On Fault Injections in Generalized Feistel Networks

Hélène Le Bouder, Gaël Thomas, Yanis Linge, Assia Tria

On Fault Injections in Generalized Feistel Networks

Hélène Le Bouder*†, Gaël Thomas‡, Yanis Linge§ and Assia Tria¶†
*École Nationale Supérieure des Mines de Saint Etienne
Email: lebouder@emse.fr
†Centre de Microélectronique de Provence Gardanne France
‡XLIM – UMR CNRS 7252
123, avenue Albert Thomas – 87060 Limoges Cedex
Email: gael.thomas@xlim.fr
§STMicroelectronics Rousset France
¶CEA Tech

***Abstract*—In this paper, we propose a generic method to assess the vulnerability to Differential Fault Analysis of generalized Feistel networks (GFN). This method is based on an in-depth analysis of the GFN properties. First the diffusion of faults is studied, both at the block level and at the S-box level, in order to have a fault which maximizes the number of S-boxes impacted by a fault. Then the number of faults in an S-box required to find the key is evaluated. By combining these results, a precise assessment of the vulnerability to fault attacks of GFN can be made. This method is then used on several examples of Feistel ciphers.**

***Index Terms*—Fault injections attack, differential fault analysis, Generalized Feistel Networks**

I. Introduction

Feistel and generalized Feistel networks are a family of widespread block symmetric ciphers. Fault injection attacks consist in disturbing the circuit behaviour in order to alter the correct processing of the algorithm. Among those, the Differential Fault Analysis (DFA) is a cryptanalytic technique that exploits erroneous results of fault injections. Since injecting faults on a cryptographic circuit may irreversibly damage the device, limiting the number of fault injections is often a critical aspect of DFA. In this paper, a generic method is proposed to evaluate the vulnerabilities of the block registers in generalized Feistel networks. This method is based on the study of the diffusion properties both of the function used inside the Feistel network and of the Feistel network itself.

The fewer single-bit faults are required in a block to find the key, the more vulnerable this block is. This knowledge allows an attacker to choose the best target for its fault injection, but it also enables the designer to focus countermeasures on the vulnerable spots.

No fault attack is presented in this paper, we focus on the diffusion properties to assess a "potential" vulnerability but do not perform a complete fault attack since it is not our objective.

The paper is organised as follows. Section II gives the context of generalized Feistel networks and fault attack injections on them. Our approach is described in Section III. Section IV presents our results on some examples of known generalized Feistel networks. Finally, the conclusion is drawn in Section V.

II. Context and State of the art

A. Feistel Networks

1) Definition: A Feistel network is an iterative construction allowing to create a bijection from functions that need not be bijective. It was invented at IBM by Horst Feistel in the 1970's, during the development of the Lucifer cipher, later known as the Data Encryption Standard (DES) [18]. A Feistel network divides the input into two blocks of equal size. One of these two blocks goes into a subkey-dependent function, called the Feistel function, and the result is xored to the other block, as depicted on Fig. 1. This construction is then repeated several times, switching the roles of the two blocks each time. The main characteristics of this construction are:

- only half of the state is modified, using the Feistel function.
- the construction is invertible, even if the Feistel function itself is not.
- the inverse bijection follows the same construction up to reversing the subkeys order.

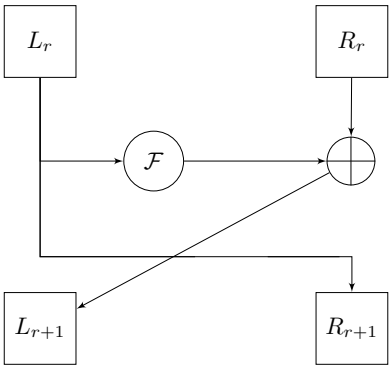

Fig. 1. A classical (two blocks) Feistel cipher.

Introduced at CRYPTO'89 by Zheng *et al.* [19], Generalized Feistel Networks (GFNs) are a broader class of schemes taking the main ideas of the Feistel network but dividing the internal state into more than two blocks. As done in [2], one round of a GFN is divided into two successive transformations: the non-linear layer and the permutation layer. The non-linear layer is

made of one or more key-dependent Feistel functions $\mathcal{F}$. They have some of the blocks of the GFN internal state as input and their output is xored to some other blocks. The permutation layer consists in rearranging the different blocks of the internal state. An example of GFN is given on Fig. 2.

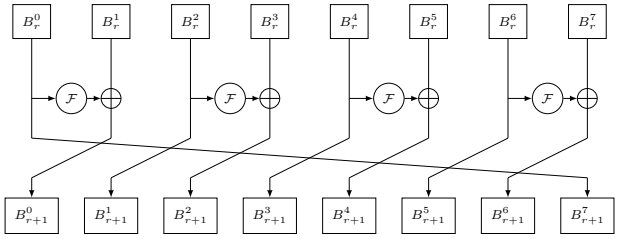

Fig. 2. Example of a generalized Feistel network with 8 blocks.

The blocks of a generalized Feistel network are denoted B_r^i where i is the register/block index (ranging from 0 to $\mathbf{b}-1$ for $\mathbf{b}$ blocks) and r is the current round (ranging from 0 to $\mathbf{r}$ for $\mathbf{r}$ rounds).

2) The Feistel function: We suppose that the Feistel function $\mathcal{F}$ is made of three kinds of component.

- The Xor with the subkey. Since in a single round, more than one block can go into an $\mathcal{F}$ function, we denote the different subkey blocks K_r^λ with $\lambda \in [\![0, \Lambda-1]\!]$ for Λ subkey blocks. For example, the GFN on Fig. 2 has $\Lambda = 4$ subkey blocks.
- The S-boxes. They are the non-linear part of the GFN and provide Shannon's confusion.
- The "mixing" functions. They are linear and provide diffusion inside a block.

B. Examples of Feistel ciphers

1) DES: The Data Encryption Standard (DES) [18] was developed by IBM and standardized by the NIST in 1977. It is the most famous Feistel cipher and is depicted on Fig. 3. Prior to the 16-round Feistel network itself, a bit-wise permutation IP is applied to the 64-bit internal state. The inverse permutation IP^{-1} finishes the encryption process. The Feistel function $\mathcal{F}$ is illustrated on Fig. 4. It acts on 32 bits and consists in 4 steps:

- Expansion E which maps 32 bits into 48 bits by duplicating half of the bits.
- Xor with the 48-bit subkey K_r, $r \in [\![1, 16]\!]$.
- S-boxes S_i, $i \in [\![1, 8]\!]$, which substitute a 6-bit input for a 4-bit output.
- Bit-wise permutation P of 32 bits.

2) MIBS: MIBS is a cipher presented at CANS'09 [6] and aimed at low-resource environments such as RFID tags or sensor networks. It is made of 32 iterations of a classical Feistel networks (see Fig. 1) acting on two blocks of 32 bits each and accepts key of 64 or 80 bits.

As depicted on Fig. 5, MIBS Feistel function operates on 32 bits divided into 8 4-bit words (nibbles) and is made of three consecutives steps:

- Xor with the subkey.

