 16 different values. If the values of 15 cells on the same line are known; by elimination, the last cell is equal to the missing integer in interval $[\![0, 15]\!]$. The number of possible s-boxes is $(16!)^{4 \cdot 8} \approx 2^{1376}$. It would be natural to think that $4 \cdot 16 - 1 = 63$ relations are needed to define an s-box up to a translation. But, P2 implies that with 14 adequate relations we can find all the relations on this line. Thus to define an s-box up to a translation, $4 \cdot 14 = 56$ relations are sufficient.

Property P2 allows to eliminate wrong hypotheses for differentials Δ_{y_i}. Indeed, let m_{i_0}, m_{i_1} be any two inputs which are associated with the same line; $S_i(m_{i_0}) = y_{i_0} \neq y_{i_1} = S(m_{i_1})$. Thus, $\forall m_{i_2}$, an input such that $S_i(m_{i_2}) = y_{i_2}$, we have:

$$S_i(m_{i_0}) \oplus S_i(m_{i_2}) = y_{i_0} \oplus y_{i_2} \neq y_{i_1} \oplus y_{i_2} = S_i(m_{i_1}) \oplus S_i(m_{i_2})$$

In practice, for a relation (4) with $m_{i_0} \neq m_{i_1}$ we have one of the following properties: either (9) or (10).

• m_{i_0}, m_{i_1} belong to the same line. Let m_{i_2} be any input, different from m_{i_0} and m_{i_1}:

$$S_i(m_{i_0}) \oplus S_i(m_{i_1}) = \Delta_{y_i} \Rightarrow \begin{cases} S_i(m_{i_0}) \oplus S_i(m_{i_2}) \neq \Delta_{y_i} \\ S_i(m_{i_1}) \oplus S_i(m_{i_2}) \neq \Delta_{y_i} \end{cases} \tag{9}$$

• m_{i_0} and m_{i_1} are associated with two different lines l and l'. Let $m_{i_2} \neq m_{i_0}$ be an input associated with l. Let $m_{i_3} \neq m_{i_1}$ be an input associated with l':

$$S_i(m_{i_0}) \oplus S_i(m_{i_1}) = \Delta_{y_i} \Rightarrow \begin{cases} S_i(m_{i_0}) \oplus S_i(m_{i_3}) \neq \Delta_{y_i} \\ S_i(m_{i_1}) \oplus S_i(m_{i_2}) \neq \Delta_{y_i} \end{cases} \tag{10}$$

In FIRE attack [12], property P2 is not fully exploited. Thereby properties (9) and (10) are not used.

5 Implementation and results

5.1 Experimental Validation

In order to validate the fault model, we conducted an experimental campaign of faults against a smart-card that embeds a hardware DES accelerator. Of course, the DES module in this smart-card is genuine (i.e. the s-boxes are original), but the kind of faults that will appear is likely to be representative of a DES despite its s-boxes are customized.

The smart-card is fabricated in CMOS 130 nm technology. Internally, the DES runs at 66 MHz under a voltage supply of 1.3 V. The DES co-processor computes one round per clock cycle. Its s-boxes are implemented in logic gates, and not in ROM. It is known that in such designs, the critical path goes through the s-boxes, because they are made up of several layers of gates.

The experiment consisted in reducing the power supply and to check for the integrity of the DES computation. For each voltage value, the DES module was submitted 1000 single-block encryptions in ECB (Electronic Code Book) mode of operation (i.e. without initial vector nor chaining).

The produced cipher-texts were processed with two metrics. The first one is simply the *proportion of incorrect encryptions*. Intuitively, the smaller the supplied voltage, the more incorrect encryptions are obtained. The second metric is the *proportion of incorrect encryptions caused by a bit-flip*. To compute this metric, a software version of the DES has been programmed, in which any intermediate bit can be flipped: thus, there are 64×16 possible locations, corresponding to the 64 bits of the LR register and to the 16 rounds. If for one of these positions, the software yields the same erroneous cipher-text as obtained experimentally, then it is confirmed that this a bit-flip indeed occurred in the circuit. Of course, the proportion of incorrect encryptions is greater than proportion of incorrect encryptions caused by a bit-flip. The difference is labeled "multiple faults".

