Cryptanalyse sur l'attaque par injection de faute sur le DES

A été rédigé pour le Contrôle Continu de Calcul Sécurisé (MIN17218)

Nom, prénom : **SOMANTRI Naufal**
No. Étudiant : **22406972**
Professeur : **Louis GOUBIN**

M1 Informatique
naufal.somantri@ens.uvsq.fr
https://github.com/uvsq22406972/CalculSecurise_DFA_DES.git

Université Paris-Saclay
Année 2024/2025

Table des matières

Question 1 (Attaque par faute sur le DES)

Décrire le plus précisément que vous pouvez le principe d'une attaque par fautes contre le DES. On supposera que l'attaquant est capable d'effectuer une faute sur la valeur de sortie R_{15} du 15^{eme} tour.

Réponse

Dans une attaque par fautes contre le DES, nous allons exploiter les erreurs lors du chiffrement de plusieurs textes clairs en utilisant la même clé secrète pour ainsi trouver des informations sur cette clé. Nous allons supposer un attaquant que nous allons appeler "Charlie". Charlie dispose de plusieurs paires de textes chiffrés (C, C^*) dont chaque paire est associée à un texte clair P qui a été chiffré correctement (C) et mal chiffré (C^*). Cette attaque consiste donc à analyser la différence entre C et C^*, qu'on va nommer $\Delta(C)$.

D'après l'attaque décrite par Biham et Shamir [1], on suppose qu'un bit de la partie droite du chiffrement DES a été modifié sur une position aléatoire lors d'un tour de Feistel dans le chiffré fauté (C^*). Dans cette question, on s'intéresse lorsque la faute intervient sur la valeur de sortie R_{15} du 15^{eme} tour. Donc, Charlie doit induire, pour chacune des 32 positions de bits possibles, une faute affectant un seul bit de R_{15}.

Cette attaque est connue pour sa rapidité, notamment si on la compare avec l'attaque par recherche exhaustive en $O(2^{56})$. En pratique, la modification de l'exécution correcte du chiffrement (induire le bit erroné) peut être produite en utilisant des impulsions lumineuses ou des champs magnétiques dans le circuit. Regardons ensemble le schéma de Feistel de Menezes et al. [2] :

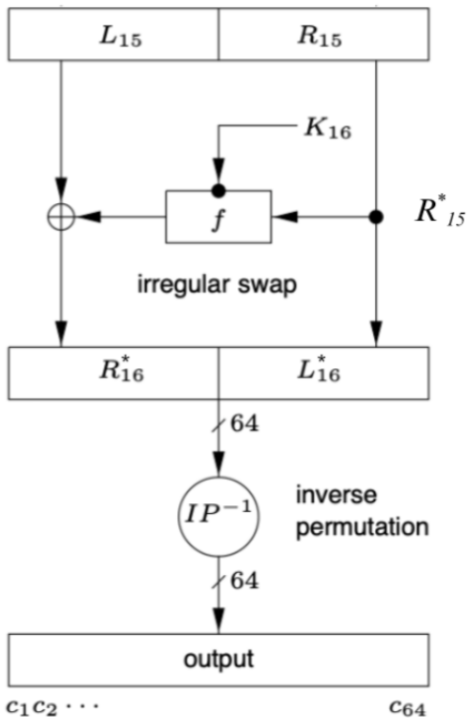

FIGURE 1 – Schéma Feistel du dernier tour du DES avec erreur sur R_{15}

On remarque dans la figure 1 que l'injection de la faute sur R_{15} implique une faute sur R_{16} et L_{16}, dont on va le noter comme R^*_{16} et L^*_{16}. Voici à quoi ressemble l'équation du 16^{eme} tour avec et sans injection de faute.

$$\text{sans faute d'injection :} \quad \begin{cases} R_{16} = L_{15} \oplus f_{K_{16}}(R_{15}) \\ L_{16} = R_{15} \end{cases}$$

$$\text{avec faute d'injection :} \quad \begin{cases} R^*_{16} = L_{15} \oplus f_{K_{16}}(R^*_{15}) \\ L^*_{16} = R^*_{15} \end{cases}$$

Par la suite, nous allons introduire une notation ΔR_{15} qui signifie la différence de XOR entre R_{15} et R^*_{15}, ainsi que $\varepsilon \in \{0,1\}^{32}$ qui signifie l'erreur induite :

$$\Delta R_{15} = R_{15} \oplus R^*_{15} = \varepsilon$$

Idem pour ΔL_{15} , ΔR_{16} , et ΔL_{16} :

$$\begin{aligned} \Delta L_{15} &= 0 \\ \Delta R_{16} &= R_{16} \oplus R^*_{16} \\ \Delta L_{16} &= L_{16} \oplus L^*_{16} = \Delta f_{16} \end{aligned}$$

Pour ΔL_{15}, cela implique que $L_{15} = L^*_{15}$, donc aucune faute sur L_{15}.

En supposant une injection par faute sur la valeur de sortie R_{15}, on a donc cette équation pour R^*_{15} :

$$R^*_{15} = R_{15} \oplus \varepsilon$$

En analysant ces équations, on peut en tirer une conclusion que, pour que Charlie puisse récupérer la valeur de la clé K_{16} qui est utilisée pour chiffrer au 16^{eme} tour, il doit comparer la valeur de XOR entre R_{16} et R^*_{16}, soit ΔR_{16} :

$$\begin{aligned} \Delta R_{16} &= L_{15} \oplus f_{K_{16}}(R_{15}) \oplus L_{15} \oplus f_{K_{16}}(R^*_{15}) \\ \Delta R_{16} &= \underbrace{L_{15} \oplus L_{15}}_{=0} \oplus f_{K_{16}}(R_{15}) \oplus f_{K_{16}}(R^*_{15}) \\ \Delta R_{16} &= f_{K_{16}}(R_{15}) \oplus f_{K_{16}}(R^*_{15}) \\ \Delta R_{16} &= f_{K_{16}}(L_{16}) \oplus f_{K_{16}}(L^*_{16}) \end{aligned}$$

Maintenant, je vais expliquer comment résoudre cette équation. Soit $K_{16,i}$ la clé utilisée dans le 16^{eme} tour avec i le i^{eme} boîte de substitution, S_i le i^{eme} boîte de substitution, et E_i le chiffrement effectué sur le i^{eme} boîte de substitution. Voici le schéma de la fonction interne f du DES par Menezes et al. [2]

