CRYPTOGRAPHIE

Nous allons dans ce projet écrire un programme de cryptographie ; la cryptographie étant un domaine de recherche très complexe, nous allons nous limiter à l'écriture d'un programme « simple » de cryptographie mais le projet reste cependant assez complexe.

Le principe de la cryptographie est d'utiliser une clef qui permet de crypter un texte, ce texte ne pourra être décodé qu'à l'aide de cette clef (la clef devant être bien évidemment connu seulement de l'émetteur et du récepteur de messages).

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.

Vue générale du SDES

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

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

Ces étapes travaillent toutes sur un octet sont respectivement :

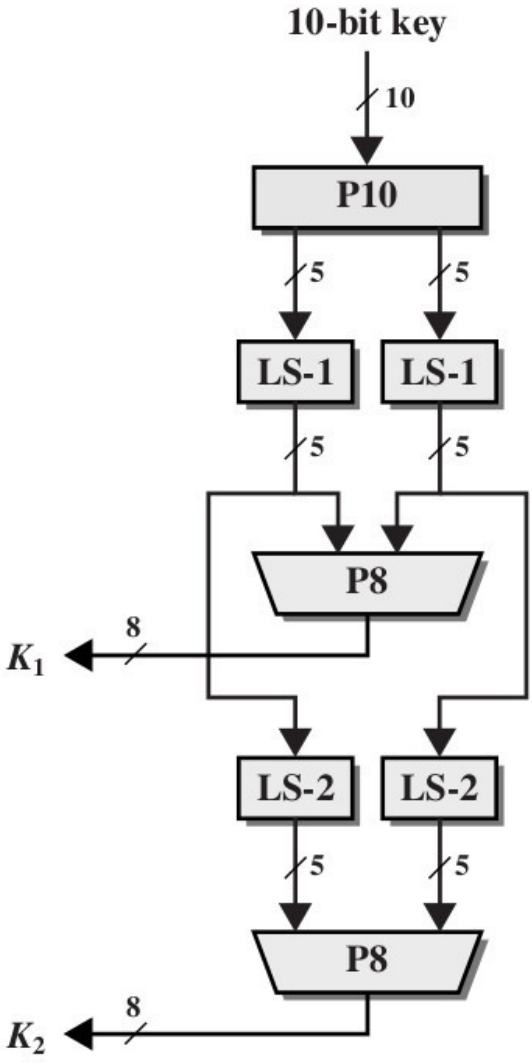
- une permutation initiale des bits (IP)
- une fonction complexe appelée fk qui sera explicitée en TD et comprend des permutation, des substitutions et qui utilise la clef de cryptage
- une fonction de permutation (SW) qui échange les 4 premiers bits avec les 4 suivants
- une nouvelle application de fk
- une permutation qui est l'inverse de la permutation initiale

La partie complexe de l'algorithme qui utilise la clef de cryptage utilise en fait deux clefs de cryptage de 8 bits appelées K1 et K2. Il aurait été possible de fournir deux clefs de 8 bits (ou une de 16 bits) ou d'utiliser une clef de 8 bits deux fois mais le compromis choisi est de générer 2 clefs de 8 bits à partir des 10 bits de la clef originale. C'est ce que nous allons étudier dans la section suivante.

Génération des clefs de SDES

Obtention de la clef K1

SDES génère deux clefs de 8 bits à partir de la clef de 10 bits partagée par l'émetteur et le destinataire d'un message. La figure présente les étapes nécessaires pour produire ces deux sous-



clefs.

Dans une première étape, la clef de 10 bits est permutée. Si les 10 bits de la clef sont respectivement désignés par k1,k2,k3,k4,k5,k6,k7,k8,k9 et k10, alors le résultat de la permutation appelée P10 est définie de la manière suivante :

P10(k1, k2,k3,k4,k5,k6,k7,k8,k9,k10) = (k3,k5,k2,k7,k4,k10,k1,k9,k8,k6)

Ce qui peut être résumé par la figure suivante :

P10									
3	5	2	7	4	10	1	9	8	6

Chaque cellule dans le tableau indique la nouvelle position du bit i dans la clef P10. Par exemple, le 7ème bit de la clef se retrouve à la 4ème position de P10.

