DFA-DES

RAKOTOMANGA Andrianina

Avril 2021

Table des matières

1 Question 1

L'attaque par faute sur le DES à la position 15 sur la valeur de sortie R15 consiste à envoyer dans le fonctionnement du DES une faute dans un bout du message en train d'être chiffré, plus précisément dans le bout du message de R15, donc au 15ème tour du DES. Cet envoi de faute permet de déduire avec plusieurs chiffré faux obtenu et le chiffré juste de retrouver la clé de chiffrement du message clair. Une faute est envoyé en modifiant un bit de la clé visé. Cette manipulation dans un algorithme cryptographique ne peut être que réalisé directement sur un support physique.

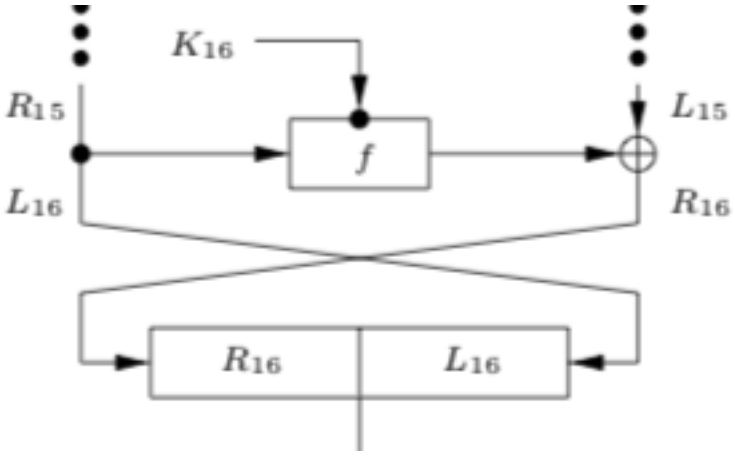

FIGURE 1 – fautes sur R15

Dans la figure ci-dessus, nous envoyons une faute au niveau de R_{15} que nous notons $R15^*$ ainsi lors de l'envoi de la faute on obtient en résultat final : $R16^*$ et $L16^* = R15^*$. Pour essayer de retrouver la clé du DES on va utilisé l'opérateur OU exclusif XOR entre le chiffré juste et les plusieurs chiffrés faux, nous allons

exploiter les différences entre $(L16^* \oplus L16)$ pour trouver les portions différentes de K16 qui est de 48 bits.

En effet selon le schéma du DES et de Feistel on a :
$$\begin{cases} L16 = R15 \\ R16 = L15 \oplus f(R15, K16) \end{cases}$$

De même pour les chiffré fauté :
$$\begin{cases} L16^* = R15^* \\ R16^* = L15 \oplus f(R15^*, K16) \end{cases}$$

Avec 32 chiffré faux nous pouvons déduire à l'aide d'opération logique la clé K_{16} puis la clé K

2 Question 2

2.1 Théorie

Après avoir décrit la méthode d'injection de faute sur le DES, nous allons expliciter une manière efficace de trouver K16 (la clé générée pour le 16 tour du DES), de taille 48 bits à partir du message clair, de son chiffré ainsi que des 32 chiffrés faux générée par l'injection de faute.
En reprenant les équations ci-dessus on peut appliquer un ou exclusif sur $R16^*$ et $R16$ tels que :
$R16^* \oplus R16 = (L15 \oplus f(R15^*, K16)) \oplus (L15 \oplus f(R15, K16))$
Nous pouvons eliminer les $L15$ ce qui nous donne :
$R16^* \oplus R16 = f(R15^*, K16) \oplus f(R15, K16)$

En détaillant le fonctionnement de la fonction de feistel f nous sommes capable de préciser sa nature tel que $f(R15, K16) = P(S(E(R15) \oplus K16))$

S représentant la concatenation des 8 sboxes pour chaque part de 6 bits vers 4 bits de $E(R15) \oplus K16$ (idem pour $R15^*$) tel que.
$P(S(E(R15) \oplus K16)) = P(S1(E(R15) \oplus K16))_{(1,6)}||...||P(S8(E(R15) \oplus K16))_{(43,48)}$

On annule donc la permutation P :
$P^{-1}(R16^* \oplus R16) = (S1(E(R15) \oplus K16)_{(1,4)}||...||S8(E(R15) \oplus K16)_{(29,32)}) \oplus (S1(E(R15^*) \oplus K16)_{(1,4)}||...||S8(E(R15^*) \oplus K16)_{(29,32)})$