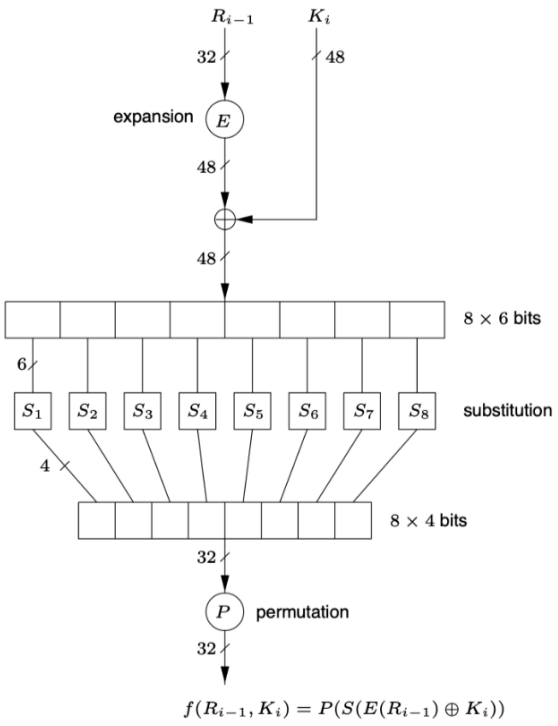

FIGURE 2 – Fonction interne f du DES

Dans la figure 2, la fonction f prend R_{15} en entrée (32 bits) et une clé $K_{16,i}$ (48 bits). Ensuite, il faut tout d'abord appliquer la fonction E, courte pour extension, sur R_{15} afin de passer de 32 à 48 bits en dupliquant certains bits. Maintenant, on peut faire un XOR entre R_{15} et $K_{16,i}$. Après cela, on divise le résultat du XOR en 8 boîtes de substitutions, donc 6 bits chacuns. Ces boîtes de substitutions calculent de manière indépendante entre l'une et l'autre pour en final avoir 4 bits en sortie chacuns. On reviendra donc au nombre de bits de R_{15} de départ, 32 bits. Et à la fin, il faut appliquer une fonction de permutation (P) sur les 32 bits choisis.

Voici comment on représente l'équation de la fonction interne f du DES avec et sans injection de faute sur la valeur de sortie de R_{15} :

$$\text{sans faute d'injection :} \quad \begin{cases} f_{K_{16}}(R_{15}) &= P\Big(S_i\big(E_i(R_{15}) \oplus K_{16,i}\big)\Big) \\ f_{K_{16}}(L_{16}) &= P\Big(S_i\big(E_i(L_{16}) \oplus K_{16,i}\big)\Big) \end{cases}$$

$$\text{avec faute d'injection :} \quad \begin{cases} f_{K_{16}}(R^*_{15}) &= P\Big(S_i\big(E_i(R^*_{15}) \oplus K_{16,i}\big)\Big) \\ f_{K_{16}}(L^*_{16}) &= P\Big(S_i\big(E_i(L^*_{16}) \oplus K_{16,i}\big)\Big) \end{cases}$$

Et voici la représentation de la répartition des bits dans les boîtes de substitution :

$$\begin{aligned}
\text{Bits 1 à 6} &: S_1\big(E_1(R^*_{15}) \oplus K_{16,1}\big)\\
\text{Bits 7 à 12} &: S_2\big(E_2(R^*_{15}) \oplus K_{16,2}\big)\\
\text{Bits 13 à 18} &: S_3\big(E_3(R^*_{15}) \oplus K_{16,3}\big)\\
\text{Bits 19 à 24} &: S_4\big(E_4(R^*_{15}) \oplus K_{16,4}\big)\\
\text{Bits 25 à 30} &: S_5\big(E_5(R^*_{15}) \oplus K_{16,5}\big)\\
\text{Bits 31 à 36} &: S_6\big(E_6(R^*_{15}) \oplus K_{16,6}\big)\\
\text{Bits 37 à 42} &: S_7\big(E_7(R^*_{15}) \oplus K_{16,7}\big)\\
\text{Bits 43 à 48} &: S_8\big(E_8(R^*_{15}) \oplus K_{16,8}\big)
\end{aligned}$$

Pour réussir à appliquer cette attaque par faute sur le DES, il faut impérativement supprimer la permutation P de l'équation. En faisant cela, on aura la même position de bits entre la fin de la fonction interne f et à la sortie des boîtes de substitution. Pour cela, on applique la permutation inverse P^{-1} :

$$\Delta R_{16} \;=\; f_{k_{16}}\big(R_{15}\big) \;\oplus\; f_{k_{16}}\big(R^*_{15}\big)$$

$$P^{-1}\big(\Delta R_{16}\big) = P^{-1}\Big(P\big(S_i(E_i(R_{15}) \oplus K_{16,i})\big) \;\oplus\; P\big(S_i(E_i(R^*_{15}) \oplus K_{16,i})\big)\Big)$$

Avec la propriété de XOR ($f(a \oplus b) = f(a) \oplus f(b)$) :

$$P^{-1}\big(\Delta R_{16}\big) = P^{-1}\Big(P\big(S_i(E_i(R_{15}) \oplus K_{16,i})\big)\Big) \;\oplus\; P^{-1}\Big(P\big(S_i(E_i(R^*_{15}) \oplus K_{16,i})\big)\Big)$$

$$\Longleftrightarrow$$

$$P^{-1}\big(\Delta R_{16}\big) = S_i\big(E_i(R_{15}) \oplus K_{16,i}\big) \;\oplus\; S_i\big(E_i(R^*_{15}) \oplus K_{16,i}\big)$$