Si la clef initiale est (1010000010), P10 sera (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 séparément. 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 clef P10 précédemment utilisé 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							
6	3	7	4	8	5	10	9

Le résultat obtenu après cette permutation est la sous-clef appelée K1. Dans cet exemple, vous devriez obtenir la valeur suivante pour K1 à partir de la CI (10100100).

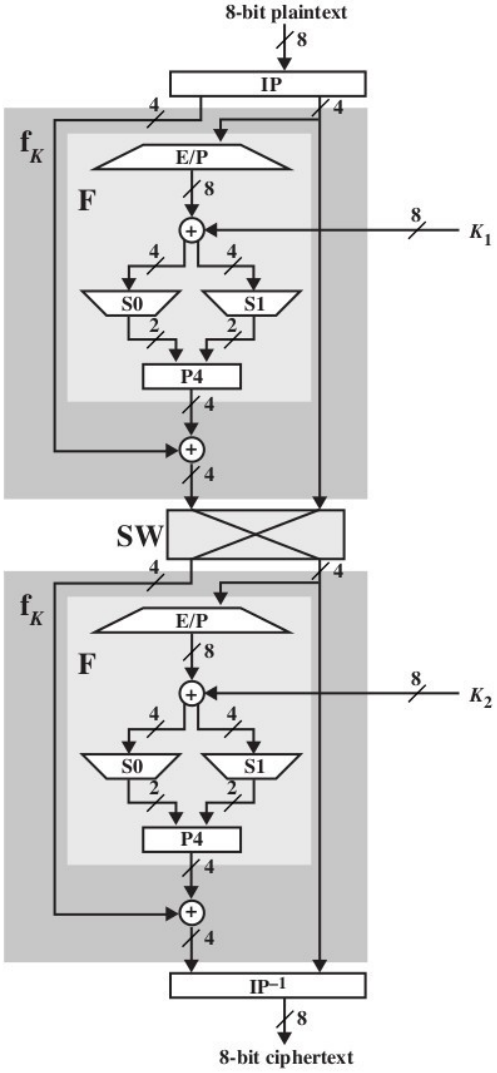
Obtention de la clef K2

Nous repartons des deux suites de 5 bits obtenues après le décalage et nous réalisons à nouveau deux décalages gauche supplémentaire sur chacune des deux suites. Dans l'exemple utilisé, les valeurs (00001 11000) deviennent (00100 00011). On applique une nouvelle fois la permutation P8 pour obtenir la clef K2.

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

Cœur du cryptage

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



Permutations initiale et finales

Une permutation dans notre algorithme de cryptographie prend en entrée un caractère correspondant à 8 bits. La permutation initiale (IP) est la fonction suivante :

IP							
2	6	3	1	4	8	5	7

A la fin de cette phase de cryptage, c'est la permutation inverse IP^{-1} qui est utilisée :

IP^{-1}							
4	1	3	5	7	2	8	6

Il est facile de voir que la composition de IP avec IP-1 retourne la valeur initiale :

$IP^{-1}(IP(X)) = X$

La fonction fk

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 fk 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 fk, et F une fonction associant 4 bits à 4 autres bits, alors :

$fk(G,D) = (L \oplus F(D,SC),D)$

avec SC une sous-clef, et $\oplus$ est l'opérateur ou exclusif (XOR).

Si à la sortie de la permutation IP (figure), l'octet a la valeur (10111101) et F(1101,SC) = (1100) pour la clef SC, alors fk(10111101) = (01011101) car (1011)$\oplus$ (1110) = (0101).un nombre de 4 bits.