On décompose l'équation pour les 8 sbox tels que :
$P^{-1}(R16^* \oplus R16)_{i,i+3} = Sx(E(R15) \oplus K16)_{(i,i+3)} \oplus Sx(E(R15^*) \oplus K16)_{(i,i+3)}$

Avec ces equations seuls K16 est inconnu,nous ecartons d'abord les $P^{-1}(R16^* \oplus R16)_{i,i+3} = 0$ car cela veut dire qu'il n'apporte aucune informations sur les différences entre la portions de clé possible sur une sortie de sbox (nous les appellerons clé de portion candidate).

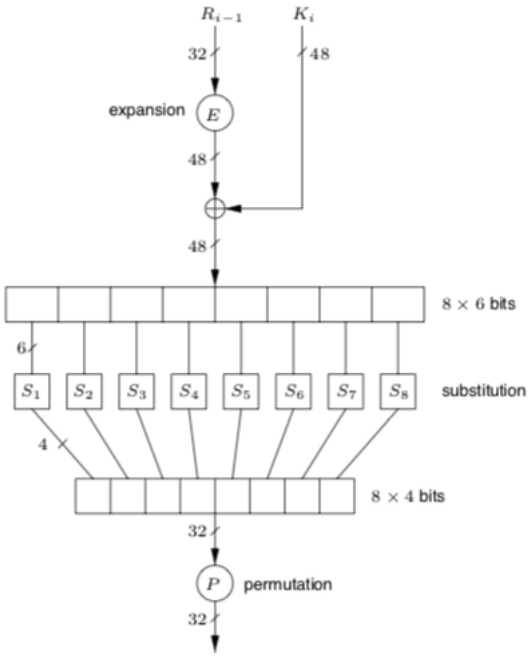

FIGURE 2 – fautes sur R15

Après avoir regrouper les portions de chiffré faux correspondant à l'endroit où on peut trouver leur fautes et sur quel sboxes ces fautes se trouvent, on effectue une attaque exhaustive sur $K16$ qui a $64 = 2^6$ possibilité pour chaque chifré faux candidats d'une sbox. Ce qui donne une complexité de $2^3 * 2^6 * 6 = 2^{10}$ (6 étant le nombre de portion de clé candidate de chiffré faux pour chaque sbox) calcul à effectué pour trouver les portions de clé possible pour chaque sbox.

Lorsque qu'on a déduit nos portions de clé, on vérifie que l'équation soit juste : $P^{-1}(R16^* \oplus R16)_{i,i+3} = Sx(E(R15) \oplus K16)_{(i,i+3)} \oplus Sx(E(R15^*) \oplus K16)_{(i,i+3)}$
Si l'equation est vérifié, la portion de clé devient candidate, cela veut dire qu'elle est une possibilité de portion de clé pour la portion de sbox courante.

Sur toute les portions de clé candidate de chaque sboxes on effectue l'intersection de toute les portions trouvées pour chaque sboxes une par une. L'intersection des portions de clé devient la bonne portion de la clé K16. Répeter pour chaque sboxes de 1 à 8 en concatenant le résultats.
Compléxité total de l'attaque sur K16 :$2^3 * 2^6 * 6 * 2^5 = 3 * 2^{15}$

2.2 Pratique

L'attaque est effectué par les fonctions "R15xorfaulted" permettant de trouver les portions de clé à tester pour chaque chiffré faux et pour chaque sbox, Ainsi que "crackK16" qui permet de trouver K16 grâce à la recherche exhaustive sur ces portions de clé de chaque sbox.

Avec cela on peut effectuer nos attaques exhaustive sur ces $6 * 8$ portion de chiffré faux "crackK16"

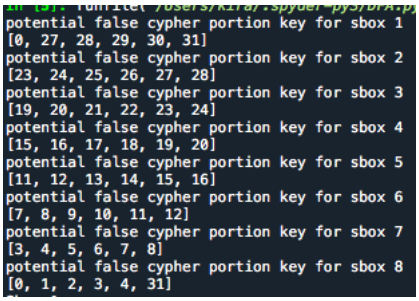

FIGURE 3 – portion de chiffré faux potentiel pour chaque sbox

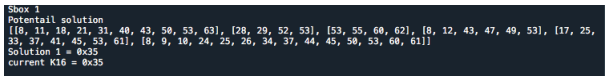

FIGURE 4 – Exemple pour la sbox 1 de recherche de clé potentielle

En exemple la portion du chiffré faux 0 (donc le premier chiffré faux) a pu detecter les portions de clés possible 8, 11, 18, 21, 31, 40, 43, 50, 53, 63 (de 0 à 63 possible). La portion de clé qui se répètent dans ces 6 listes de la premières sbox est 53 (35 en hexadecimale) ce sera donc la portion de clé correcte pour les 6 premier bits de K16 on la concatene donc à notre clé finale K16.

