Table des matières

1 Attaque par faute sur le DES **2**

2 Application concrète **4**
2.1 *Décrire précisément ce que vous faites pour retrouver la clé.* . . . 4
2.2 *Donnez les 48 bits de clé obtenus grâce à cette attaque par faute.* 5

3 Retrouver la clé complète du DES **5**
3.1 *Expliquer comment on peut retrouver les 8 bits manquants.* . . . 5
3.2 *Donnez la valeur de la clé secrète qui vous a été attribuée.* 6

4 Contre-mesures **6**

1 Attaque par faute sur le DES

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

Tout d'abord, on suppose que l'attaquant a accès à la carte réalisant les calculs et qu'il peut récupérer le chiffrement associé à un message en clair qu'il fournit. On suppose en outre que cet attaquant dispose d'un moyen d'injection de faute qu'il peut mettre en oeuvre pour fausser la valeur de sortie de R_{15} lors du $15^{ème}$ tour de l'algorithme. Le but de l'attaquant est alors de retrouver la clé du DES en utilisant l'injection de fautes.

Remarque 1 *Dans la suite, une valeur fautée sera dénotée par le symbole $\star$. Par exemple la valeur de R_{15} fautée sera $R_{15}^{\star}$.*

Remarque 2 *La notation suivante : $V_{x \to y}$ indiquera que l'on considère uniquement les bits d'indices x à y de la valeur V (cette numérotation commençant à gauche de la représentation binaire et à l'indice 1).*

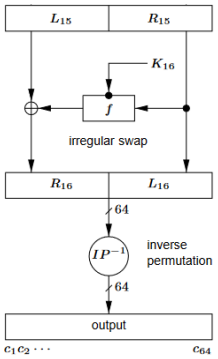

FIGURE 1 – Dernier round et fin du DES

En première constatation, on déduit du schéma les équations suivantes :

$$\begin{cases} L_{16} &= R_{15} \\ R_{16} &= L_{15} \oplus f(R_{15}, K_{16}) \\ L_{16}^{\star} &= R_{15}^{\star} \\ R_{16}^{\star} &= L_{15} \oplus f(R_{15}^{\star}, K_{16}) \end{cases}$$

On va donc chercher à retrouver la sous-clé K_{16} en se servant de la connaissance de R_{15} et $R_{15}^{\star}$.

Décrivons en détail la méthodologie de l'attaque.

1. Pour un même message en clair, l'attaquant génère le message chiffré correct correspondant et un ensemble de messages fautés (dans notre cas 32 chiffrés fautés). Il faudra faire en sorte d'avoir un nombre suffisant de chiffrés fautés, avec des fautes réparties sur chaque portion de 6 bits des valeurs de R_{15}.

2. En appliquant la permutation initiale IP au chiffré correct, il récupère les valeurs R_{16} et $L_{16} = R_{15}$. Il procède de la même manière avec les chiffrés fautés pour récupérer les $R^{\star}_{15}$.

3. Pour chaque chiffré fauté, on calcule :

$$\begin{aligned} R_{16} \oplus R^{\star}_{16} &= (L_{15} \oplus f(R_{15}, K_{16})) \oplus (L_{15} \oplus f(R^{\star}_{15}, K_{16})) \\ &= f(R_{15}, K_{16}) \oplus f(R^{\star}_{15}, K_{16}) \\ &= P(S(E(R_{15}) \oplus K_{16})) \oplus P(S(E(R^{\star}_{15}) \oplus K_{16})) \end{aligned}$$

Comme la permutation P de la fonction f du DES est linéaire, on a :

$$P^{-1}(R_{16} \oplus R^{\star}_{16}) = S(E(R_{15}) \oplus K_{16}) \oplus S(E(R^{\star}_{15}) \oplus K_{16})$$

Or, la sortie de la fonction S correspond à la concaténation des sorties des différentes S-boxes du DES. Chacune de ces S-Boxes prend en argument 6 bits d'entrée et donne une sortie sur 4 bits. Pour chacun des chiffrés fautés, on obtient donc 8 équations :