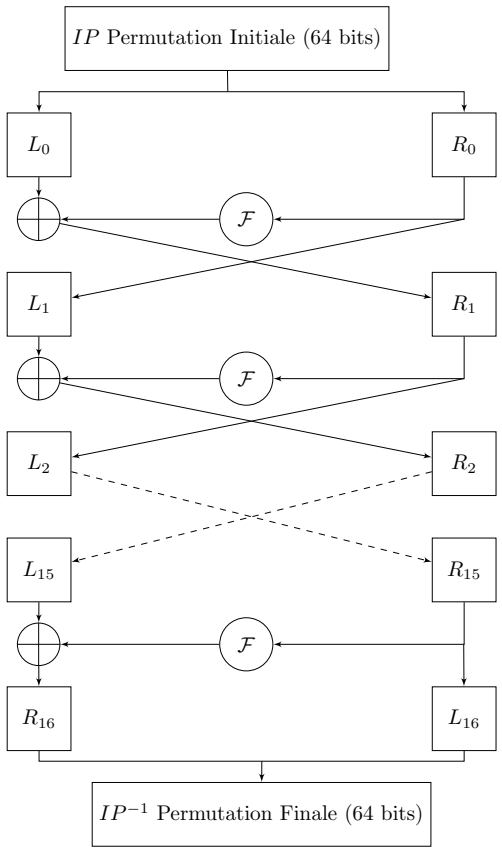

Fig. 3. Overview of Data Encryption Standard.

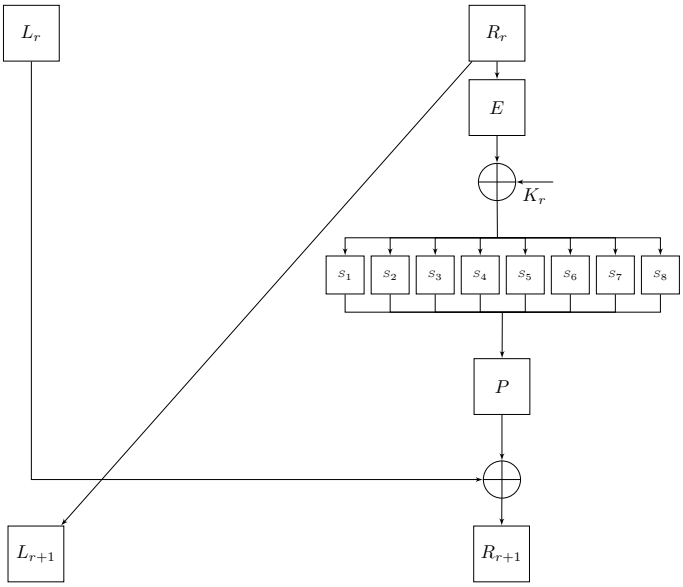

Fig. 4. Feistel function $\mathcal{F}$ of DES.

- a layer of 8 parallel 4×4 S-boxes.
- a linear mixing layer MC acting at nibble level.

3) TWINE: TWINE is a 64-bit block cipher presented at SAC '12 [16] aiming at a reduced circuit area while avoiding non-software-friendly operations (such as bit permutations) to

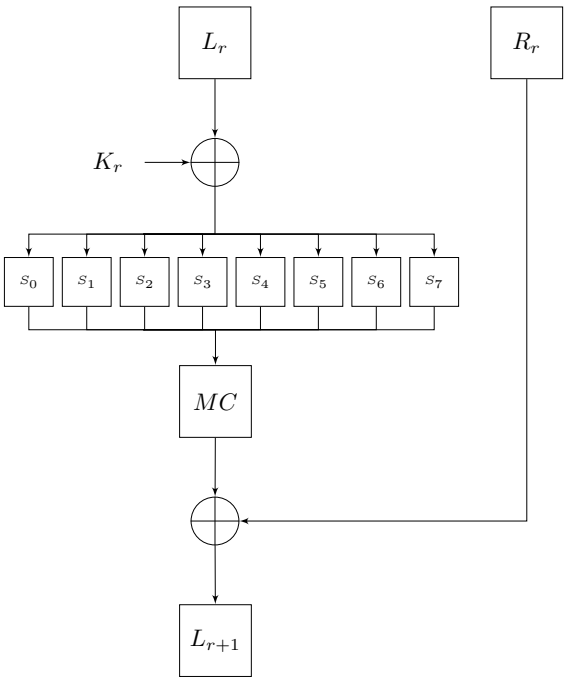

Fig. 5. Structure of the MIBS Feistel function.

maintain decent performances when implemented in software. It is a GFN with 16 blocks, 4 bits each, that can accept either 80-bits or 128-bits keys. The permutation layer that rearranges the blocks at the end of each round was chosen to maximise diffusion between blocks, according to a result of [15]. Compared to the cyclic shift, this permutation requires twice less rounds for an input difference to influence all the blocks. One round of TWINE is depicted on Fig. 6. The whole encryption process consists in 36 rounds of this scheme for both key lengths. The Feistel functions is used 8 times per

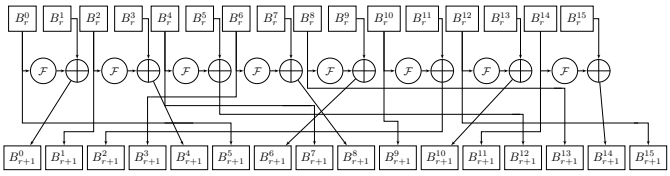

Fig. 6. One round of TWINE.

round and is consecutively made of:

- a 4-bit Xor with a subkey block.
- a single 4×4 S-box.

4) CLEFIA: CLEFIA is a 128-bit block cipher presented at FSE '07 [14] with key sizes 128, 192 or 256 bits. It is a GFN with 4 blocks, 32 bits each, as depicted on Fig. 7. At each round, it uses 2 slightly different Feistel functions $\mathcal{F}_0$ and $\mathcal{F}_1$. Function $\mathcal{F}_0$ is given on Fig. 8. They are both made of three consecutive steps:

- Xor with the subkey,
- 4 parallel 8×8 S-boxes. Both functions $\mathcal{F}_0$ et $\mathcal{F}_1$ uses the two same S-boxes S_0 and S_1 but in different orders.

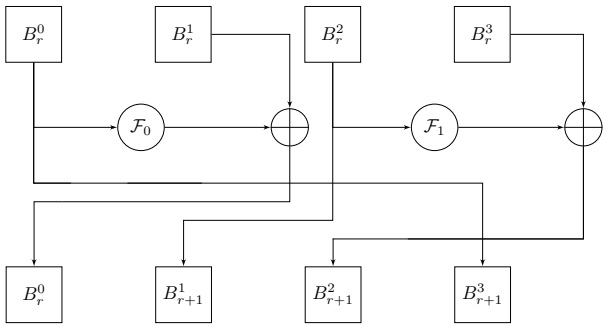

Fig. 7. Overview of CLEFIA.

- two linear diffusion layers, MC_0 for $\mathcal{F}_0$, MC_1 for $\mathcal{F}_1$ based on finite field operations (much like AES Mix-Columns).

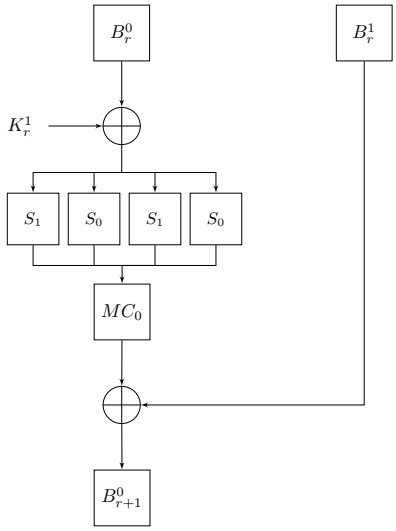

Fig. 8. The Feistel function $\mathcal{F}_0$ of CLEFIA.