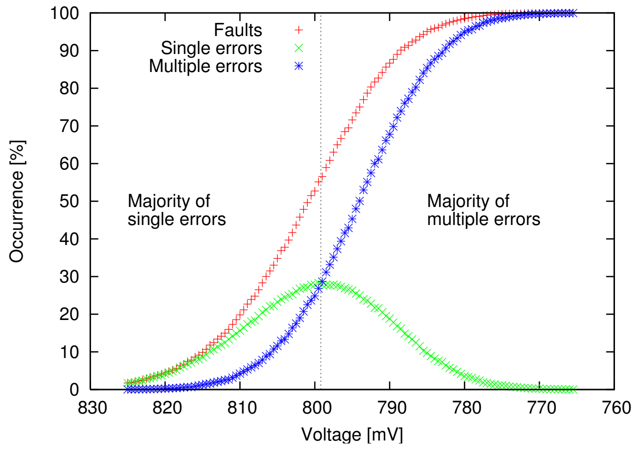

Fig. 4. Proportion of incorrect encryptions, and proportion of incorrect encryptions due to bit-flips, as obtained from a hardware DES module.

The results are shown in Fig. 4. It can be seen that the DES starts to produce erroneous encryptions at a relatively low voltage. Nonetheless, the chip remains functional, and so the voltage can be lowered even more. We can see that there is a range, between 0.825 and .820, where all the faults are bit-flips. Below this interval, a bit-flip cannot explain the fault found in the ciphertext. This is due to the fact many bit-flips occur, which hinders our reverse-engineering attack. Nonetheless, our attack perfectly works in the identified voltage range $[0.825, 0.820]$.

5.2 Implementation

Graphs The relations in s-boxes can be represented with eight disjoint graphs G_i, one for each s-box. The nodes of a graph represent the 64 different input values of an s-box S_i. Two nodes are linked by one edge, such that its weight is equal to the number of possible Δ_{y_i} for these inputs. At the beginning all possible differentials are possible. The weight of an edge is equal to 16, or 15 if the edge connects two inputs of a same line. The attack ends when all the weights are equal to one. Eventually, the total weight is equal to $\sum_{i=1}^{63} i = 2016$.

Algorithms In A, Algorithm 1 *Attack* keeps running until s-boxes are retrieved, i.e. all G_i, $i \in [\![1, 8]\!]$ have not a total weight of 2016. A fault injection e gives a set I of equations (8) as explained in 4.1 . It is stored in a list L. It is split in eight, and each part is exploited by the sub-function Algorithm 2 *Update_ graph* which updates graphs G_i, $i \in [\![1, 8]\!]$.

More precisely Algorithm 2 in A uses relations r associated with an s-box S_i. A relation r contains two inputs m_i and m'_i and a list of possible Δ_{y_i} extracted from I_i or from properties propagation. If there is only one Δ_{y_i}, r is a relation (4). In this case properties (5) and (9) or (10) can be applied.

- *Simplify r:* if the number of Δ_{y_i} can be reduced, i.e. there are some e_{b_j} such that $(P^{-1}(e_{b_j}))_i = 0$.
- *Update G_i with r:* the Δ_{y_i} listed in r stays assigned to the edge which links the m_i node with the m^*_i node. The weight of this edge becomes equal to the number of possible Δ_{y_i} in r.
- *Update L_i with r:* if it can eliminate some e_{b_j} in a set of equation I.
- *Update G_i with* (5)*:* new relations are computed with r, G_i and the transitive property of xor (5).
- *Update G_i with property* (9) *or* (10)*:* on the s-box lines: Δ_{y_i} is withdrawn from the edges which link the node m_i, according to property (9) or (10).

