TP/Projet : L'algorithme SDES

Introduction

Nous allons utiliser l'algorithme du Simplified-DES (ou SDES) qui est une version simplifiée du DES (Data Encryption Standard). Les principes sont identiques mais la mise en œuvre est plus simple.

1 Vue générale du SDES

L'algorithme d'encryptage du SDES travaille sur la représentation ASCII du texte. Chaque caractère étant codé par 8 bits, ce sont ces 8 bits qui seront modifiés par l'algorithme de cryptage.

Pour utiliser la méthode SDES, on utilisera également une clé de 10 bits appelée clé initiale ou CI (par exemple : 1011110101). Cette clé servira au cryptage et au décryptage du texte. La méthode de cryptage comprend 5 étapes qui sont présentées à la figure 1.

Ces étapes travaillent toutes sur un octet, et sont respectivement :
- Une permutation initiale des bits (IP).
- Une fonction complexe appelée f_k qui comprend des permutation, des substitutions et qui utilise la clé de cryptage.
- Une fonction de permutation (SW) qui échange les 4 premiers bits avec les 4 suivants.
- Une nouvelle application de f_k.
- Une permutation qui est l'inverse de la permutation initiale (IP^{-1}).

La partie complexe de l'algorithme qui utilise la clé de cryptage utilise en fait deux clés de cryptage de 8 bits appelées K_1 et K_2.

Nous allons maintenant détaillé les différentes parties de l'algorithme. Pensez à vous référer à la figure 1 à chaque étape afin de bien suivre (et comprendre) le déroulement de l'algorithme.

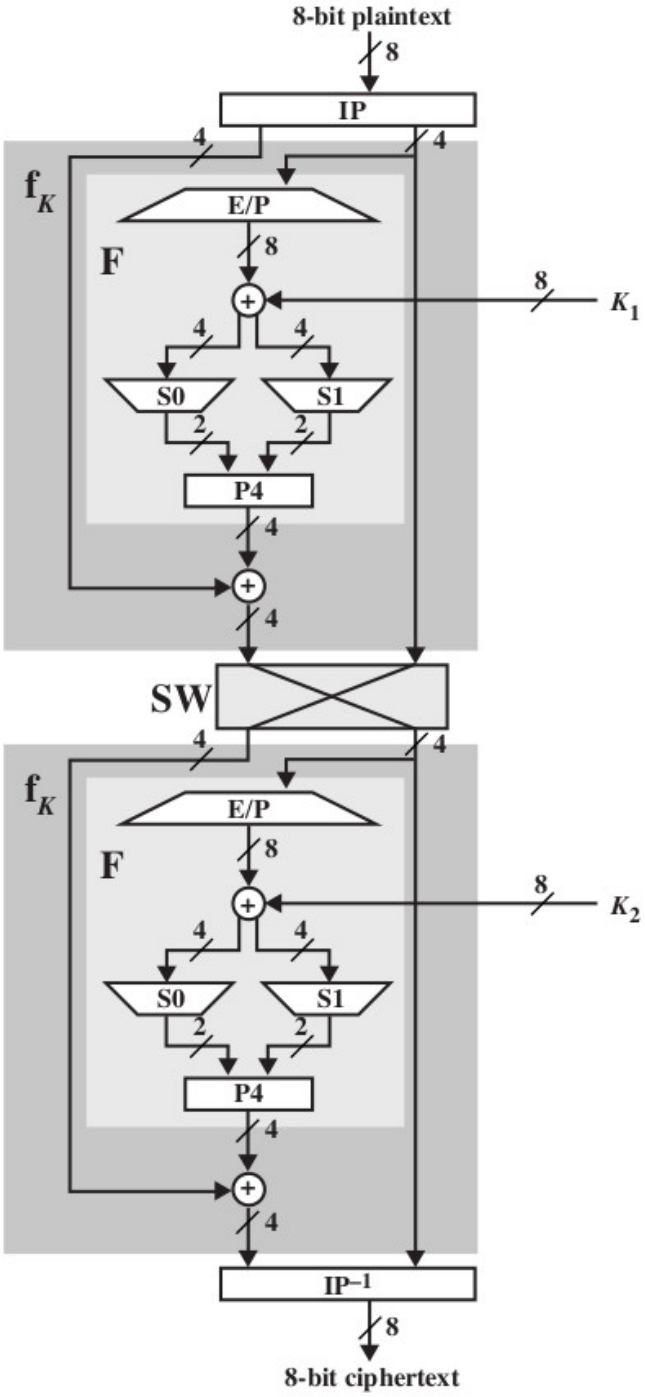

FIGURE 1 – Cœur du SDES.