C. Attacks using Fault Injections

1) Fault Injections: Fault injection attacks consist in disturbing the circuit behaviour in order to alter the correct processing of the algorithm. Unfortunately, in fault injection attacks the circuit can be damaged. Sakiyama *et al.* in [12] emphasize the importance of limiting the number of fault injections. So, our analysis is conducted with this goal in mind.

By design, algorithms are made secure against classical attacks. For example, in the case of the classical Feistel cipher, Luby and Rackoff demonstrated in [9] the following result: no efficient algorithm exists to distinguish a 4-round Feistel cipher with pseudo-random Feistel functions from a random permutation with only the observation of (plaintext, ciphertext) pairs.

Despite this security, an algorithm may be vulnerable to physical attacks. There are two families of physical cryptanalysis : SCA (Side Channels Analysis) attacks; and FI (Fault Injection) attacks. In this paper we will only focus on fault

injections attacks. One of the first fault attack was proposed against a Feistel cipher (the DES) by Biham and Shamir in [4] in order to recover the cipher-key.

In our approach, the fault attack changes one single-bit of a block B_r^i. The single-bit fault model is a common one which account for an attack where a low stress injection method is used.

2) DFA on a Generalized Feistel Network: The Differential Fault Analysis (DFA) is a powerful cryptanalytic technique that exploits differences between the correct ciphertext and erroneous results due to fault injections. The attack path for DFA on a Feistel cipher is classically done on the last round as in [4] for DES [18]. The attack path is shown on Fig. 9 and is repeated for each subkey block K_r^λ. From now on, variables with a superscript * (star) denote the faulted version of a data and the letter Δ is used to denote the difference between the correct value and the faulted one. E.g. $B^{i*}_{\mathbf{r}-1}$ denotes the faulted version of $B^i_{\mathbf{r}-1}$, and $\Delta_{B^i_{\mathbf{r}-1}} = B^{i*}_{\mathbf{r}-1} \oplus B^i_{\mathbf{r}-1}$. Thus, if $\mathcal{F}$ has

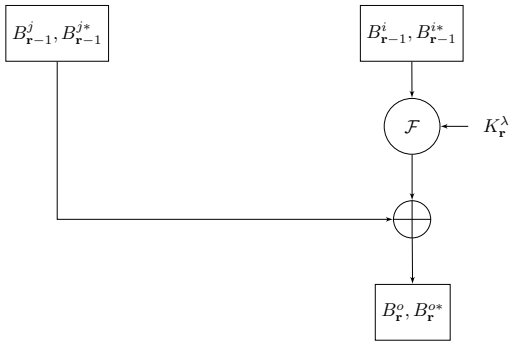

Fig. 9. Attack path of a fault injection on a Feistel cipher.

block $B^i_{\mathbf{r}-1}$ as input and has its output xored with $B^j_{\mathbf{r}-1}$ to produce $B^o_{\mathbf{r}}$, we have the following relation:

$$\begin{aligned} \Delta_{B^o_{\mathbf{r}}} &= B^o_{\mathbf{r}} \oplus B^{o*}_{\mathbf{r}} \\ &= \mathcal{F}\left(B^i_{\mathbf{r}-1}, K^\lambda_{\mathbf{r}}\right) \oplus \mathcal{F}\left(B^{i*}_{\mathbf{r}-1}, K^\lambda_{\mathbf{r}}\right) \oplus \Delta_{B^j_{\mathbf{r}-1}} \end{aligned} \quad (1)$$

However it is possible for the subkey to be xored at the end of the Feistel function. In this case, we have:

$$\begin{aligned} \Delta_{B^o_{\mathbf{r}}} &= \mathcal{F}\left(B^i_{\mathbf{r}-1}\right) \oplus K^\lambda_{\mathbf{r}} \oplus \mathcal{F}\left(B^{i*}_{\mathbf{r}-1}\right) \oplus K^\lambda_{\mathbf{r}} \oplus \Delta_{B^j_{\mathbf{r}-1}} \\ &= \mathcal{F}\left(B^i_{\mathbf{r}-1}\right) \oplus \mathcal{F}\left(B^{i*}_{\mathbf{r}-1}\right) \oplus \Delta_{B^j_{\mathbf{r}-1}} \end{aligned}$$

This equation does not depend on the subkey, so another attack path must be used instead. This case happens for example in the ciphers SIMON [1] or Piccolo [13]. Another case to deal with is when blocks $B^i_{\mathbf{r}-1}$ and $B^{i*}_{\mathbf{r}-1}$ are masked by a whitening key that is xored between the last round of the Feistel construction and the ciphertext as in Piccolo [13] for example. These two particular cases are not taken into account in this study but are not mandatory for fault injections on GFNs to work, see e.g. [7] for a successful attack on Piccolo.

Physical attacks are based on a divide-and-conquer strategy: the targeted subkey block $K^\lambda_{\mathbf{r}}$ is cut into L pieces. That is, the whole subkey is then cut into Λ blocks $K^\lambda_{\mathbf{r}}$, each block being cut into L pieces $K^{\lambda,l}_{\mathbf{r}}$. How this second division is done depends on the S-box layer: one piece of subkey per S-box. An hypothesis on $K^{\lambda,l}_{\mathbf{r}}$ is denoted k and $\hat{k}$ denotes the correct value.

III. Description of the generic approach

In this section a generic method is proposed to assess the vulnerability of a particular block in a GFN. In particular the most vulnerable block B_r^e, the best suited for injecting single-bit faults, is searched (e is the block index and r is the round number). A block is the most vulnerable when injecting single-bit faults at this location minimizes the total number of faults required to find the key. We wish to have a good diffusion of the fault as to attack as many subkey bits as possible for each fault injection. *A priori*, this depends on two things: the overall structure of the Feistel network but also the Feistel function used.

A. At the Feistel network level

At the Feistel network level, the diffusion of faults in the blocks are studied which allows to predict the propagation of faults in the blocks.

1) Full diffusion delay: We define here the notion of (block-wise) diffusion in a generalized Feistel network, as given in [2]. A block B_0^i **influences** block B_r^j if B_0^i effectively appears when expressing B_r^j as a function of $B_0^0, \cdots, B_0^{\mathbf{b}-1}$. A block B_0^i **has diffused** at round r if B_0^i influences all the B_r^j for $j \in [\![0, \mathbf{b}-1]\!]$. If all the blocks B_0^i have diffused at round r, the Feistel network is said to have reach **full diffusion**, i.e. each output block B_r^j depends on all input blocks B_0^i. The minimum number of rounds required to reach full diffusion is called full diffusion delay, and denoted d. By definition of the full diffusion delay, it is unnecessary to inject the fault before the d latest rounds as doing so will not further propagate the fault but may complexify the analysis. We then focus on the blocks B_r^e with $e \in [\![0, \mathbf{b}-1]\!]$, $r \in [\![\mathbf{r}-(d+1), \mathbf{r}-1]\!]$ where e indicates the block index where the single-bit fault has been injected.

2) Matrix Representation of Feistel Networks: At SAC '13, Berger *et al.* [2] presented a general framework to represent generalized Feistel networks using matrices. The matrix representation of a GFN is the adjacency matrix of the graph representing influence relations in the GFN: when block B_{r+1}^i is influenced by B_r^j, a non zero coefficient at position (i, j) in the matrix is written. The non zero coefficients of this matrix can either be 1 or $\mathcal{F}$ depending on whether the influence is direct or through a Feistel function $\mathcal{F}$. They then compute powers of that (adjacency) matrix to determine the full diffusion delay. Technically, the computation is done over the coefficient ring of polynomials $\mathbb{Z}[\mathcal{F}]$. In this paper, we reuse this representation but in a slightly improved way.