Maintenant, on l'écrit sous forme i $\in \{1, 2, 3, 4, 5, 6, 7, 8\}$:

$$\begin{aligned}
\text{Bits 1 à 6} &: S_1\big(E_1(R_{15}) \oplus K_{16,1}\big) \oplus S_1\big(E_1(R^*_{15}) \oplus K_{16,1}\big)\\
\text{Bits 7 à 12} &: S_2\big(E_2(R_{15}) \oplus K_{16,2}\big) \oplus S_2\big(E_2(R^*_{15}) \oplus K_{16,2}\big)\\
\text{Bits 13 à 18} &: S_3\big(E_3(R_{15}) \oplus K_{16,3}\big) \oplus S_3\big(E_3(R^*_{15}) \oplus K_{16,3}\big)\\
\text{Bits 19 à 24} &: S_4\big(E_4(R_{15}) \oplus K_{16,4}\big) \oplus S_4\big(E_4(R^*_{15}) \oplus K_{16,4}\big)\\
\text{Bits 25 à 30} &: S_5\big(E_5(R_{15}) \oplus K_{16,5}\big) \oplus S_5\big(E_5(R^*_{15}) \oplus K_{16,5}\big)\\
\text{Bits 31 à 36} &: S_6\big(E_6(R_{15}) \oplus K_{16,6}\big) \oplus S_6\big(E_6(R^*_{15}) \oplus K_{16,6}\big)\\
\text{Bits 37 à 42} &: S_7\big(E_7(R_{15}) \oplus K_{16,7}\big) \oplus S_7\big(E_7(R^*_{15}) \oplus K_{16,7}\big)\\
\text{Bits 43 à 48} &: S_8\big(E_8(R_{15}) \oplus K_{16,8}\big) \oplus S_8\big(E_8(R^*_{15}) \oplus K_{16,8}\big)
\end{aligned}$$

On obtient ainsi 8 équations. On utilise seulement ceux dont $P^{-1}(\Delta R16) \neq 0$, car si $P^{-1}(\Delta R16) = 0$, on aura une équation de ce sort :

$$\begin{gathered}
S_i\big(E_i(R_{15}) \oplus K_{16,i}\big) \;\oplus\; S_i\big(E_i(R^*_{15}) \oplus K_{16,i}\big) = 0\\
\Longleftrightarrow\\
S_i\big(E_i(R_{15}) \oplus K_{16,i}\big) = S_i\big(E_i(R^*_{15}) \oplus K_{16,i}\big)\\
\Longleftrightarrow\\
E_i(R_{15}) = E_i(R^*_{15})\\
\Longleftrightarrow\\
E_i(\varepsilon) = 0
\end{gathered}$$

Ce qui indique que le vecteur d'erreur n'affecte pas les bits d'entrée de la i^{eme} boîte de substitution. Ainsi, avec ces 8 équations à notre disposition, le seul inconnu dans chaque équation est le $K_{16,i}$ (6 bits) pour chaque paire (C, C^*) dont on retrouve des boîtes de substitution où $P^{-1}(\Delta R_{16}) \neq 0$. Il faut donc maintenant tester toutes les valeurs possibles de $K_{16,i}$ en utilisant la recherche exhaustive pour chaques boîtes de substitution. Il y a donc 2^6 valeurs de $K_{16,i}$ possibles pour une boîte de substitution. Comme on a 8 boîtes de substitutions, il y a donc $8 \times 2^6 = 2^3 \times 2^6 = 2^9$ valeurs de $K_{16,i}$ possibles. En parcourant les 32 chiffrés fautés, soit 2^5, on arrive donc à une complexité de $O(2^{14})$

Question 2 (Application concrète)

Dans le fichier enonce-cc.txt chacun(e) d'entre vous trouvera un exemple correspondant à l'exécution du DES sur un message clair :

une première fois sans injection de faute, ce qui donne donc le chiffré juste ;

puis 32 fois avec diverses injections de fautes, qui donnent donc 32 chiffres faux ;

Ces 33 exécutions utilisent bien sûr le même message clair en entrée, et la même clé. En revanche pour chaque étudiant, il s'agit d'une clé différente. Le but de la question est pour chacun(e) d'entre vous, de retrouver la clé qui lui a été assignée.

2.1 Décrire précisément ce que vous faites pour retrouver la clé.

Réponse

On applique des fonctions qui exécute étape par étape ce qui a été cité à la réponse de la première question. Pour cette question, je vais expliquer en différentes étapes. J'ai implémenté un code en java avec la gestion de projet Maven (J'espère qu'implémenter le code me rapportera des points bonus, car c'était compliqué).

Voici le lien GitHub : `https://github.com/uvsq22406972/CalculSecurise_DFA_DES.git`

Après d'avoir mis en place les fonctions de bases (Intersection, XOR, conversion bit en hexa, etc.) et d'initialiser les variables du DES (IP, IP^{-1}, $PC1$, $PC2$, P, P^{-1}, E), j'ai effectué ces étapes :

Première étape : Création de R_{16} et L_{16}

- Je prends le texte chiffré correct : `2E 89 6E 21 5F 48 95 31`
- J'applique IP (Initial Permutation) au bloc de 64 bits, en mode EBC, pour avoir : $R_{16} = \text{IP(Cipher)}[0..31]$ et $L_{16} = \text{IP(Cipher)}[32..63]$.

Deuxième étape : Calculer ΔR_{16} et $P^{-1}(\Delta R_{16})$
Pour chaque texte chiffré fauté — i étant l'indice du chiffré fauté ($i \in \{1, 2, \ldots, 32\}$) :

- On calcule la différence entre le chiffré fauté et le chiffré correct :

$$\Delta R_{16}^{(i)} = R_{16} \oplus R_{16}^{*(i)}.$$

– On initialise ensuite $P^{-1}\big(\Delta R_{16}^{(i)}\big)$, que j'appelle pour la suite ΔS :

$$\Delta S \;=\; P^{-1}\big(\Delta R_{16}\big).$$

Le ΔS (32 bits) se décompose ensuite en 8 boîtes de substitution différents, avec 4 bits pour chacune.

Troisième étape : Recherche des candidats de clés
Pour chacun des 8 boîtes de substitution :

– On reconstruit les entrées à l'entrée des boîtes de substitution (appelées E-expansions) de R_{15} (le registre juste avant le dernier tour) pour le texte correct et fauté ($E(R_{15})$ et $E(R_{15}^{*(i)})$) ayant 48 bits chacun.

– Pour chaque valeur de k possibles de la sous-clé de K_{16} en segment de 6 bits, on calcule :

$$s_1 \;=\; S_j\big(E(R_{15})\big) \;\oplus\; k_j, \qquad s_2 \;=\; S_j\big(E(R_{15}^{*(i)})\big) \;\oplus\; k_j.$$

La valeur de k possible est donc entre 0 à 63 inclus, sachant que chaque box contient 6 bits et qu'en représentation décimale, la valeur min possible est 0 et la valeur max possible est 63).

– On garde ensuite seulement où $\Delta S^{(i)}_{[4j\,..\,4j+3]} \neq 0000$:

$$s_1 \;\oplus\; s_2 \;=\; \Delta S^{(i)}_{[4j\,..\,4j+3]}.$$

– On appelle $C_j^{(\text{fauté})}$ l'ensemble local d'une boîte de substitution j :

$$C_j^{(\text{fauté})} \;=\; \big\{\, k_j \in \{0,1,\ldots,63\} \;\big|\; \text{l'équation validé} \,\big\}.$$

Il y aura plusieurs élément dans $C_j^{(\text{fauté})}$ qu'on appelle un ensemble local de candidats de clé.