Description de F

L'entrée de la fonction F comporte 4 bits que nous appellerons pour simplifier (n1 n2 n3 n4). La première opération est une opération d'expansion, permutation permettant d'obtenir un groupe de 8 bits appelé EP.

EP							
4	1	2	3	2	3	4	1

Afin de simplifier la suite, il est plus simple d'écrire le résultat d'EP sous la forme suivante :

n4	n1	n2	n3
n2	n3	n4	n1

La clef K1 (k11,k12,k13,k14,k15,k16,k17,k18) est ajoutée par un ou-exclusif de la manière suivante :

n4⊕k11	n1⊕k12	n2⊕k13	n3⊕k14
n2⊕k15	n3⊕k16	n4⊕k17	n1⊕k18

On renomme les 8 bits obtenus par le calcul suivant de la manière suivante :

p0,0	p0,1	p0,2	p0,3
p1,0	p1,1	p1,2	p1,3

Les 4 premiers bits (première colonne du calcul, ligne p0,?) sont utilisés comme entrée de la matrice S0 afin de produire 2 bits en sortie. Les 4 bits de la seconde ligne (ligne p1,?) sont utilisés afin de produire une sortie de 2 bits supplémentaires grâce à la matrice S1.

Voici les deux matrices

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

Le principe de fonctionnement des Sbox est le suivant : le premier et le 4ème bit sont traités comme un nombre entier de 2bits 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 (p0,0 p0,3) = (00) et (p0,1 p0,2) = (10), alors la sortie de la ligne 0, colonne 2 est 3 ou (11) en base 2. De manière similaire, (p1,0 p 1,3) et (p1,1 p1,2) sont utilisés pour obtenir la cellule de S1 qui donnera 2 bits supplémentaires.

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			
2	4	3	1

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

La fonction de permutation (SW)

Comme vous pouvez le constater, la fonction fk ne modifie que les 4 bits le 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 fk pour ces 4 nouveaux bits.

Dans cette deuxième application de la fonction fk, les fonctions et permutations EP, S0, S1 et P4 sont identiques par contre la clef utilisée est K2.

Exercices – Phase TD

A partir de 2 caractères « A » et « e » dont les codes ASCII sont respectivement 65 et 101 et d'une clef de 10 bits « 1111011001 »

Calculs des clefs

1. transformer ces deux caractères en un octet de 8 bits
2. à partir de la clef, calculer P10 `[1, 0, 1, 1, 1, 1, 1, 0, 0, 1]` ;
3. Réaliser un décalage circulaire gauche sur P10 (sur chacun des 5 bits de P10) ;
4. Appliquer P8 `[1, 1, 0, 1, 0, 1, 1, 1]` et en déduire la valeur de K1 `[1, 1, 0, 1, 0, 1, 1, 1] ;`
5. réaliser un décalage circulaire gauche 2 fois sur les 10 bits fournis en entrée de P8, appliquer la permutation P8 afin d'obtenir la clef K2 `[0, 1, 1, 0, 1, 1, 0, 1]`

Phase de cryptage

Pour chacun des deux caractères réaliser la procédure d'encryption, certains exemples sont donnés pour le caractère A `[0, 1, 0, 0, 0, 0, 0, 1]`

1. Commencer par calculer la permutation initiale IP `[1, 0, 0, 0, 0, 1, 0, 0]` ;
2. à partir des 4 bits les plus à droite [`0, 1, 0, 0]`, calculer ep `[0, 0, 1, 0, 1, 0, 0, 0]` ;
3. en déduire la valeur des 8 bits p en réalisant un ou exclusif avec la clef K1 `[1, 0, 0, 1, 0, 1, 1, 0]` ;
4. calculer les valeurs données par S0 (11) et S1 (00) ;
5. calculer P4 `[1, 0, 0, 1]`
6. Réaliser un ou-exclusif avec les 4 bits le plus à gauche pour obtenir la sortie de ces 4 bits pour la fonction fk `[0, 0, 0, 1]`
7. Faire le même processus pour les 4 bits le plus à droite pour calculer la nouvelle valeur de P4 `[1, 1, 0, 1]` ;
8. Appliquer IP-1 pour obtenir la sortie finale cryptée du caractère A `[1, 0, 1, 1, 0, 1, 1, 0]` ;
9. Appliquer le même processus pour le caractère « e » sortie : `[1, 1, 1, 1, 0, 1, 1, 0]`.