Après analyse des 8 sbox on obtient

```
K16 = D78E9B9E3DC8
```

3 Question 3

3.1 Théorie

Une fois que l'on à trouver K16, il suffit d'observer le fonctionnement du key-Schedule du DES

Au niveau de K16 on passe par $PC2$ puis $PC1$ il nous suffit donc de passer dans les permutation inverse de celle-ci. Cependant pour effectuer $PC2^{-1}$ il faut prendre en compte que cette opération passe de 56 à 48 bits il y'a donc 8 bits perdu après l'opération qui sont les bits au position : 9,18,22,25,35,38,43,54. voire Constante du DES.
Pour $PC1$, on sait que les bits perdu sont les bits de parité au poids faible, pour les retrouver il suffit de completer ces bits en fonction de la parité de chaque portion de bits. Aucune recherche exhaustive est nécessaire.

Après avoir reconstituer les deux boites de permutation $PC1_{-1}$ et $PC2_{-1}$ Il nous suffit d'appliquer la formule $(PC1^{-1}PC2^{-1}(K16))$, retrouver la position des bits perdu et effectuer une recherche exhaustive (2^8) sur les bits perdu ainsi que de retrouver les bits de parité.
La complexité finale de l'attaque globale sera de 2^{15}

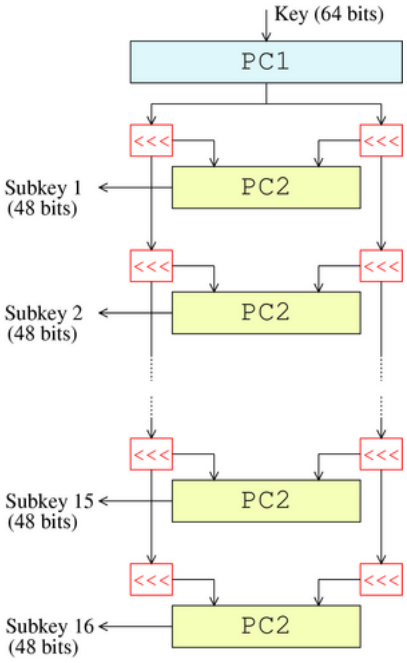

FIGURE 5 – DES Keyschedule

3.2 Pratique

L'attaque est effectué par les fonctions "crackK" et "setParityBit".
Le dernier obstacle est de récupérer les bits perdu après le passage de l'opération $K = (PC1^{-1}PC2^{-1}(K16))$
les fonctions "recoverPosition" et "findposition" applique une analyse des permutations inverse 1 et 2 pour retrouver les positions initilisé à 0 de la $PC2^{-1}$
$[5, 24, 7, 16, 6, 10,$
$20, 18, 0, 12, 3, 15,$
$23, 1, 9, 19, 2, 0,$
$14, 22, 11, 0, 13, 4,$
$0, 17, 21, 8, 47, 31,$
$27, 48, 35, 41, 0, 46,$
$28, 0, 39, 32, 25, 44,$
$0, 37, 34, 43, 29, 36,$
$38, 45, 33, 26, 42, 0,$
$30, 40]$
en passant la $PC2^{-1}$ dans $PC1^{-1}$ on retrouve la position des bits

$[18, 19, 4, 40, 26, 37, 46, -1,$
$20, 9, 13, 30, 33, 0, 0, -1, 10,$
$1, 0, 0, 45, 44, 41,$
$-1, 6, 23, 11, 42, 38, 25, 35,$
$-1, 16, 15, 22, 8, 36, 32, 48,$
$-1, 7, 3, 14, 21, 29, 39, 27,$
$-1, 24, 12, 0, 17, 43, 0, 31,$