2 Générations des clés

2.1 Obtention de K_1

La figure 2 résume comment générer les deux sous clés de 8 bits à partir de la clé originale de 10 bits.

Dans une première étape, la clé de 10 bits est permutée. Si les 10 bits de la clé sont respectivement désignés par k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8,k_9 et k_{10}, alors le résultat de la permutation appelée $P10$ est définie de la manière suivante :

$P10(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}) = (k_3, k_5, k_2, k_7, k_4, k_{10}, k_1, k_9, k_8, k_6)$

Par exemple, $P10(1010000010) = 1000001100$.

Ensuite, on réalise un décalage circulaire à gauche (ou rotation gauche) de 1 bit de manière séparée sur chacun des 5 premiers bits et chacun des 5 derniers bits. Un décalage d'un bit à gauche consiste à décaler d'une position vers la gauche chacun des bits.

Le décalage à gauche sur la clé $P10$ précédemment utilisée permet d'obtenir (00001 11000).

Nous réalisons une dernière permutation $P8$ qui réalise une permutation sur 8 des 10 bits de la manière suivante :

$P8(k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_9, k_{10}) = (k_6, k_3, k_7, k_4, k_8, k_5, k_{10}, k_9)$

2.2 Obtention de K_2

Nous repartons des deux suites de 5 bits obtenues après la première rotation et nous réalisons à nouveau deux décalages gauche supplémentaires sur chacune des deux suites (3 rotations auront été faites en tout donc). Dans l'exemple utilisé, les valeurs (00001 11000) deviennent (00100 00011). On applique une nouvelle fois la permutation $P8$ pour obtenir la clé K_2.

En partant de CI, vous devriez obtenir la clé $K2$ (01000011).

3 Coeur du cryptage

Nous allons maintenant nous intéresser au coeur de l'algorithme de cryptographie qui comme indiqué dans la présentation comprend 5 étapes successives.

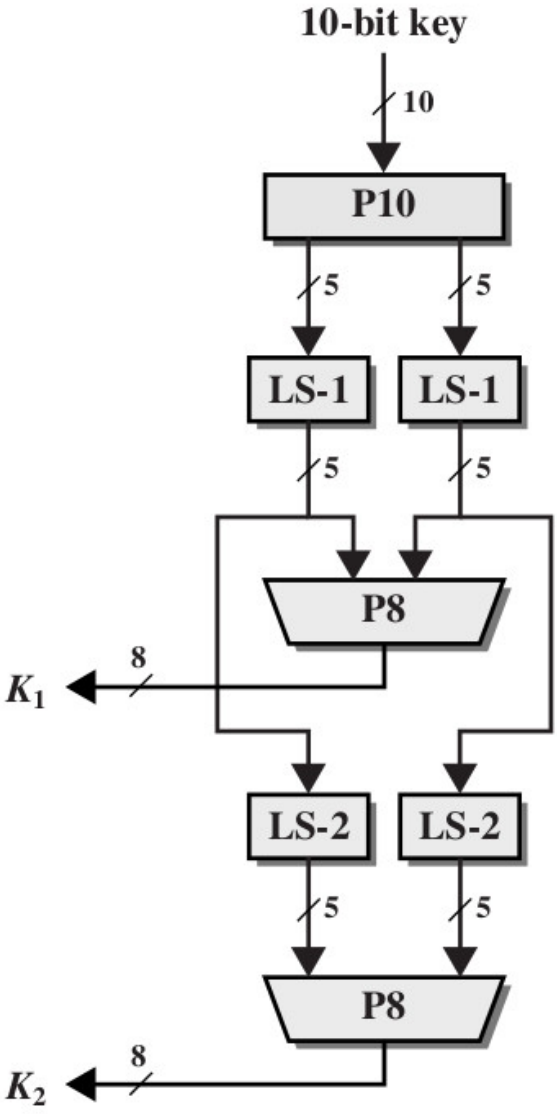

FIGURE 2 – Génération des sous-clés.

3.1 Permutations initiale et finale

Une permutation dans l'algorithme prend en entrée un caractère correspondant à 8 bits. La permutation initiale (IP) est la fonction suivante :
$IP(n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8) = (n_2, n_6, n_3, n_1, n_4, n_8, n_5, n_7)$

On donne également :
$IP^{-1}(n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8) = (n_4, n_1, n_3, n_5, n_7, n_2, n_8, n_6)$

Il est facile de voir que la composition de IP avec IP^{-1} donne la valeur initiale ($IP^{-1}(IP(X)) = X$).