– On obtient ainsi, pour chaque texte fauté, un ensemble local de candidats de clé de taille 6 bits pour une boîte de substitution j.

$$C_j^{(1)},\; C_j^{(2)},\; \ldots,\; C_j^{(32)}$$

Quatrième étape : Intersection des ensembles de candidats de clés

– Seul les bons 6 bits k_j satisferont toutes les équations simultanément. On peut l'écrire mathématiquement de la façon suivante :

$$C_j^{(\text{final})} \;=\; \bigcap_{i=1}^{32} C_j^{(i)}$$

– Idéalement, après plusieurs chiffrés fautés, il ne restera qu'un seul unique candidat de clé 6 bits par S-box. Dans notre cas, 32 chiffrés fautés est suffisant pour retrouver la clé K_{16}.

– En concaténant les $C_j^{(\text{final})}$ de chaque boîte de substitution, on obtiendra la sous-clé K_{16} de taille 48 bits. En effet, comme on a 8 boîtes de substitution de taille 6 bits, on aura bien 48 bits.

$$K_{16} \;=\; C_1^{(\text{final})} \mid C_2^{(\text{final})} \mid \ldots \mid C_8^{(\text{final})}$$

Cinquième étape : Reconstruction de la clé complète (64 bits)

- On reprends K_{16} et on lui applique une permutation inverse PC2 pour obtenir une clé partielle de taille 56 bits, avec 8 bits de parité qu'on a marqué par "x".
- On énumère toutes les 2^8 valeurs possibles pour ces bits de parité de la façon suivante :
  - ☐ On remplace "x" par l'une des 2^8 valeurs possibles.
  - ☐ On injecte ces 56 bits dans l'emplacement inverse de PC1 pour obtenir 64 bits (avec "y" aux positions de parité).
  - ☐ On calcule les bits de parité (impair) pour chaque octet et on obtient une clé DES valide à 64 bits.
- Pour chaque candidat, on déchiffre le texte chiffré correct et on compare au texte clair attendu.
- La seule clé qui redonne le texte clair de départ (`BA7337581520D580`) est la bonne.

Grâce à cette chaîne d'attaques différentielle, on remonte ainsi de 32 fautés + 1 exécution correcte jusqu'à K_{16}, puis on termine en recherche exhaustive légère sur 2^8 possibilités pour retrouver la clé DES complète.

2.2 Donnez les 48 bits de clé que vous obtenez grâce à cette attaque par fautes.

Réponse

Voici le résultat de l'intersection des ensembles de clés candidats pour chaque boîte de substitution :

$$
\begin{array}{lcll}
\text{Bits 1 à 6} & : & K_{16,1} & = 001100 \\
\text{Bits 7 à 12} & : & K_{16,2} & = 100101 \\
\text{Bits 13 à 18} & : & K_{16,3} & = 001110 \\
\text{Bits 19 à 24} & : & K_{16,4} & = 101000 \\
\text{Bits 25 à 30} & : & K_{16,5} & = 010000 \\
\text{Bits 31 à 36} & : & K_{16,6} & = 111010 \\
\text{Bits 37 à 42} & : & K_{16,7} & = 001010 \\
\text{Bits 43 à 48} & : & K_{16,8} & = 010110
\end{array}
$$

Donc, K_{16} = `001100 100101 001110 101000 010000 111010 001010 010110`

Question 3 (Retrouver la clé complète du DES)

Retrouver la clé complète du DES

Dans la question précédente, on obtient 48 bits de la clé (qui fait au total 56 bits).

3.1 Expliquer comment on peut retrouver les 8 bits manquants.

Réponse
Avant d'expliquer comment on peut retrouver les 8 bits manquants, je vais expliquer les notions de PC1 et PC2.

- PC1 sert à enlever les 8 bits de parité à partir de la clé maître K (on passe de 64 bits à 56 bits). Pour réaliser cela, on supprime les bits de parité aux positions 8, 16, 24, 32, 40, 48, 56 et 64. Ensuite, on permute les 56 bits restants selon un ordre fixé par la table PC1. À la fin, on obtient deux registres C_0 et D_0 de 28 bits chacun (leur concaténation fait 56 bits).

  $$\mathrm{PC}_1(K) \;=\; C_0 \,\|\, D_0$$

  Le but est de mélanger et d'éliminer les bits de parité pour préparer les rotations circulaires des tours.

- PC2 sert à concaténer $C_i \,\|\, D_i$ après rotation au tour i. On fait alors 16 tours pendant lesquels on fait une rotation circulaire (Shift) à gauche de v_i des registres C_i et D_i

  $$v_i \quad \begin{cases} 1 & \text{si } i \in \{1, 2, 9, 16\}, \\ 2 & \text{sinon} \end{cases}$$

  On sélectionne 48 bits parmi ces 56, en supprimant cette fois-ci les bits aux positions 9, 18, 22, 25, 35, 38, 43 et 54 de $C_i \,\|\, D_i$. On permute ensuite les 48 bits sélectionnés selon la table PC2 pour produire K_i (la sous-clé du tour i) qui sera injectée dans la fonction de Feistel à chaque tour. À la fin, on y passe donc de 56 bits à 48 bits.

À la fin, on remarque qu'au bout de 16 tours, on applique une rotation circulaire complet sur les 28 bits de C_0 et D_0. On a donc les propriétés suivants :

$$C_{16} = C_0$$
$$D_{16} = D_0$$

Maintenant, je vais répondre à la question principale où on fera l'inverse de ce que j'ai expliqué en haut. À partir de la clé K_{16} de taille 48 bits, on applique l'inverse de PC2 pour passer de 48 à 56 bits. Les 8 positions marquées "X" correspondent exactement à 8 positions dans $C_0 \,\|\, D_0$ qu'on n'a pas pu déterminer par l'attaque sur les boîtes de substitution (sur les positions 9, 18, 22, 25, 35, 38, 43 et 54) :

$$PC_2^{-1}(K_{16}) \;=\; 00110100x11100010x000x01x110110011x10x1101x0000000110x00$$

On fait une recherche exhaustive sur les valeurs marqués "X" sur notre $PC2^{-1}(K_{16})$ en les remplaçant par des 0 ou 1. Donc, il y a au total $2^8 = 256$ valeurs possibles. Pour toutes ces valeurs possibles, on applique l'inverse de PC1 sur cette clé. On obtient ainsi les 8 bits de parité manquants pour obtenir une clé de 64 bits qu'on va le marquer comme "X" qu'on va les remplacer par 0 ou 1 en suivant la règle des bits de parité impair (Standard pour le DES) pour obtenir la vraie clé de DES (La clé maître K). On testera ensuite cette clé en déchiffrant le chiffré correct. Si on retombe sur le texte clair connu, c'est la bonne clé. Pour cela on peut utiliser une calculette DES.