5.3 Results

We have simulated 1000 attacks with Matlab. The statistical results are detailed in Table 2. Our experiments showed that 423 faults were needed in average to define all s-boxes up to a translation thanks to the injection of faults in $R14$. It is 2.45 times better than the previous FIRE attack [12] which inject faults in $R15$ and needs 1040 faults.

Table 2. Statistics about the number of faults necessary to succeed an attack (estimated from 1000 attacks)

statistic tool	without P1 and P2	with P1 and P2
mean	423.07	234.76
standard deviation	63.30	34.08
median	413	231
minimum	313	168
maximum	654	394

Furthermore taking into account P1 and P2 improves again our attack. Property P2 decreases greatly the needed number of faults. Property P1 simplifies the algorithm, since it reduces the size of the set I. Indeed, only 234 faults were needed in average. Finally our attack is 4.44 times better than the previous FIRE attack [12].

We study an example of attack, illustrated in Fig. 5. This attack needed 352 faults to define s-boxes up to a translation. Compared to the mean this case is deliberately chosen bad in order to observe the attack behaviour. The number of new relations obtained at each fault is given in Fig. 5.

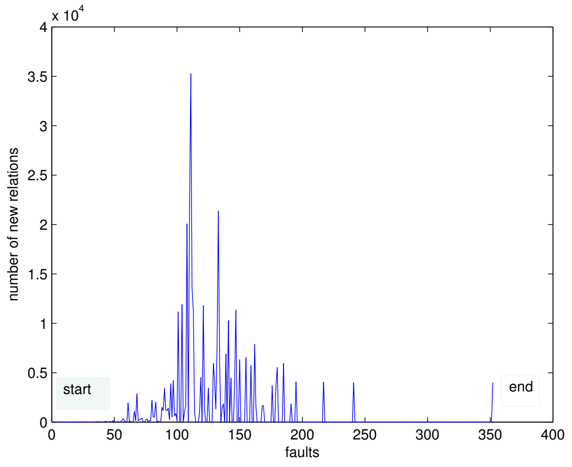

Fig. 5. Number of new relations obtained at each fault for an instance of an attack which needs 352 faults

At the start, a fault gives between 2 and 8 relations. It is expected since at the beginning we have only few relations in graphs. Starting from approximately 50 faults, the algorithm obtains a lot of relations. The different properties allow to bring out many relations. The number of new relations (4) discovered at each fault injection, starts by increasing at the beginning of the attack and then decreases to zero. Starting from approximately 250 faults to the last one, faults bring almost no new relations.

Due to the randomness of faults, at the end of an attack, Algorithm 1 *Attack* waits for an adequate fault to be injected. The other faults do not give any new relations. Algorithm 1 eliminates all relations obtained by these faults, because there are already known. Algorithm 2 *Update_ graph* is not called any more.

Exhaustive search The previous paragraph shows that the last faults in attack do not bring a lot of information. In this paragraph, the new idea is to stop the attack before s-boxes are defined up to a translation and finish in exhaustive search.

When Algorithm 1 *Attack* is stopped at a chosen number of faults, the total graph weight is different than 2016. There are edges with a weight not equal to 1. Relations (4) have to be guessed to find the s-boxes.

Using properties (5), (9) and (10), each new guess eliminates some possible relations (4) and thus modifies the weight of several edges. The weight of the minimum spanning tree of the graph is the upper bound on the number of guesses. Indeed this weight represents the total number of guesses if guesses are independent. Kruskal's algorithm [19] allows us to find the spanning tree and its weight. Thanks to this algorithm we have a correct maximization of the number of guesses to finish our attack.