Let D be $\mathbb{N} \cup \{-\infty\}$ endowed with the two internal laws $\max$ and $+$, defined over the integers $\mathbb{N}$ and extended to D in the obvious way: $\forall a \in D,\ \max(a, -\infty) = \max(-\infty, a) = a$ and $(-\infty) + a = a + (-\infty) = -\infty$. Then D is an idempotent commutative semiring, also known as commutative dioid. A semiring is an algebraic structure very similar to a ring except

that the first law ($\max$ in our case) forms only a commutative monoid instead of a commutative group, i.e. the requirement to have a first-law inverse for every element is discarded. A semiring is said idempotent when $\forall a \in D, \max(a,a) = a$, and commutative when the second law ($+$ in our case) is commutative. A dioid is simply an idempotent semiring.

This object represents degrees of polynomials with the convention that the degree of the zero polynomial is $-\infty$. In our case, a polynomial in the Feistel function $\mathcal{F}$ is studied. We then define the dioidic matrix representation $\mathcal{M}$ of a GFN to be the matrix with coefficients in D such that coefficient at position (i,j) can either be:

- 0, when block B^i_{r+1} is influenced by B^j_r directly.
- 1, when block B^i_{r+1} is influenced by B^j_r via the Feistel function $\mathcal{F}$.
- $-\infty$, otherwise, i.e. not influenced.

For example, the matrix $\mathcal{M}$ of the GFN on Fig. 2 is given on Fig. 10.

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty \\ -\infty & -\infty & 0 & -\infty & -\infty & -\infty & -\infty & -\infty \\ -\infty & -\infty & 1 & 0 & -\infty & -\infty & -\infty & -\infty \\ -\infty & -\infty & -\infty & -\infty & 0 & -\infty & -\infty & -\infty \\ -\infty & -\infty & -\infty & -\infty & 1 & 0 & -\infty & -\infty \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & 0 & -\infty \\ -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & 1 & 0 \\ 0 & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty \end{pmatrix}$$

Fig. 10. Matrix of the GFN given on Fig. 2.

The link with the matrix representation introduced in [2] is as follows. The coefficients in our matrix represent the degree of the coefficient at the same position in the matrix of [2]. That is, we are interested in how many Feistel functions are passed through to go from an input block to an output block, which is represented by the degrees in the polynomials in the Feistel function $\mathcal{F}$.

3) Using the Matrix Representation for Fault Injections: For a subkey block $K^\lambda_{\mathbf{r}}$ to be attacked using fault injections, the block $B^i_{\mathbf{r}-1}$ at the input of the λ-th Feistel function $\mathcal{F}$ must be faulted. Thus the block where the fault is done must influence that block $B^i_{\mathbf{r}-1}$. We recall that due to the structure of a (generalized) Feistel network, the input of a Feistel function at round $\mathbf{r}-1$ **must** be outputted as a ciphertext block to have an invertible construction.

Let B^e_r be the block where the fault is injected and V the e-th standard basis vector (which has then $-\infty$ coefficients on every coordinates except the e-th one which is 0). The vector $V_{\mathcal{F}}$ defined by the vector matrix product given on equation (2) allows to know for each block at the output of the penultimate round whether the fault injected in B^e_r has influenced that particular block and how many Feistel functions $\mathcal{F}$ has been crossed doing so.

$$V_{\mathcal{F}} = \mathcal{M}^{\mathbf{r}-(r+1)} \cdot V \tag{2}$$

If at the index i, corresponding to block $B^i_{\mathbf{r}-1}$, we observe:

- $-\infty$: the block is not faulted.
- 0 : the block contains the single-bit fault.
- $x \in \mathbb{N}$: the block is faulted and the fault has been through x Feistel functions.

Hence for each fault, the number n_λ of subkeys blocks attacked after r rounds can be computed.

B. At the Feistel function level

Recall that a subkey block is split into L pieces (one per S-box). In the Feistel function, a non-linear layer is necessary. This layer is composed of one or several (L) S-boxes. This division has an impact on the propagation of the fault inside the Feistel function. Since an S-box must be faulted for the attacker to be able to gain knowledge on the corresponding subkey piece, knowing into which S-boxes the fault is propagated is of the utmost importance.

Studying the Feistel function allows to identify how many pieces of the subkey blocks are attacked by each fault injection. This number is denoted n_l.

One should also be able to find all the possible differentials $\Delta_{B^j_{\mathbf{r}-1}}$. Indeed, as $B^j_{\mathbf{r}-1}$ might be faulted, knowing the different values of $\Delta_{B^j_{\mathbf{r}-1}}$ is necessary in order to be able to use equation (1) to retrieve the key. The rest of this section deals with the diffusion properties of the Feistel function.

1) Number of pieces of subkey attacked: The first thing to determine from the Feistel function is the number of pieces of the subkey which are attacked, *i.e.* the number of faulted S-boxes. Looking at the Feistel function, it is possible to know what output bits can be faulted with a single-bit fault at the input. And, since we know what S-boxes depend on which input bits of the Feistel function, it is possible to deduce which S-boxes will be faulted in the next Feistel function.

So from the diffusion of the bits in the Feistel function it is possible to compute the number n_l of pieces of the subkey block attacked by a fault injection, depending on the number of Feistel functions crossed which is $V_{\mathcal{F}}(i)$, the i-coordinate of vector $V_{\mathcal{F}}$. Care should be taken not to forget the case where the fault has not been through any function. For example in DES, a single-bit fault can impact up to 2 S-boxes because of Expansion E.

2) Search for the values of $\Delta_{B^j_{\mathbf{r}-1}}$: If the fault is not propagated to $B^j_{\mathbf{r}-1}$, the value of $\Delta_{B^j_{\mathbf{r}-1}}$ in equation (1) is equal to 0. In the other case, the value of $\Delta_{B^j_{\mathbf{r}-1}}$ is unknown.

$V_{\mathcal{F}}(j)$, the j-th coordinate of vector $V_{\mathcal{F}}$, indicates if $B^j_{\mathbf{r}-1}$ has been faulted and how many times the fault passed through the Feistel function.

Simply put, the study of the diffusion of bits in the Feistel function allows to evaluate how many bits are potentially faulted in $B^j_{\mathbf{r}-1}$ when the number of times the injected fault crossed the Feistel function is known. For example, if the fault do not cross any Feistel function, the fault being single-bit, there are as many possible differentials as bits in $B^j_{\mathbf{r}-1}$. As a consequence, it is possible to evaluate the information brought by the fault since we can evaluate the incertitude on $\Delta_{B^j_{\mathbf{r}-1}}$.

But this approach can be improved by studying the actual values of the possible differentials. For that purpose, the single faulted bit in B_r^e is searched. Several hypotheses are often possible.

In order to find it the diffusion result on the next round is used, *i.e.* the value:

$$W_{\mathcal{F}} = \mathcal{M} \cdot V_{\mathcal{F}}$$

The minimal but finite value (*i.e.* in $\mathbb{N}$) in the vector $W_{\mathcal{F}}$ corresponds to the faulted block where the fault crossed the Feistel function a minimum number of times.

If this value is equal to 0, the single-bit fault can be observed in the ciphertext on the corresponding block.