3.2 Faites-le, et donnez ainsi la valeur complète de la clé qui vous a été assignée.

Réponse D'après le programme que j'ai créé, j'obtiens :

$$K \;=\; PC_1^{-1}\big(PC_2^{-1}(K_{16})\big)$$

$$K \;=\; \texttt{0110101x0000111x1010011x0000001x1100010x1101010x0101011x0000001x}$$

Pour remplacer les "X", il faut respecter la règle de parité impair. C'est-à-dire, pour chaque octet, il faut un nombre impair de bits "1". En décomposant par groupe de 8 bits, on a donc :

$$\begin{array}{lcllcl}
\texttt{0110 101X} & \Rightarrow & 4 \text{ bits de } 1 & \Rightarrow & X = 1 \Rightarrow \texttt{0110 1011} & := \texttt{6B} \\
\texttt{0000 111X} & \Rightarrow & 3 \text{ bits de } 1 & \Rightarrow & X = 0 \Rightarrow \texttt{0000 1110} & := \texttt{0E} \\
\texttt{1010 011X} & \Rightarrow & 4 \text{ bits de } 1 & \Rightarrow & X = 1 \Rightarrow \texttt{1010 0111} & := \texttt{A7} \\
\texttt{0000 001X} & \Rightarrow & 1 \text{ bit de } 1 & \Rightarrow & X = 0 \Rightarrow \texttt{0000 0010} & := \texttt{02} \\
\texttt{1100 010X} & \Rightarrow & 3 \text{ bits de } 1 & \Rightarrow & X = 0 \Rightarrow \texttt{1100 0100} & := \texttt{C4} \\
\texttt{1101 010X} & \Rightarrow & 4 \text{ bits de } 1 & \Rightarrow & X = 1 \Rightarrow \texttt{1101 0101} & := \texttt{D5} \\
\texttt{0101 011X} & \Rightarrow & 4 \text{ bits de } 1 & \Rightarrow & X = 1 \Rightarrow \texttt{0101 0111} & := \texttt{57} \\
\texttt{0000 001X} & \Rightarrow & 1 \text{ bit de } 1 & \Rightarrow & X = 0 \Rightarrow \texttt{0000 0010} & := \texttt{02}
\end{array}$$

La clé complète est donc :

$$K \;=\; \texttt{6B 0E A7 02 C4 D5 57 02}$$

On voit bien que toutes les octets respectent la règle de *parité impair*.

De plus, d'après le site mentionné dans la feuille du CC (`http://www.emvlab.org/descalc/`), cette clé permet bien d'obtenir le texte clair qui m'a été assigné dans le fichier `enonce-cc.txt`.

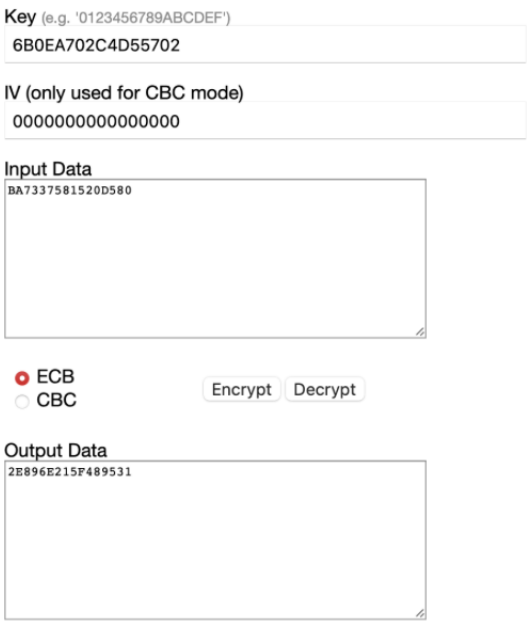

FIGURE 3 – Le texte clair obtenu en utilisant la clé K

Question 4 (Fautes sur les tours précédents)

Les questions précédentes supposent que l'attaquant provoque une faute sur la valeur de sortie R_{15} du 15^{eme} tour. Décrivez le plus précisément possible le fonctionnement d'une attaque en supposant cette fois que la faute est provoquée sur la valeur de sortie R_{14} du 14^{eme} tour. Même question si la faute est provoquée sur la valeur de sortie R_{13} du 13^{eme} tour. Et ainsi de suite. Estimez à chaque fois la complexité de l'attaque. Jusqu'à quel tour l'attaque est-elle réaliste ?

Réponse

– **Faute provoquée sur la valeur de sortie** R_{14}

Voici à quoi ressemble l'équation du 15^{eme} tour avec et sans injection de faute.

$$\text{sans injection de faute :} \quad \begin{cases} R_{15} = L_{14} \oplus f_{k_{15}}(R_{14}) \\ L_{15} = R_{14} \end{cases}$$

$$\text{avec injection de faute :} \quad \begin{cases} R^*_{15} = L_{14} \oplus f_{k_{15}}(R^*_{14}) \\ L^*_{15} = R^*_{14} \end{cases}$$

Supposons une injection par faute sur la valeur de sortie R_{14}, on a donc cette équation pour R^*_{14}, avec ε l'erreur induite sur le DES :

$$R^*_{14} = R_{14} \oplus \varepsilon$$

$$\Longleftrightarrow$$

$$\Delta R_{14} = R_{14} \oplus R^*_{14} = \varepsilon$$

Pour mieux le visualiser, l'article de Matthieu Rivain [3] a bien montré le schéma d'attaque d'injection par faute sur le DES au début de la 15^{eme} tour :