$$\begin{aligned}
&P^{-1}(R_{16} \oplus R^{\star}_{16})_{1\to4} = S_1(E(R_{15})_{1\to6} \oplus K_{16_{1\to6}}) \oplus S_1(E(R^{\star}_{15})_{1\to6} \oplus K_{16_{1\to6}}) \\
&P^{-1}(R_{16} \oplus R^{\star}_{16})_{5\to8} = S_2(E(R_{15})_{7\to12} \oplus K_{16_{7\to12}}) \oplus S_2(E(R^{\star}_{15})_{7\to12} \oplus K_{16_{7\to12}}) \\
&P^{-1}(R_{16} \oplus R^{\star}_{16})_{9\to12} = S_3(E(R_{15})_{13\to18} \oplus K_{16_{13\to18}}) \oplus S_3(E(R^{\star}_{15})_{13\to18} \oplus K_{16_{13\to18}}) \\
&P^{-1}(R_{16} \oplus R^{\star}_{16})_{13\to16} = S_4(E(R_{15})_{19\to24} \oplus K_{16_{19\to24}}) \oplus S_4(E(R^{\star}_{15})_{19\to24} \oplus K_{16_{19\to24}}) \\
&P^{-1}(R_{16} \oplus R^{\star}_{16})_{17\to20} = S_5(E(R_{15})_{25\to30} \oplus K_{16_{25\to30}}) \oplus S_5(E(R^{\star}_{15})_{25\to30} \oplus K_{16_{25\to30}}) \\
&P^{-1}(R_{16} \oplus R^{\star}_{16})_{21\to24} = S_6(E(R_{15})_{31\to36} \oplus K_{16_{31\to36}}) \oplus S_6(E(R^{\star}_{15})_{31\to36} \oplus K_{16_{31\to36}}) \\
&P^{-1}(R_{16} \oplus R^{\star}_{16})_{25\to28} = S_7(E(R_{15})_{37\to42} \oplus K_{16_{37\to42}}) \oplus S_7(E(R^{\star}_{15})_{37\to42} \oplus K_{16_{37\to42}}) \\
&P^{-1}(R_{16} \oplus R^{\star}_{16})_{29\to32} = S_8(E(R_{15})_{43\to48} \oplus K_{16_{43\to48}}) \oplus S_8(E(R^{\star}_{15})_{43\to48} \oplus K_{16_{43\to48}})
\end{aligned}$$

Parmi ces équations, seules celles telles que $P^{-1}(R_{16} \oplus R^{\star}_{16})_{x \to y} \neq 0$ nous intéressent. En effet, dans le cas contraire, on a :

$$\begin{aligned} & S_k(E(R_{15})_{x \to y} \oplus K_{16_{x \to y}}) \oplus S_k(E(R^{\star}_{15})_{x \to y} \oplus K_{16_{x \to y}}) = 0 \\ \iff & S_k(E(R_{15})_{x \to y} \oplus K_{16_{x \to y}}) = S_k(E(R^{\star}_{15})_{x \to y} \oplus K_{16_{x \to y}}) \end{aligned}$$

Dans ces équations, les bits de $R^{\star}_{15}$ ne sont pas affectées par une erreur et on ne peut donc en déduire de l'information sur les bits de K_{16}.

4. Considérons alors un unique chiffré fauté. "Résoudre" une des équations consiste alors à trouver un ensemble de valeurs possibles pour la portion de K_{16} intervenant dans l'équation. Pour cela, on fait une recherche exhaustive sur ces valeurs possibles et on conserve uniquement celles qui vérifient l'équation.

   Lorsqu'on a procédé ainsi pour chaque chiffré fauté, on a donc pour chaque portion de K_{16} un ensemble d'ensembles de valeurs possibles. Or, on sait que pour chaque portion, la véritable valeur vérifie toujours les équations. Par conséquent, cette véritable valeur doit être présente dans tous les ensembles de valeurs possibles pour la portion. En faisant l'intersection de ces ensembles, on retrouve par conséquent la portion de K_{16}. Une fois cela fait pour chaque portion de K_{16}, on concatène les résultats obtenus pour retrouver K_{16}.

5. Une fois K_{16} connue, on pourra retrouver la valeur de la clé principale K, par un procédé décrit dans la suite de ce devoir.

2 Application concrète

Le code associé à la récupération de la sous-clé K_{16} est dans le fichier *K16_recovery.py* sur le repository Git.

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

Dans notre cas, le message clair, le chiffré correct et les 32 chiffrés fautés nous sont déjà fournis, ce qui équivaut à l'étape 1. On poursuit alors les opérations explicitées précédemment jusqu'à obtenir les ensembles d'ensembles de portions de clés possibles au cours de l'étape 4. On fait l'intersection de ses sous-ensembles de portions pour obtenir chaque portion de K_{16}.

- Portion $K16_{1\to6}$: L'intersection des sous-ensembles vaut : **010100**.
- Portion $K16_{7\to12}$: L'intersection des sous-ensembles vaut : **110100**.
- Portion $K16_{13\to18}$: L'intersection des sous-ensembles vaut : **100001**.
- Portion $K16_{19\to24}$: L'intersection des sous-ensembles vaut : **111000**.
- Portion $K16_{25\to30}$: L'intersection des sous-ensembles vaut : **001011**.
- Portion $K16_{31\to36}$: L'intersection des sous-ensembles vaut : **100100**.
- Portion $K16_{37\to42}$: L'intersection des sous-ensembles vaut : **100100**.
- Portion $K16_{43\to48}$: L'intersection des sous-ensembles vaut : **011101**.