If the value is equal to 1, the differential of the corresponding block $\Delta_{B_{\mathbf{r}}^{\min(W_{\mathcal{F}})}}$ is computed. For all single-bit faults, it is often possible to compute exhaustively all possible differentials at the output of a Feistel function $\mathcal{F}$. As a consequence, knowing $\Delta_{B_{\mathbf{r}}^{\min(W_{\mathcal{F}})}}$ restricts the set of possible single-bit faults that have been injected.

If the minimal value in $W_{\mathcal{F}}$ is greater or equal to 2, the above method might be still possible. However, since for each intermediate difference tuple (i.e difference after the first, second,..., penultimate function), all the output differences (after the last function) must be computed, the complexity of such an algorithm grows exponentially fast in the size of one block and in the value of $\min(W_{\mathcal{F}})$ and quickly becomes infeasible. Besides, due to Feistel function being designed for maximazing diffusion inside a block, it is likely that this will not help restricting the possible single-bit faults injected any longer.

If $\Delta_{B_{\mathbf{r}}^{\min(W_{\mathcal{F}})}}$ allows to restrict the set of fault candidates, each one of these candidates is propagated to obtain a reduced set of potential differentials $\Delta_{B_{\mathbf{r-1}}^j}$. The union of all possible differentials (for all fault candidates) indicates the number of possible differentials $\Delta_{B_{\mathbf{r-1}}^j}$. Yet this approach is not perfect since two differentials can be xored in the datapath. This problem has no consequences for us since we only try to evaluate the number of faults required to find the key and we do not construct a fault attack. But a cryptanalyst would have to take it into account to conceive a fault attack.

C. At the S-box level

In this section the information on the key brought by a fault crossing an S-box is evaluated. The value n, the number of faults in an S-box to find the corresponding subkey piece, is computed. The attack path is reduced to the S-box level, considering that what happens in the Feistel function $\mathcal{F}$ before the Xor with the subkey is known and that the S-boxes are the only non-linear part of the function. This case is summed up by equation (3), where x the input of the S-box, x^* the faulted input and Δ_y the output differential are known.

$$S(x \oplus \hat{k}) \oplus S(x^* \oplus \hat{k}) = \Delta_y \tag{3}$$

A method inspired by classical differential cryptanalysis is used. Differential cryptanalysis is a chosen plaintext attack introduced by Biham and Shamir in 1991 [3]. It consists in considering plaintexts pairs having a fixed difference and in studying how this difference propagates throughout encryption.

If we write $e = x \oplus x^*$ in equation (3), we have:

$$S(x \oplus \hat{k}) \oplus S(x \oplus e \oplus \hat{k}) = \Delta_y \tag{4}$$

Each time another fault is injected, values of x, e and Δ_y change while $\hat{k}$ stays the same.

We are then interested in the minimal number n of experiences (fault injections) such that being given $x_0, \cdots, x_{n-1}$, $e_0, \cdots, e_{n-1}$ and $\Delta_{y_0}, \cdots, \Delta_{y_{n-1}}$, there is only one key for which the n equations (4) simultaneously hold. The S-box S is supposed to be a vectorial boolean function from $\mathbb{F}_2^u$ into $\mathbb{F}_2^v$. Let $a \in \mathbb{F}_2^u$ and $b \in \mathbb{F}_2^v$, $\mathcal{S}_{a,b}$ denotes the set of all inputs z such that if an input difference a is applied, an output difference b is observed. Said otherwise:

$$\mathcal{S}_{a,b} = \{z \in \mathbb{F}_2^u \mid S(z) \oplus S(z \oplus a) = b\}.$$

By symmetry of this equation, $\#\mathcal{S}_{a,b}$ is even. Let δ denotes the base-2 logarithm of the maximum size of the sets $\mathcal{S}_{a,b}$ taken over all $a \neq 0$ and b, that is:

$$2^\delta = \max_{a \neq 0,\, b} \#\mathcal{S}_{a,b}$$

Since the $\mathcal{S}_{a,b}$'s are of even size and not all empty, it follows that $\delta \geq 1$. More generally, $\delta \geq u - v$. Indeed, let a be a non-zero fixed input difference, then $\sum_{b \in \mathbb{F}_2^v} \#\mathcal{S}_{a,b} = 2^u$. Hence there exists a b so that $\#\mathcal{S}_{a,b} \geq 2^u/2^v$ and thus $\delta \geq u - v$. Finally, if x is drawn uniformely at random in $\mathbb{F}_2^u$ then

$$\Pr\left[x \in \mathcal{S}_{a,b}\right] = \frac{\#\mathcal{S}_{a,b}}{2^u} \leq 2^{\delta - u}.$$

A key k is then a solution of equation (4) if and only if $x \oplus k \in \mathcal{S}_{e,\Delta_y}$. So the number of solutions of equation (4) is $\#\mathcal{S}_{e,\Delta_y}$. The set of z such that $z \oplus x \in \mathcal{S}_{e,\Delta y}$ is denoted by $x \oplus \mathcal{S}_{e,\Delta_y}$. Then k is solution of equation (4) if and only if $k \in x \oplus \mathcal{S}_{e,\Delta_y}$. For n fault injections, we are then interested in the probability

$$\Pr\left[k \in \bigcap_{s=0}^{n-1} (x_s \oplus \mathcal{S}_{e_s,\Delta_{y_s}})\right].$$

The faults e_s are independent from each others. The Δ_{y_s} and x_s are independent and identically distributed. That is, they are random variables with the same probability law and are mutually independent. Thus we can write:

$$\Pr\left[k \in \bigcap_{s=0}^{n-1} \left(x_s \oplus \mathcal{S}_{e_s,\Delta_{y_s}}\right)\right] = \prod_{s=0}^{n-1} \Pr\left[k \in (x_s \oplus \mathcal{S}_{e_s,\Delta_{y_s}})\right] \leq 2^{n(\delta - u)}$$

Besides, this probability is lower bounded by 2^{-u} since there exists a solution. Hence the number of faults n searched,

the number of faults on the S-box required to find the corresponding subkey piece, is such that $2^{n(\delta-u)} \leq 2^{-u}$, which leads to

$$n \geq \left\lceil \frac{u}{u-\delta} \right\rceil \tag{5}$$

In practice, there are mainly two cases possible:

- $u \times u$ bijective S-boxes, with δ as small as possible to best resist differential attacks and essentially $u = 4$ or $u = 8$. We know that $\delta \geq 1$. However, at present time, the only known example of S-boxes that reach this bound are for odd u or $u = 6$. Still, there are countless examples of S-boxes with $\delta = 2$. The case $u = 4$ is very well studied, see e.g. the classifications of 4×4 S-boxes of Leander and Poschamann [8] or of Saarinen [11] and it is now very easy to find a 4×4 S-box with minimum δ. The other standard case is $u = 8$. If there is no known global classifications of 8×8 S-boxes at present day, there are many examples in the literature, the most famous being the S-box of the AES. In both cases ($u = 4$ and $u = 8$), $n = 2$ is obtained.
- 6×4 S-boxes as in DES. In general, a 6×4 S-box verifies $\delta \geq 6 - 4 = 2$ and hence $n = 2$. However, because of their structure the DES S-boxes do not reach this bound but verify $\delta = 4$ instead, hence $n = 3$ for DES.

Anyway, since the value of δ is also required to assess the security of the cipher against differential cryptanalysis, it is, by design of the cipher, always possible to compute it in reasonnable time.