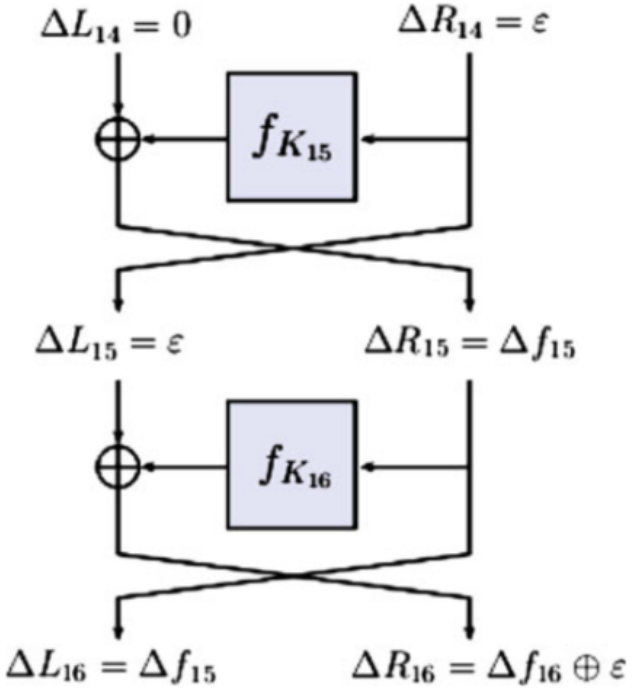

FIGURE 4 – Propagation de l'erreur ε injectée sur R_{14}

Si la faute a été injectée à la valeur de sortie de R_{14}, l'attaquant (Charlie) ne connaît pas la valeur de ε, ni la valeur de K_{16}. Pour réaliser l'attaque, il faut donc commencer à transformer l'équation de la fonction de Feistel afin de trouver un ensemble de solutions possibles pour ε pour ensuite essayer de déduire la valeur de K_{16}. On peut commencer par élaborer la valeur de ΔR_{15} en fonction de R_{14}^*, R_{14} :

$$\Delta R_{15} \;=\; f_{k_{15}}\big(R_{14}\big) \;\oplus\; f_{k_{15}}\big(R_{14}^*\big)$$

Maintenant, on peut essayer de trouver les valeurs possibles de ε. On a deux types d'erreur ε selon la valeur d'entrée d'un S-box. Selon le schéma de Matthieu Rivain [3] :

FIGURE 5 – Deux cas : (gauche) le bit d'erreur ne se duplique pas dans l'expansion ; (droite) le bit est dupliqué et affecte deux boîtes de substitution.

On constate que sur l'image gauche, le bit d'erreur en entrée n'affecte qu'une seule boîte de substitution, car elle n'a pas été dupliquée dans la fonction d'expansion. Alors que sur la droite, le bit d'erreur a été dupliqué dans la fonction d'expansion. Ainsi, l'erreur affecte deux boîtes de substitution. En considérant ces deux différents cas et les boîtes de substitution affectés, on obtiendra les valeurs d'ε possibles.

À partir des valeurs possibles de ε, nous allons commencer à déduire la valeur de K_{16}. À partir de la définition de R_{16} dans la fonction de Feistel, on en déduit que :

$$\begin{gathered}
R_{16} = L_{15} \;\oplus\; f_{k_{16}}\big(R_{15}\big) \\
\Longleftrightarrow \\
R_{16} = L_{15} \;\oplus\; f_{k_{16}}\big(L_{16}\big) \\
\Longleftrightarrow \\
L_{15} \;\oplus\; R_{16} = f_{k_{16}}\big(L_{16}\big) \\
\Longleftrightarrow \\
L_{15} \;\oplus\; R_{16} = P\Big(S_i\big(E_i(L_{16}) \;\oplus\; K_{16,i}\big)\Big) \\
\Longleftrightarrow \\
R_{14} \;\oplus\; R_{16} = P\Big(S_i\big(E_i(L_{16}) \;\oplus\; K_{16,i}\big)\Big)
\end{gathered}$$

On parcourt de la même façon pour R_{16}^*. On obtiendra donc :

$$R_{14}^* \;\oplus\; R_{16}^* \;=\; P\Big(S_i\big(E_i(L_{16}^*) \;\oplus\; K_{16,i}\big)\Big)$$

En faisant un XOR entre ces deux équations, on obtiendra donc :

$$(R_{14} \oplus R_{16}) \;\oplus\; (R_{14}^* \oplus R_{16}^*) = P\Big(S_i\big(E_i(L_{16}) \oplus K_{16,i}\big)\Big) \;\oplus\; P\Big(S_i\big(E_i(L_{16}^*) \oplus K_{16,i}\big)\Big)$$
$$\Longleftrightarrow$$

$$R_{14} \oplus R_{14}^* \oplus R_{16} \oplus R_{16}^* = P\Big(S_i\big(E_i(L_{16}) \oplus K_{16,i}\big)\Big) \oplus P\Big(S_i\big(E_i(L_{16}^*) \oplus K_{16,i}\big)\Big)$$
$$\iff$$
$$\varepsilon \oplus \Delta R_{16} = P\Big(S_i\big(E_i(L_{16}) \oplus K_{16,i}\big)\Big) \oplus P\Big(S_i\big(E_i(L_{16}^*) \oplus K_{16,i}\big)\Big)$$

On enlève la fonction de permutation sur la partie droite :

$$P^{-1}\big(\varepsilon \oplus \Delta R_{16}\big) = P^{-1}\Big(P\big(S_i\big(E_i(L_{16}) \oplus K_{16,i}\big)\big) \oplus P\big(S_i\big(E_i(L_{16}^*) \oplus K_{16,i}\big)\big)\Big)$$

Avec la propriété du XOR $f(a \oplus b) = f(a) \oplus f(b)$:

$$P^{-1}\big(\varepsilon \oplus \Delta R_{16}\big) = P^{-1}\Big(P\big(S_i\big(E_i(L_{16}) \oplus K_{16,i}\big)\big)\Big) \oplus P^{-1}\Big(P\big(S_i\big(E_i(L_{16}^*) \oplus K_{16,i}\big)\big)\Big)$$
$$\iff$$
$$P^{-1}\big(\varepsilon \oplus \Delta R_{16}\big) = S_i\big(E_i(L_{16}) \oplus K_{16,i}\big) \oplus S_i\big(E_i(L_{16}^*) \oplus K_{16,i}\big)$$