Table 3. Results for 100 attacks with different numbers of faults

Number of faults	Average of number of s-boxes which are retrieved up to a translation	Median of maximal number of guesses to define s-boxes up to a translation	Maximum number of guesses to totally define s-boxes
120	0.04	$4.549 \cdot 10^{42}$	2^{174}
140	0.89	$9.5105 \cdot 10^{14}$	2^{82}
160	2.76	62208	2^{47}
180	4.53	16	2^{36}
200	6.06	8	2^{35}
220	6.93	4	2^{33}
240	$7,5$	0	2^{32}

Table 3 presents three results for a given number of faults on 100 attacks. First, it gives the number of s-boxes in average which are defined up to a translation. The second column indicates the maximum number of guesses which are used to retrieve s-boxes up to a translation. We prefer to use the median rather than the mean because of the presence of sparse extreme values which alter the representative power of the mean number. The last column gives a rough estimate of the number of guesses needed to define fully the s-boxes. It takes into account the 2^{32} guesses on s-boxes when there are defined up to a translation.

According to his computational power, an attacker can stop the attack at the number of fault given in Table 3. For example, with a computational power of 2^{47}, we need only 160 faults to retrieve fully the 8 s-boxes by an achievable exhaustive search. Recall that the exhaustive search of the s-boxes (without FIRE) would require an infeasible amount of $2^{4\times 2^6 \times 8} = 2^{2048}$ hypotheses.

We have briefly studied a single bit fault attack on $R13$. But a single bit fault affects too many bits in Δ_z. The guesses on the injected fault are too numerous to be operated.

6 Conclusion

In this paper we have presented a FIRE attack on a pseudo-DES which has unknown s-boxes. Our attack is inspired by [12] and decreases the required number of faults. Our improvement is based on the innovative idea to exploit faults striking the penultimate round ($R14$) instead of the last one ($R15$). Although the differential equations are more complicated, they are also more rich in that they allow a parallel attack on a large number of s-boxes. This cuts the number of faults to succeed the attack by a factor of two. Furthermore, our attack has the remarkable property that it can easily be educated with a priori information about the s-boxes. Typically, by considering basic properties of balancedness and propagation criteria, the attack's convergence can be sped up by an additional factor two.

Concretely, we show that by considering this fault injection, 423 faults were needed in average to define all s-boxes up to a translation. It is 2.45 times better than the previous FIRE attack [12] which injects faults in $R15$. Our attack requires no a priori knowledge on the s-boxes. However, in practice, it is desirable that Feistel schemes' s-boxes have some properties. We show that if we assume that s-boxes have some selected properties, then our attack can be made even faster. Indeed, this strategy is winning since it converges to the correct solution in about 4.44 times fewer faults than the FIRE attack [12]. Finally we need 234 faults in average to define 8 s-boxes up to a translation. The attack limits the number of fault injections which can damage the circuit.

Furthermore we show that a trade-off between the number of injected faults and the final exhaustive search can be tuned. For example, with a computational power of 2^{47}, we need only 160 faults to retrieve fully the 8 s-boxes by an achievable exhaustive search.

Acknowledgements. The authors would like to thank Ronan Lashermes, Manuel San Pedro, Guillaume Reymond and Loïc Zussa for their valuable contributions to the development and understanding of the issues discussed in the paper.

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. Parts of this research has been founded by the **MARSHAL+** project, "*Mechanisms Against Reverse-engineering for Secure Hardware and Algorithms*", from call FUI 12, co-labellized by the competitivity clusters System@tic et SCS.