Now what happens when there is incertitude on the Δ_y value is studied. We have the following equation:

$$S(x \oplus \hat{k}) \oplus S(x \oplus e \oplus \hat{k}) = \Delta_{y_1} \text{ or } \cdots \text{ or } \Delta_{y_J}, \tag{6}$$

where J denotes the number of possible Δ_y. Since for a fixed input difference a, the sets $\mathcal{S}_{a,b_t}$ are pair-wise disjoints, we have :

$$\Pr\left[z \in \bigcup_{t=1}^{J} \mathcal{S}_{a,b_t}\right] = \sum_{t=1}^{J} \Pr\left[z \in \mathcal{S}_{a,b_t}\right]$$

As in the case where there is only one Δ_y, k is solution of equation (6) if and only if

$$k \in x \oplus \bigcup_{t=1}^{J} \mathcal{S}_{e,\Delta_{y_t}}$$

Notice that the union of subkey hypotheses $\bigcup_t (x \oplus \mathcal{S}_{e,\Delta y_t})$ is interesting only when it is different from the set of all the hypotheses.

If one does this experiment n times, we have:

$$\Pr\left[k \in \bigcap_{s=0}^{n-1} \left(x_s \oplus \bigcup_{t=1}^{J} \mathcal{S}_{e_s,\Delta_{y_{s,t}}}\right)\right] = \prod_{s=1}^{n-1} \Pr\left[k \in x_s \oplus \bigcup_{t=1}^{J} \mathcal{S}_{e_s,\Delta_{y_{s,t}}}\right] \tag{7}$$

Therefore the minimum number of faults n_J, where J is the incertitude (the number of possible Δ_y), to retrieve the key is the smallest n such that the term on equation (7) becomes smaller than 2^{-u}, *i.e.* small enough to eliminate all the keys but the right one.

IV. Algorithm and Results

A. Algorithm to find vulnerable locations for a fault injection

In this section, a method is given in order to find good locations to inject a single-bit fault into a generalized Feistel network. Algorithm 1 describes this procedure. Its inputs are the Feistel function $\mathcal{F}$ and the GFN matrix $\mathcal{M}$. For every block B_r^e with $e \in [\![0, \mathbf{b}-1]\!]$ and $r \in [\![\mathbf{r}-(d+1), \mathbf{r}-1]\!]$ where the fault can be injected, the algorithm returns the number of subkey blocks and, for each of them, the number of pieces of subkey and the number of differentials to take into account.

Using the Feistel network, for each block candidate for a fault injection, the number n_λ of subkey blocks attacked is first computed. Then for each of the subkey blocks $K_{\mathbf{r}}^{\lambda}$, the algorithm computes the number of pieces attacked (worst case and best case). Then if $V_{\mathcal{F}}(j) \neq -\infty$, i.e. if the block xored at the Feistel function output is faulted, giving a first estimation of the number of possibles differentials. The algorithm reuses then the Feistel network to find the least influenced ciphertext block (but influenced still) by the fault, to further reduce the number of possible differentials. If that number does not allow to eliminate some subkey candidates, we consider that the subkey piece $K_{\mathbf{r}}^{\lambda}$ cannot be attacked using this fault. Finally, we compute the number n_J of faults required to retrieve the remaining candidates from the average number of Δ_y involved.

This algorithm gives a first insight on where to inject the fault. Yet its genericity does not rule out that a better fault attack exists on a particular GFN. Moreover, it is quite possible to imagine an attack that injects faults in more than one block or with a different fault model.

B. Results on some Feistel ciphers

We have tested our approach on several Feistel ciphers. The exposed results focus essentially on three parameters. First, the number n_λ of subkey blocks attacked and for each of them the number n_l of pieces attacked, which together are a measure of the fault efficiency: the higher these numbers, the better the fault. And second the number of differences $\Delta_{B_{\mathbf{r}-1}^j}$ on the output block (for each of the n_λ blocks attacked) which is a measure of the fault inefficiency: the more $\Delta_{B_{\mathbf{r}-1}^j}$ there are, the more faults will be required to find the subkey, the worst case being that the subkey cannot be found. A good attacker will then have to find where to inject the fault to balance those parameters that tend to grow together but have opposite results.

In this section, the matrices have coefficients in the dioid D, assuming matrix vector product. The value $-\infty$ are replaced by '.' (dot) for readability.

1) DES: Our algorithm is tested on DES since it was the most well known 2-block Feistel cipher.

In the following, details of our analysis are given:

- Number of rounds: $\mathbf{r} = 16$.

Algorithm 1 Finding the block where the least fault injections are required to attack the cipher.

Require: the Feistel function $\mathcal{F}$, the GFN matrix $\mathcal{M}$
Ensure: all the B_r^e ($e \in [\![0, \mathbf{b}-1]\!]$ and $r \in [\![\mathbf{r}-(d+1), \mathbf{r}-1]\!]$) with the corresponding n_λ, n_l, J and n_J values.

Compute the full diffusion delay d using $\mathcal{M}$.
Deduce the number of S-boxes L from the Feistel function $\mathcal{F}$.
for 0 to d, number of times the fault crossed $\mathcal{F}$ **do**
  Compute the associated number of pieces attacked n_l
  Compute the number J of output differences possible and the number of faults n_J required to retrieve a subkey piece
end for
for $\forall B_r^e$ with $e \in [\![0, \mathbf{b}-1]\!], r \in [\![\mathbf{r}-(d+1), \mathbf{r}-1]\!]$ **do**
  Create the e-th basis vector V
  $V_{\mathcal{F}} \leftarrow \mathcal{M}^{\mathbf{r}-(r+1)} \cdot V$
  $W_{\mathcal{F}} \leftarrow \mathcal{M} \cdot V_{\mathcal{F}}$
  Deduce the number of blocks that can be attacked n_λ
  for All the subkey blocks $K_{\mathbf{r}}^\lambda$ attacked **do**
    Identify the input block $B_{\mathbf{r}-1}^i$ and the block xored at the output of the Feistel function $B_{\mathbf{r}-1}^j$
    Find the corresponding n_l from $V_{\mathcal{F}}(i)$
    if $V_{\mathcal{F}}(j) \neq -\infty$ **then**
      if $V_{\mathcal{F}}(j) \geq 2$ **then**
        Remove $K_{\mathbf{r}}^\lambda$ from the list of blocks that can be attacked
      else
        Compute the number of possible $\Delta_{B_{\mathbf{r}-1}^j}$
        if the smallest coefficient $\neq -\infty$ of $W_{\mathcal{F}}$ is less or equal to 1 **then**
          Reduce the number of possible $\Delta_{B_{\mathbf{r}-1}^j}$
        end if
        Deduce the corresponding J
        if $J \geq \#\{k\}$ **then**
          Remove $K_{\mathbf{r}}^\lambda$ from the list of blocks that can be attacked
        end if
      end if
    end if
    Compute the number n_J of faults required to attack that subkey block
  end for
end for

- Number de blocks: $\mathbf{b} = 2$.
- Full diffusion delay: $d = 2$.
- The GFN $\mathcal{M}$:

$$\mathcal{M} = \begin{pmatrix} . & 0 \\ 0 & 1 \end{pmatrix}.$$

- Number of subkey blocks: $\Lambda = 1$. One has to remark that $B_{\mathbf{r}-1}^i = R_{15}$ and $\Delta_{B_{\mathbf{r}-1}^j} = \Delta_{L_{15}}$.
- Number of pieces in subkey blocks (number of S-boxes): $L = 8$.
- Number of faults required to retrieve a piece of subkey using equations of the form (3): $n = 3$.