3.2 La fonction f_k

C'est la partie la plus complexe de l'algorithme SDES, elle consiste en une combinaison de permutations et de substitutions. Le principe général de f_k peut être expliqué de la manière suivante. Soit G les 4 premiers bits de gauche et D les 4 bits de droite en entrée de f_k, et F une fonction associant 4 bits à 4 autres bits, alors :
$f_k(G, D) = (G \oplus F(D, SC), D)$
avec SC une sous-clé, et $\oplus$ est l'opérateur ou exclusif (XOR).
Si à la sortie de la permutation IP, l'octet a la valeur (10111101) et $F(1101, SC) = (1100)$ pour la clé SC, alors $f_k(10111101) = (01011101)$ car $(1011) \oplus (1110) = (0101)$.

3.3 La fonction F

L'entrée de la fonction F comporte 4 bits (n_1, n_2, n_3, n_4). La première opération est une opération d'expansion-permutation permettant d'obtenir un groupe de 8 bits appelé EP.
$EP(n_1, n_2, n_3, n_4) = (n_4, n_1, n_2, n_3, n_2, n_3, n_4, n_1)$

La clé K_1 ($k_{1,1}$,$k_{1,2}$,$k_{1,3}$,$k_{1,4}$,$k_{1,5}$,$k_{1,6}$,$k_{1,7}$,$k_{1,8}$) est ajoutée par un ou-exclusif de la manière suivante :

Les 4 premiers bits sont utilisés comme entrée de la matrice S0 afin de produire 2 bits en sortie. Les 4 bits restant sont utilisés afin de produire une sortie de 2 bits supplémentaires grâce à la matrice S1.

$$S0 = \begin{pmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{pmatrix}, \qquad S1 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{pmatrix}$$

Le principe de fonctionnement des Sbox est le suivant : le premier et le 4ème bit sont traités comme un nombre entier de 2 bits qui désigne la ligne de la Sbox, le 2ème et le 3ème bit désignent la colonne de la Sbox ; la cellule correspondant est transformée en base 2 afin de produire une sortie sur 2 bits qui sera la sortie de la Sbox.

Prenons un exemple, si on a en entrée 0110 de S0, alors la sortie sera 3 en décimal donc (11) en base 2.

On obtient donc en sortie de ces Sbox 4 bits, ces 4 bits subissent une dernière permutation appelée P4 qui est la sortie de la fonction F.

$P4(n_1, n_2, n_3, n_4) = (n_2, n_4, n_3, n_1)$

Le calcul de f_k se fait ensuite simplement en réalisant un ou exclusif avec les 4 bits les plus à gauche des 8 bits d'entrée (sortie de IP).

3.4 La fonction de permutation (SW)

Comme la fonction f_k ne modifie que les 4 bits les plus à gauche de l'entrée. La fonction SW échange les 4 bits les plus à gauche avec les 4 bits les plus à droite et on réalise une deuxième application de la fonction f_k pour ces 4 nouveaux bits.

Dans cette deuxième application de la fonction f_k, tout est identique, sauf que la clé K_2 est utilisée.

4 Exercices/TD

A partir de 2 caractères 'A' et 'e' dont les codes ASCII sont respectivement 65 et 101 et d'une clé de 10 bits "1111011001".

4.1 Calculs des clés

1. A partir de la clé, calculer P10.
2. Réaliser un décalage circulaire gauche sur P10 (sur chacun des 5 bits de P10).

3. Appliquer P8 et en déduire la valeur de K1.
4. Réaliser un décalage circulaire gauche 2 fois sur les 10 bits fournis en entrée de P8, appliquer la permutation P8 afin d'obtenir la clé K2.

4.2 Phase de cryptage

1. Transformer ces deux caractères en un octet de 8 bits.
2. Commencer par calculer la permutation initiale IP.
3. A partir des 4 bits les plus à droite, calculer ep.
4. En déduire la valeur des 8 bits p en réalisant un ou exclusif avec la clé K_1.
5. Calculer les valeurs données par S_0 et S_1.
6. Calculer $P4$.
7. Réaliser un ou-exclusif avec les 4 bits le plus à gauche pour obtenir la sortie de ces 4 bits pour la fonction f_k.
8. Faire le même processus pour les 4 bits le plus à droite pour calculer la nouvelle valeur de $P4$.
9. Appliquer IP^{-1} pour obtenir la sortie finale cryptée du caractère A.
10. Appliquer le même processus pour le caractère 'e'.