$-1, 5, 0, 2, 0, 34, 28, 47, -1]$
les bits perdus sont à la position : 14, 15, 19, 20, 51, 54, 58, 60.après applications des deux permutations.
Pour les remettre dans leurs positions effectives dans la clé finale on effectue un déplacement de 64 bits (taille de la clé).
Leurs positions sont donc à 50, 49, 45, 44, 13, 10, 6, 4. On effectue la recherche exhaustive sur ces 8 bits perdu puis on verifie avec une execution du DES si on trouve la bonne clé. On rajoute ensuite les bits de parité.

```
K = 34F17ADF58B99264
```

3.2.1 Preuve

voire Annexe

4 Question 4

Pour la faute sur le 14 tour donc sur $R14$ nous allons réutiliser la même méthodes employer sur R15.
$R15 = L14 \oplus f(R14, K15)$ de même pour $R15^*$
on fait : $R15^* \oplus R15 = f(R14^*, K15) \oplus f(R14, K15)$
puis : $P^{-1}(R15^* \oplus R15) = (S1(E(R14) \oplus K15)_{(1,4)}||...||S8(E(R14) \oplus K15)_{(29,32)}) \oplus (S1(E(R14^*) \oplus K15)_{(1,4)}||...||S8(E(R14^*) \oplus K15)_{(29,32)})$
On sait par l'équation du DES que : $R14 = L15 = R16 \oplus f(R15, K16)$
En appliquant sur $P^{-1}(R14 \oplus R16) = P^{-1}(R16 \oplus f(L16, K16) \oplus R16) = P^{-1}(f(L16, K16)) = S(E(L16) \oplus K16)$
Idem pour $R14^*$. On effectue une attaque exhaustive pour trouver les potentielles R14 (on connait les R15 mais pas R14).
Puis on effectue une attaque exhaustive sur tout les R14 trouvés via l'équation : $P^{-1}(R15^* \oplus R15) = (S1(E(R14) \oplus K15)_{(1,4)}||...||S8(E(R14) \oplus K15)_{(29,32)}) \oplus (S1(E(R14^*) \oplus K15)_{(1,4)}||...||S8(E(R14^*) \oplus K15)_{(29,32)})$
On applique la même technique que celle pour le tour 15 sauf que l'on a besoin de trouver un valeur suplémentaire (car la faute se propage de manière différente si on l'injecte à un tour plus bas) ce qui augmente le nombre de calcul. La complexité sera de $2^{14^2} : 2^{14}$ pour la recherche de R14 et 2^{14} pour K15
Si on veut trouver la compléxité en fonction du *iemetour*, il suffit de trouver la R_{i-1} portion de la clé puis de remonter à la K_i clé. Ces recherche exahaustive augmente en calcul par 2^{14^i} par tour (i équivaut à nombre de tour de la faute - 15).
à partir du 10ème tour il sera impossible de calculer la clé de manière calculatoire avec notre méthode.

5 Question 5

Les contres mesures face au injection de fautes dans le DES sont possibles à tous les niveaux :

Niveau logicielle : Pour cela il sufit de lancer deux ou n fois le DES sur la même portion de clé/clair et de vérifié si le chiffré est correct et est identique. Cela

permet de détecter une injection de faute.Le temps de calcul sera multiplié par le nombre d'execution supplémentaire du DES. Elle ne permet pas d'éviter une injection mais de prévenir de cette dernière.

Niveau algorithmique : Il nous faut tester l'implementation du DES en y ajoutant des phases de tests continu directement dans l'algorithme.En vérifiant les valeurs courantes de portion de clé ou de chiffré dans un tour de DES on permet à l'algorithme de détecté des anomalies. On adresse un seuil réglementaire sur le nombre d'anomalies detecté et on affiche/arrête le programme si le seuil est dépassé. Le temps d'éxécution dépendra du nombre de tests éffectué.

Niveau Matérielle : Sur le matérielle informatique, il nous est possible de directement sécuriser le système on annulant les effets physiques sur le systèmes. Les injections de fautes se font majoritairement directement sur les composantes physique, via suréxposition, laser, injection electrique/electromagnétique etc... Pour pouvoir contrecarrer ces attaques on peut renforcer le matériel informatique en protection solaire, filtre opaque matériaux non reflexif...Aucun temps d'execution n'y est affecté mais cela augmenterait le coups des matériaux informatique pour ce logiciel.

Annexe

Question 4