2.2 *Donnez les 48 bits de clé obtenus grâce à cette attaque par faute.*

On récupère alors : $K_{16} = 010100\ 110100\ 100001\ 111000\ 001011\ 100100\ 100100\ 011101$.

3 Retrouver la clé complète du DES

Le code associé à la récupération de la clé principale K est dans le fichier *K_recovery.py* sur le repository Git.

3.1 *Expliquer comment on peut retrouver les 8 bits manquants.*

Intéressons-nous à la façon dont les sous-clés sont générées dans l'algorithme DES.

1. Premièrement, on fournit une clé K de 64 bits. On supprime alors les bits : 8, 16, 24, 32, 4, 48, 56, 64 qui sont les bits de parité.
2. On a à présent une clé sur 56 bits. On applique la permutation $PC1$ sur ces bits, on attribue les 28 premiers bits à C_0 et les 28 suivants à D_0.
3. On fait alors 16 tours pendant lesquels on fait une rotation circulaire à gauche de v_i des registres C_i et D_i (avec $v_i = 1$ si $i \in [1, 2, 9, 16]$ et $v_i = 2$ sinon). On obtient alors C_{i+1} et D_{i+1}. On applique la permutation $PC2$ sur ces deux valeurs pour obtenir la sous-clé K_{i+1} sur 48 bits.

Remarque 3 *On remarque qu'avec $PC2$, on passe de 56 à 48 bits. En effet, $PC2$ supprime les bits : 9, 18, 22, 25, 35, 38, 43 et 54.*

Remarque 4 *Les registres C et D sont sur 28 bits. Au terme des 16 tours, on a appliqué une rotation circulaire de 28 bits. On en déduit donc que :* $C_{16} = C_0$ *et* $D_{16} = D_0$*.*

- Premièrement, on prend K_{16} et on lui applique l'inverse de la permutation PC2. Par la première remarque, on ignore donc la valeur des bits 9, 18, 22, 25, 35, 38, 43 et 54 (marqués par 'x').

  $$PC2^{-1}(K_{16}) = 00100111x00000011x000x11x011011100x10x0001x1101001000x11$$

- Par la deuxième remarque, on sait qu'on a $C_0||D_0$. On fait, une recherche exhaustive de la valeur de la clé : on remplace les 'x' par des 0 ou des 1 (en tout on a $2^8 = 256$ combinaisons possibles). On applique alors l'inverse de la permutation $PC1$ à la clé sur 56 bits obtenue. Puis on ajoute les bits de parité pour obtenir une clé sur 64 bits. On utilise la calculette DES pour vérifier que le déchiffrement du chiffré correct à l'aide de la clé testée donne le texte en clair.

3.2 *Donnez la valeur de la clé secrète qui vous a été attribuée.*

En appliquant la méthode décrite précédemment, on obtient la valeur de clé secrète suivante :

$$K = F6\ B2\ 8D\ 00\ 12\ 9A\ 27\ 28$$

4 Contre-mesures

On distingue deux types de contre-mesures contre les attaques par injection de fautes. Les contre-mesures algorithmiques (qui utilisent les données calculées par la carte pour mettre en place un mécanisme de vérification) et les contre-mesures physiques (qui cherchent à protéger directement les composants de la carte d'une injection d'erreur).

- *Contre-mesures algorithmiques :* Comme expliqué précédemment, on cherche à mettre en place un mécanisme de vérification des calculs de la carte, cela en dégradant le moins possible le temps d'exécution de l'algorithme. La méthode la plus intuitive est de demander à la carte de calculer deux fois le même chiffrement et de comparer les chiffrés obtenus en sortie. Si ces chiffrés diffèrent, alors la carte refuse de renvoyer un quelconque résultat. Cette protection peut être valable car l'injection d'erreur est à priori aléatoire : il est assez peu probable que deux injections d'erreurs successives produisent la même erreur sur les mêmes bits. Ainsi, à quelques rares exceptions près, la carte ne renverra pas de chiffrés fautés : il n'est alors plus possible de mener l'attaque différentielle par injection

de fautes. Le temps de calcul est alors deux fois celui du DES normal, ce qui reste acceptable étant donné qu'en pratique on utilise plutôt le triple DES (donc trois fois le temps d'un DES).

- *Contre-mesures physiques :* Une idée de protection physique serait l'ajout de sondes relevant les tensions et températures dans les différents composants de la carte. Lorsque certains seuils de tension/température sont franchis, la carte considère qu'elle n'est plus en état de fonctionnement normal et cesse son fonctionnement jusqu'à retrouver des seuils acceptables. Avec ce type de protection, le temps de calcul reste le même. Cependant cette protection n'est pas toujours adaptée ou nécessite une réflexion avant mise en place : il ne faut pas que des éléments extérieurs naturels (comme la température plus élevée dans un pays chaud) entraînent un dépassement des seuils et donc un arrêt de fonctionnement de la carte.