Comme avant, nous allons séparer cette fonction en 8 boxes de substitution ($i \in \{1, 2, \ldots, 8\}$) :

$$\begin{aligned}
\text{Bits 1 à 6} &: S_1\big(E_1(L_{16}) \oplus K_{16,1}\big) \oplus S_1\big(E_1(L_{16}^*) \oplus K_{16,1}\big) \\
\text{Bits 7 à 12} &: S_2\big(E_2(L_{16}) \oplus K_{16,2}\big) \oplus S_2\big(E_2(L_{16}^*) \oplus K_{16,2}\big) \\
\text{Bits 13 à 18} &: S_3\big(E_3(L_{16}) \oplus K_{16,3}\big) \oplus S_3\big(E_3(L_{16}^*) \oplus K_{16,3}\big) \\
\text{Bits 19 à 24} &: S_4\big(E_4(L_{16}) \oplus K_{16,4}\big) \oplus S_4\big(E_4(L_{16}^*) \oplus K_{16,4}\big) \\
\text{Bits 25 à 30} &: S_5\big(E_5(L_{16}) \oplus K_{16,5}\big) \oplus S_5\big(E_5(L_{16}^*) \oplus K_{16,5}\big) \\
\text{Bits 31 à 36} &: S_6\big(E_6(L_{16}) \oplus K_{16,6}\big) \oplus S_6\big(E_6(L_{16}^*) \oplus K_{16,6}\big) \\
\text{Bits 37 à 42} &: S_7\big(E_7(L_{16}) \oplus K_{16,7}\big) \oplus S_7\big(E_7(L_{16}^*) \oplus K_{16,7}\big) \\
\text{Bits 43 à 48} &: S_8\big(E_8(L_{16}) \oplus K_{16,8}\big) \oplus S_8\big(E_8(L_{16}^*) \oplus K_{16,8}\big)
\end{aligned}$$

Avec cette équation, on va faire une recherche exhaustive sur la clé K_{16} selon les valeurs possibles de ε. Cette fois, on garde une valeur possible de $K_{16,i}$ ssi la valeur de $K_{16,i}$ satisfait

$$P^{-1}\big(\varepsilon \oplus \Delta R_{16}\big) = P^{-1}\big(\Delta R_{16}\big)$$

pour au moins une valeur possible de ε. On sait que la complexité de cette attaque sur K_{16} est de $\mathcal{O}(2^{14})$.

Pour la suite, il faut faire une deuxième recherche exhaustive pour trouver la clé K_{15}. Pour celui-ci, on se rappelle de cette équation :

$$\Delta R_{15} = f_{k_{15}}\big(R_{14}\big) \oplus f_{k_{15}}\big(R_{14}^*\big)$$

En suivant les étapes similaires à ΔR_{16}, on obtiendra :

$$P^{-1}\big(\Delta R_{15}\big) = S_i\big(E_i(R_{14}) \oplus K_{15,i}\big) \oplus S_i\big(E_i(R_{14}^*) \oplus K_{15,i}\big)$$

Nous allons encore une fois le diviser en 8 boîtes de substitution ($i \in \{1, 2, \ldots, 8\}$) :

$$
\begin{aligned}
\text{Bits 1 à 6} &: S_1\big(E_1(R_{14}) \oplus K_{15,1}\big) \oplus S_1\big(E_1(R^*_{14}) \oplus K_{15,1}\big) \\
\text{Bits 7 à 12} &: S_2\big(E_2(R_{14}) \oplus K_{15,2}\big) \oplus S_2\big(E_2(R^*_{14}) \oplus K_{15,2}\big) \\
\text{Bits 13 à 18} &: S_3\big(E_3(R_{14}) \oplus K_{15,3}\big) \oplus S_3\big(E_3(R^*_{14}) \oplus K_{15,3}\big) \\
\text{Bits 19 à 24} &: S_4\big(E_4(R_{14}) \oplus K_{15,4}\big) \oplus S_4\big(E_4(R^*_{14}) \oplus K_{15,4}\big) \\
\text{Bits 25 à 30} &: S_5\big(E_5(R_{14}) \oplus K_{15,5}\big) \oplus S_5\big(E_5(R^*_{14}) \oplus K_{15,5}\big) \\
\text{Bits 31 à 36} &: S_6\big(E_6(R_{14}) \oplus K_{15,6}\big) \oplus S_6\big(E_6(R^*_{14}) \oplus K_{15,6}\big) \\
\text{Bits 37 à 42} &: S_7\big(E_7(R_{14}) \oplus K_{15,7}\big) \oplus S_7\big(E_7(R^*_{14}) \oplus K_{15,7}\big) \\
\text{Bits 43 à 48} &: S_8\big(E_8(R_{14}) \oplus K_{15,8}\big) \oplus S_8\big(E_8(R^*_{14}) \oplus K_{15,8}\big)
\end{aligned}
$$

La recherche exhaustive permet de trouver des valeurs possibles de K_{15} en $\mathcal{O}(2^{14})$. Ainsi, il nous faut donc en total $\mathcal{O}(2^{28})$ lorsque le bit d'erreur est injecté à la valeur de sortie de R_{14}.

– **Faute provoquée sur la valeur de sortie R_i**

Pour chaque faute sur la valeur de sortie R_i, on procède de la même façon que l'erreur sur la valeur de sortie de R_{14} et R_{15}. C'est-à-dire, on commencera par chercher en effectuant la recherche exhaustive sur les valeurs possibles de K_{16}, puis de K_{15}, etc. jusqu'à K_{i+1}. Ceci implique une suite de recherches exhaustives. Au total, on fera $16 - i$ recherches exhaustives et, pour chaque recherche exhaustive effectuée, on double la complexité.

D'après la loi de Moore, la puissance de calcul disponible double tous les 18 mois. En citant le *CM Cours 3 de l'UE Cryptographie (MIN15122)*, la puissance de calcul considérant faisable en 2020 est de 2^{70} dans le domaine civil et 2^{90} dans le domaine gouvernemental. Donc, en 2025 (environ 60 mois après 2020), la puissance de calcul considérant faisable est de 2^{73} dans le domaine civil et 2^{93} dans le domaine gouvernemental.

– **14e tour** : $i = 13$, donc la faute se trouve sur la valeur de sortie R_{13}. Il faut donc rechercher les valeurs possibles de K_{16}, K_{15} et K_{14}, donc 3 recherches exhaustives au total. Donc, la complexité est de $\mathcal{O}\big((2^{14})^3\big) = \mathcal{O}(2^{42}) \Rightarrow$ Faisable dans le domaine civil et gouvernemental.