Then the Feistel function is studied :

- Because of Expansion E, a single bit input difference can impact 2 S-boxes.
- A property of DES S-boxes is that a single bit difference at the input of an S-box ensures at least a two-bit difference at the output.
- the bit-permutation P is made such that the 4 output bits of an S-box are sent to 4 different S-boxes at the input of the next round.

Thus even when the fault does not go through a Feistel function, up to 2 S-boxes can still be attacked. If the fault goes through exactly one Feistel function, there are between 2 and 8 faulted output bits. After passing through the Feistel function, 2 to 8 bits are faulted at the output, leading to $32*(256-1-8)$ possible differentials.

Conversely, assuming a single-bit input differential, observing the output of the Feistel function is characteristic enough to reduce the number possible input differentials from 32 to only 2.

In the case of the DES, the fault injection on L_{15} is eliminated since it does not allow to attack the key. Results of our analysis are summed up in Table I where n_l is the number of pieces of subkey blocks attacked and Δ is the number of guesses on $\Delta_{L_{15}}$. The number n_λ of subkey blocks attacked is of course $n_\lambda = 1$ since there is only one Feistel function per round.

TABLE I
RESULTS OF OUR ANALYSIS ON THE DES

Blocks B	$V_{\mathcal{F}}$	$W_{\mathcal{F}}$	n_l	Δ
R_{15} or L_{14}	$(.,0)$	$(1,0)$	$1 \leq n_l \leq 2$	1
R_{14} or L_{13}	$(0,1)$	$(2,1)$	$2 \leq n_l \leq 8$	2
R_{13}	$(1,2)$	$(3,2)$	$2 \leq n_l \leq 8$	$32 * 247$

Our method shows that it is more judicious to inject faults in R_{14} for the DES. Yet Biham and Shamir have shown in [4] that, for the DES, a fault in R_{13} leads to a more efficient attack. To find this result, they have studied the round function in depth, while our approach is generic.

Moreover, the study by Rivain in [10] showed that injecting faults before, in rounds $9, 10, 11, 12$ (beyond the full diffusion delay), requires a bigger number of faults to find the key.

2) MIBS: This cipher is chosen because, as DES, it is a classical Feistel network and thus it is mainly the Feistel function that matters for our study. Moreover, contrarily to DES, the S-boxes are bijective, which is more common among block ciphers.

In the following, details of our analysis are given:

- Number of rounds: $\mathbf{r} = 32$.
- Number of blocks: $\mathbf{b} = 2$.
- Full diffusion delay: $d = 2$,
- The GFN $\mathcal{M}$:

$$\mathcal{M} = \begin{pmatrix} 1 & 0 \\ 0 & . \end{pmatrix}.$$

- Number of subkey blocks: $\Lambda = 1$. One has to remark that $B^i_{\mathbf{r}-1} = L_{31}$ and $\Delta_{B^j_{\mathbf{r}-1}} = \Delta_{R_{31}}$.
- Number of pieces in subkey blocks (number of S-boxes): $L = 8$.
- Number of faults required to retrieve a piece of subkey using equations of the form (3): $n = 2$.

The diffusion function following the S-boxes strictly preserves the partition into 4-bit blocks defined by the S-boxes. Hence the study of MIBS Feistel function is as follows:

- Each input bit faults exactly one S-box. S-boxes are bijective, so modifying one bit in input means modifying at least one bit in output, so 1 to 4 output bits can be faulted.
- If the fault has been through exactly one Feistel function, there are 5 or 6 pieces of subkey blocks that can be simultaneously attacked.
- If one observe a faulted output (assuming single-bit faulted input), one can deduce which S-box is faulted and thus the input fault is reduces to 4 possibilities.
- If the fault has been through more than one Feistel function, then all the S-boxes are faulted.

Results of our analysis are summed up in Table II where n_l is the number of pieces of subkey blocks attacked and Δ is the number of guesses on $\Delta_{R_{31}}$. As for DES, the number n_λ of subkey blocks attacked is $n_\lambda = 1$.

TABLE II
RESULTS OF OUR ANALYSIS ON MIBS

Blocks B	$V_\mathcal{F}$	$W_\mathcal{F}$	n_l	Δ
L_{31} or R_{30}	$(0,.)$	$(0,1)$	1	1
L_{30} or R_{29}	$(1,0)$	$(1,2)$	$5 \leq n_l \leq 6$	4
L_{29}	$(2,1)$	$(2,3)$	8	112

Results indicate that it seems best to attack on block L_{30}. This way though, not all the subkeys pieces can be attacked at the same time. On the contrary, injecting on L_{29} allows to attack the whole subkey, but the number of differences on $\Delta_{R_{31}}$ becomes greater. A more in-depth study of the L_{29} case may be possible, but may not be trivial to perform.

3) TWINE: TWINE was chosen because it is the exact opposite of DES and MIBS: the Feistel function is very simple, it is a single S-box. The whole study will be on the Feistel networks itself.

In the following, details of our analysis are given:

- Number of rounds: $\mathbf{r} = 36$.
- Number de blocks: $\mathbf{b} = 16$.
- Full diffusion delay: $d = 8$.
- The GFN $\mathcal{M}$:

$$\begin{pmatrix}
1 & 0 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & 0 & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & 1 & 0 & . & . & . & . \\
. & . & . & . & . & . & 0 & . & . & . & . & . & . & . & . & . \\
. & . & 1 & 0 & . & . & . & . & . & . & . & . & . & . & . & . \\
0 & . & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & 1 & 0 & . & . & . & . & . & . \\
. & . & . & . & 0 & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & 1 & 0 & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & 0 & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & 1 & 0 & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 0 & . \\
. & . & . & . & 1 & 0 & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & 0 & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 & 0 \\
. & . & . & . & . & . & . & . & . & . & . & . & 0 & . & . & .
\end{pmatrix}.$$

- Number of subkey blocks: $\Lambda = 8$.
- Number of pieces in subkey blocks (number of S-boxes by Feistel function): $L = 1$.
- Number of faults required to retrieve a piece of subkey using equations of the form (3): $n = 2$.

Since the Feistel function is essentially a single S-box (i.e. $n_l = 1$), if the fault on block $B^j_{\mathbf{r}-1}$, that is xored to the ouput of the targetted function, has been through exactly one Feistel function in the previous rounds, then this fault can take $\Delta = 14$ of the $2^4 - 1 = 15$ possible values, restricted to $\Delta = 7$ if the input fault is known. Hence, there are three favorable cases:

- $B^j_{\mathbf{r}-1}$ is not faulted: $\Delta = 1$.
- $B^j_{\mathbf{r}-1}$ contains exactly the value of the fault: $\Delta = 1$ if the value of the fault is known, $\Delta = 4$ else.
- $B^j_{\mathbf{r}-1}$ has been through exactly one function: $\Delta = 7$ if the value of the fault is known, $\Delta = 14$ else.

The best place for a fault injection on TWINE will then be the register where the fault reaches the most Feistel functions with the condition that their respective outputs must fall in the above three cases. It appears that the best case achievable is to inject a fault at round 31, on any B^i_{31} at the input of a function. This way it is possible to attack simultaneously $n_\lambda = 5$ functions on the last round of TWINE, four of them with a non faulted $B^j_{\mathbf{r}-1}$, i.e. with $\Delta = 1$, and a fifth one with $\Delta = 7$. Injecting the fault earlier in the algorithm (round 30 and before) only allows to attack at most 4 functions, which is not as interesting as the previous case. Finally, if the fault is injected at round 32 or after, then only up to 3 functions can be attacked.