References

1. Kerckhoffs, A.: La cryptographie militaire. Journal des sciences militaires **9**(1) (1883) 5–38
2. NIST: Data Encryption Standard. FIPS PUB 46-3 (1977)
3. Clavier, C.: Secret external encodings do not prevent transient fault analysis. Cryptographic Hardware and Embedded Systems-CHES 2007 (2007) 181–194
4. Luby, M., Rackoff, C.: How to construct pseudo-random permutations from pseudo-random functions. In: Advances in Cryptology–CRYPTO. Volume 85. (1986) 447
5. Daudigny, R., Ledig, H., Muller, F., Valette, F.: SCARE of the DES. In: Applied Cryptography and Network Security, Springer (2005) 19–33
6. Guilley, S., Sauvage, L., Micolod, J., Réal, D., Valette, F.: Defeating any secret cryptography with SCARE attacks. Progress in Cryptology–LATINCRYPT 2010 (2010) 273–293
7. Clavier, C.: An improved SCARE cryptanalysis against a secret A3/A8 GSM algorithm. Information Systems Security (2007) 143–155
8. Clavier, C., Isorez, Q., Wurcker, A.: SCARE of AES-like Block Ciphers by Chosen Plaintext Collision Power Analysis. In LNCS, ed.: INDOCRYPT. (Dec 7-10 2013) Mumbai, India.
9. Rivain, M., Roche, T.: SCARE of Secret Ciphers with SPN Structures. In LNCS, ed.: ASIACRYPT. (Dec 1-5 2013) Bangalore, India.
10. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems. In: CRYPTO. (1997) 513–525
11. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of CRYPTOLOGY **4**(1) (1991) 3–72
12. San Pedro, M., Soos, M., Guilley, S.: FIRE: Fault Injection for Reverse Engineering. In LNCS, ed.: Security and Privacy of Mobile Devices in Wireless Communication (WISTP). Volume 6633., Springer (2011) 280–293
13. Ming, T., Zhenlong, Q., Hui, D., Shubo, L., Huanguo, Z.: Reverse Engineering Analysis Based on Differential Fault Analysis Against Secret S-boxes. China Communications **9**(10) (2012) 10
14. Brickell, E., Moore, J., Purtill, M.: Structure in the S-Boxes of the DES. In: Advances in Cryptology—CRYPTO'86, Springer (1987) 3–8
15. NIST: Specification for the Advanced Encryption Standard. FIPS PUB 197 **197** (November 2001)
16. Sakiyama, K., Li, Y., Iwamoto, M., Ohta, K.: Information-theoretic approach to optimal differential fault analysis. IEEE Transactions on Information Forensics and Security **7**(1) (2012) 109–120
17. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New lightweight DES variants. In: Fast Software Encryption, Springer (2007) 196–210
18. Poschmann, A., Leander, G., Schramm, K., Paar, C.: New Light-Weight Crypto Algorithms for RFID. In: ISCAS. (2007) 1843–1846
19. Kruskal, J.B.: On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Proceedings of The American Mathematical Society **7** (1956) 48–48

A Appendix

Algorithm 1 *Attack* : inject faults until s-boxes are defined up to a translation.

```
Output : 8 graphs $G_i$, $i \in [\![1, 8]\!]$ of total weight 2016 for 8 s-boxes.

Initialization of 8 graphs $G_i$
Create an empty list $L$ of sets $I$ associated with $e$
while All $G_i$ have not a total weight of 2016 do
  Generate a new couple ciphertext, faulty ciphertext $(C, C^*)$
  Compute $m, m^*$ with equation (2) and $\Delta_z$ with (6)
  Write $I$ (the size of $I$ depends on the use P1 or not)
  $L = [L, I]$
  Split $I$ in sub-sets $I_i$
  for $i = 1$ to 8 do
    $Update\_graph(I_i, G_i, L)$
  end for
end while
Return the 8 graphs $G_i$
```

Algorithm 2 *Update_graph* : Store and process relations (4) and (8) for a given s-box

```
Input : one graph $G_i$, $i \in [\![1, 8]\!]$
$L$ list of sets $I$ (8)
$I_i$ a sub-set of equations on $S_i$
Output : a graph $G_i$, a list $L$

$l = r$ obtained from $I_i$
while $l$ is not empty do
  Take the first relation $r$ in $l$
  Simplify $r$ if it is possible
  if $r$ is new relation in $G_i$ then
    Update $G_i$ with $r$
    Update $L$ with $r$
    if $r$ is a relation (4) then
      Update $G_i$ with property (5)
      if $m_i$ and $m'_i$ are on same line then
        Update $G_i$ with property (9) (Only if P2)
      else
        Update $G_i$ with property (10) (Only if P2)
      end if
    end if
    Store new relations in $l$
  end if
  Remove $r$ from $l$
end while
```