– **13e tour** : $i = 12$, donc la faute se trouve sur la valeur de sortie R_{12}. Il faut donc rechercher les valeurs possibles de K_{16}, K_{15}, K_{14} et K_{13}, donc 4 recherches exhaustives au total. Donc, la complexité est de $\mathcal{O}\big((2^{14})^4\big) = \mathcal{O}(2^{56}) \Rightarrow$ Faisable dans le domaine civil et gouvernemental.

– **12e tour** : $i = 11$, donc la faute se trouve sur la valeur de sortie R_{11}. Il faut donc rechercher les valeurs possibles de K_{16}, K_{15}, K_{14}, K_{13} et K_{12}, donc 5 recherches exhaustives au total. Complexité : $\mathcal{O}\big((2^{14})^5\big) = \mathcal{O}(2^{70}) \Rightarrow$ Faisable dans les domaines civil et gouvernemental.

– **11e tour** : $i = 10$, donc la faute se trouve sur la valeur de sortie R_{10}. Recherche exhaustive sur K_{16}, K_{15}, K_{14}, K_{13}, K_{12} et K_{11} (6 recherches). Complexité : $\mathcal{O}\big((2^{14})^6\big) = \mathcal{O}(2^{84}) \Rightarrow$ Faisable dans le domaine gouvernemental, non-faisable dans le domaine civil.

– **10e tour** : $i = 9$, donc la faute se trouve sur la valeur de sortie R_9. Recherche exhaustive sur K_{16}, K_{15}, K_{14}, K_{13}, K_{12}, K_{11} et K_{10} (7 recherches). Complexité : $\mathcal{O}\big((2^{14})^7\big) = \mathcal{O}(2^{98}) \Rightarrow$ Non-faisable dans le domaine gouvernemental, non-faisable dans le domaine civil.

Par contre, si l'on considère le standard de sécurité actuel (2^{128} en 2020, 2^{131} en 2025), il est encore considéré non-sécurisé jusqu'au 8e tour (valeur de sortie de R_7) :

$$\mathcal{O}\big((2^{14})^9\big) = \mathcal{O}(2^{126}) < \mathcal{O}(2^{128}) < \mathcal{O}(2^{140}) = \mathcal{O}\big((2^{14})^{10}\big)$$

Donc, pour en être sûr, il est très recommandé de sécuriser le DES contre ce type d'attaque d'injection par faute sur les 8 derniers tours du DES. Mais, d'après mes recherches, il existe aussi des variantes plus complexes de l'attaque par injection de faute qui utilise la probabilité et

nécessite plus d'hypothèses. Ces variantes se concentrent plutôt à attaquer le début et le milieu du chiffrement DES :

- **Début (1er au 4^{e} tour)** : DFA généralisé en utilisant la collision d'internet introduit par L. Hemme [4].
- **Milieu (9^{e} au 12^{e} tour)** : DFA généralisé en utilisant la vraisemblance et le SEI (Squared Euclidian Imbalance) introduit par Matthieu Rivain [3].

Question 5 (Contre-mesures)

Imaginez une ou plusieurs contre-mesures contre ce type d'attaque par fautes sur le DES. Décrivez-la(les) le plus précisément possible et analysez l'impact sur le temps de calcul (par rapport à une implémentation non sécurisée).

Réponse

Pour répondre à cette question, je propose deux contre-mesures contre ce type d'attaque par fautes sur le DES.

- **Technique de duplication spatiale**
  D'après l'article de Barenghi [5], la duplication spatiale consiste à exécuter le chiffrement DES sur deux circuits indépendants et à comparer le résultat du chiffrement entre les deux circuits. Il existe aussi une variation de cette technique où un circuit exécute le chiffrement DES, puis un autre circuit déchiffre le résultat obtenu depuis le premier circuit. Dans cette technique, on compare le texte clair avec le texte qui a été chiffré puis déchiffré. Cette approche nécessite une plus grande consommation d'énergie par rapport à une implémentation non sécurisée, mais n'a aucun impact sur le temps de calcul.
- **Technique de duplication temporelle**
  D'après le même article [5], pour cette technique, on utilise quasiment la même méthode que la duplication spatiale, mais au lieu d'utiliser deux circuits différents, on exécute le même chiffrement DES deux ou plusieurs fois sur la même puce. La variation de cette technique va donc chiffrer et déchiffrer sur la même machine. Cette méthode augmentera le temps de calcul nécessaire de$n$ fois, avec n le nombre de répétitions, par rapport à une implémentation non sécurisée.
- **Protection matériel**
  À la première question, j'ai expliqué que les bits d'erreur peuvent être injectés en utilisant des impulsions lumineuses ou des champs magnétiques dans le circuit. Pour contrer cela, il existe des contre-mesures niveau matériel liées à la protection du matériel. Selon Paolo et al. [6], on peut par exemple ajouter une sécurisation sur le temps de propagation des multiplexages du circuit pour détecter si un bit d'erreur a été injecté. Si le temps de propagation atteint une certaine limite, un alarme se déclenche. Cette technique ne prend aucun temps de calcul supplémentaire, mais le matériel pour détecter cela coûte cher.

Références

[1] Biham, E., Shamir, A. : *Differential Fault Analysis of Secret Key Cryptosystems.* In : Kaliski Jr., B. S. (ed.) *Advances in Cryptology – CRYPTO '97.* Lecture Notes in Computer Science, vol. 1294, pp. 513–525. Springer (1997).

[2] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. : *Handbook of Applied Cryptography, Chapter 7 — "Block Ciphers".* CRC Press, 1996.

[3] Rivain, M. : "Differential Fault Analysis on Block Ciphers". In : Joye, M., Tunstall, M. (eds.), *Fault Analysis in Cryptography*, Information Security and Cryptography series, pp. 37–54. Springer-Verlag, Berlin / Heidelberg (2012).

[4] Hemme, L. : *A Differential Fault Attack against Early Rounds of (Triple-)DES.* In : Joye, M., Quisquater, J.-J. (eds.), Lecture Notes in Computer Science, vol. 202, pp. 254–267 (2001).

[5] Barenghi, A., Breveglieri, L., Koren, I., Naccache, D. : "Fault Injection Attacks on Cryptographic Devices : Theory, Practice, and Countermeasures", *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, Nov. 2012.

[6] Paolo Maistri, Régis Leveugle, Lilian Bossuet, Alain Aubert, Viktor Fischer, et al.. Electro-Magnetic Analysis and Fault Injection onto Secure Circuits. VLSI-SoC : Very Large Scale Integration - System- on-Chip, Oct 2014, Mexico, Mexico. 10.1109/VLSI-SoC.2014.7004182. emse-01099025