4) CLEFIA: CLEFIA was chosen as an intermediate case between DES and TWINE: it has more than two blocks and

the Feistel function is more than a single S-box. Thus, both the Feistel function and the Feistel network have to be studied and then mixed together.

In the following, details of our analysis are given:

- Number of rounds $\mathbf{r} = 18$, 22 or 26 depending on the key size (resp. 128, 192 or 256 bits).
- Number of blocks $\mathbf{b} = 4$.
- Full diffusion delay $d = 4$.
- The GFN $\mathcal{M}$:

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & . & . \\ . & . & 0 & . \\ . & . & 1 & 0 \\ 0 & . & . & . \end{pmatrix}.$$

- Number of subkey blocks: $\Lambda = 2$.
- Number of pieces in subkey blocks (number of S-boxes by Feistel function): $L = 4$.
- Number of faults required to retrieve a piece of subkey using equations of the form (3): $n = 2$.

The study of the CLEFIA Feistel function is as follows:

- Each input bit faults exactly one S-box. S-boxes are bijective, so modifying one bit in input means modifying at least one bit in output, so 1 to 4 output bits can be faulted.
- Because of the diffusion properties of matrices MC_0 and MC_1 of CLEFIA, if the fault has been through one Feistel function, then all the S-boxes are faulted.
- If one observe a faulted output (assuming single-bit faulted input), one can deduce which S-box is faulted and thus the input fault is reduced to 4 possibilities.
- By symmetry in the network, one can restrict the analysis to the case where the fault is injected on block B_r^0.

Results of our analysis are summed up in Table III, assuming a 128-bit key, i.e. there are 18 rounds. For each possible location of the fault, the number $n_\lambda \in [\![1, 2]\!]$ of subkey blocks attacked, as well as for each of the two functions, the number $n_l \in [\![0, 4]\!]^2$ of pieces of subkey attacked and the corresponding number Δ of differences on the output block are given.

TABLE III
RESULTS OF OUR ANALYSIS ON CLEFIA

Blocks B	$V_{\mathcal{F}}$	$W_{\mathcal{F}}$	n_λ	n_l	Δ
B_{17}^0	$(0, ., ., .)$	$(0, 1, ., .)$	1	$(1, 0)$	$(1, -)$
B_{16}^0	$(1, ., ., 0)$	$(1, 2, ., 0)$	1	$(4, 0)$	$(1, -)$
B_{15}^0	$(2, ., 0, 1)$	$(2, 3, 0, 1)$	2	$(4, 1)$	$(1, \leq 127)$
B_{14}^0	$(3, 0, 1, 2)$	$(3, 4, 1, 2)$	2	$(4, 4)$	$(4, \text{huge})$
B_{13}^0	$(4, 1, 2, 3)$	$(4, 5, 2, 3)$	2	$(4, 4)$	$(946, \text{huge})$

Results indicate that injecting on B_{17}^0 is not optimal since if the injection is made one round earlier on B_{16}^0, it is possible for the same cost, to fault the whole function $\mathcal{F}_0$ instead of a single S-box. The attack by Chen *et al.* [5] with 18 faults (for CLEFIA-128) falls in that case.

If one wishes to fault both functions, then the second one will have a non trivial Δ. The case B_{15}^0 is still manageable, but only one S-box of the second function is attacked. Fault injections on B_{15} were used by Takahashi and Fukunaga [17] to successfully attack CLEFIA-128 requiring only 2 faults. Even if they make an in-depth study of CLEFIA Feistel functions to reduce the number of faults to its minimum, this result corroborates our analysis: it is best to inject the fault on B_{15}.

It might be possible to manage the case where the fault is done on B_{14}^0. This would allow to attack all 8 S-boxes at once but it would require very deep study of the Feistel functions to tackle the problem of Δ on the second function.

V. CONCLUSION

In this paper, it has been shown that some blocks are more vulnerable than others in generalized Feistel networks. Fewer faults are needed to find the key in the most vulnerable blocks, which make them targets of choice for fault attacks. A method has been proposed to identify these blocks allowing designers to deal with them accordingly. The genericness of the proposed method implies that the vulnerability evaluation is not optimal but a method to assess the vulnerabilities automatically has been proposed.

Further work will include finding a refined version of this study that can go deeper into the analysis of the Feistel functions while maintaining genericity.

Acknowledgements. The authors would like to thank Ronan Lashermes and Thierry P. Berger 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; and partially supported by the French National Agency of Research: ANR-11-INS-011.

REFERENCES

[1] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. *IACR Cryptology ePrint Archive*, page 404, 2013.

[2] T.P. Berger, M. Minier, and G. Thomas. Extended Generalized Feistel Networks using Matrix Representation. In *Selected Areas in Cryptography - SAC 2013*, volume 8282 of *LNCS*. Springer, 2013.

[3] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72, 1991.

[4] E. Biham and A. Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *CRYPTO*, pages 513–525, 1997.

[5] H. Chen, W. Wu, and D. Feng. Differential fault analysis on clefia. In *Information and communications security*, pages 284–295. Springer, 2007.

[6] M. Izadi, B. Sadeghiyan, S. S. Sadeghian, and H. A. Khanooki. MIBS: A New Lightweight Block Cipher. In *CANS*, volume 5888 of *LNCS*, pages 334–348. Springer, 2009.

[7] K. Jeong. Differential Fault Analysis on Block Cipher Piccolo. *IACR Cryptology ePrint Archive*, 2012:399, 2012.

[8] G. Leander and A. Poschmann. On the Classification of 4 Bit S-Boxes. In *WAIFI*, volume 4547 of *LNCS*, pages 159–176. Springer, 2007.

[9] M. Luby and C. Rackoff. How to Construct Pseudo-Random Permutations from Pseudo-Random Functions. In *Advances in Cryptology–CRYPTO*, volume 85, page 447, 1986.

[10] M. Rivain. Differential Fault Analysis on DES Middle Rounds. In *CHES*, pages 457–469, 2009.

[11] M.-J. O. Saarinen. Cryptographic Analysis of All 4 x 4 - Bit S-Boxes. Cryptology ePrint Archive, Report 2011/218.

[12] K. Sakiyama, Y. Li, M. Iwamoto, and K. Ohta. Information-Theoretic Approach to Optimal Differential Fault Analysis. *IEEE Transactions on Information Forensics and Security*, 7(1):109–120, 2012.

[13] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *LNCS*, pages 342–357. Springer, 2011.

[14] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In *FSE*, volume 4593 of *LNCS*, pages 181–195. Springer, 2007.

[15] T. Suzaki and K. Minematsu. Improving the Generalized Feistel. In *FSE*, volume 6147 of *LNCS*, pages 19–39. Springer, 2010.

[16] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi. TWINE : A Lightweight Block Cipher for Multiple Platforms. In *Selected Areas in Cryptography - SAC 2012*, volume 7707 of *LNCS*, pages 339–354. Springer, 2012.

[17] J. Takahashi and T. Fukunaga. Improved differential fault analysis on clefia. In *Fault Diagnosis and Tolerance in Cryptography, 2008. FDTC'08. 5th Workshop on*, pages 25–34. IEEE, 2008.

[18] U.S Departement of commerce / national Institute of standards and Technology. Data Encryption Standart DES.

[19] Y. Zheng, T. Matsumoto, and H. Imai. On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In *Advances in Cryptology - CRYPTO '89*, volume 435 of *LNCS*, pages 461–480. Springer, 1989.