Projet

Vous devez réaliser un programme en Java permettant de crypter un texte contenant des caractères.

La structure de données que nous utiliserons pour stocker les bits sera un tableau de short (le short étant un entier qui peut contenir des valeurs comprises entre -32768 et 32767). Pour déclarer, un ensemble de 8 bits, on utilisera la déclaration suivante **short** [] mes8bits = **new short** [8] ;

Nous utiliserons dans ce projet 2 classes, une classe appelée SDES qui réalisera le cryptage proprement dit et une classe Cryptage qui lira un fichier et le cryptera en faisant appel aux méthodes de la classe SDES.

Commencer par créer la classe SDES, celle ci-contiendra les champs privés suivants :

- un champ key de type String qui servira à contenir la clef sous forme d'une chaîne de caractères ;
- un tableau de 10 short keyB qui contiendra la clef sous forme binaire ;
- un tableau de 8 short K1 qui contiendra la clef K1 ;
- un tableau de 8 short K2 qui contiendra la clef K2

Écrire la méthode SDES (constructeur) de signature `public SDES(String k)` qui initialise la clef binaire keyB à partir de la clef k passée en paramètre, puis calcule successivement les clefs K1 et K2.

Vous aurez besoin d'écrire la méthode `private void leftShift1(byte [] listeB) {}` permettant de réaliser un décalage circulaire gauche de 1 bit.

Écrire ensuite les deux méthodes (accesseurs) permettant d'accéder aux deux clefs K1 et K2 : `public short [] getK1() ; {}` et `public byte [] getK2() ;`

Écrire deux méthodes permettant respectivement de réaliser les permutations IP et IP-1 (`public void makeIP(short [] liste) {}` et `void makeIP_1(short [] liste)`.

Écrire également une méthode `public byte xor(byte entree1, byte entree2) {}` réalisant un ou-exclusif entre deux bits.

A partir de toutes ces méthodes, écrire la méthode `public void fk(short [] entree, short [] sortie) {}` qui réalise le cryptage d'un octet (caractère) de 8 bits contenu dans entree et qui stocke l'octet crypté dans sortie.

Écrire la méthode `public void cryptageSDES(short [] entree, short [] sortie, int passe)` qui permet de réaliser le cryptage complet d'un tableau de 0 et de 1 de 8 bits contenu dans entree en le plaçant dans le tableau sortie.

Écrire une méthode `public static int shortToInt(short [] entree){` qui transforme l'entrée contenue dans le tableau entier en un entier.

Écrire une méthode `public static void charToShort(char ch, short [] sortie){` qui transforme le caractère ch qui est un entier en une chaîne de 0 et de 1 qui sera placée dans le tableau sortie.

Pour tester votre programme, vous utiliserez la classe suivante pour vérifier que vous arrivez bien à crypter un fichier le cryptage étant affiché à l'écran sous forme d'entier :

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Scanner;

public class Cryptage {

	public static void main(String[] args) throws IOException {


		String key = "0010010111";
		SDES sdes = new SDES(key);

		System.out.println("K1 " + sdes.getK1());
		System.out.println("K2 " + sdes.getK2());


		char ch;

		short testShort = new short[8] ;

		InputStreamReader sr = new InputStreamReader(new
FileInputStream("/home/fonlupt/temp/texte.txt"), "utf8");


		while (sr.ready()){
			ch =  (char) sr.read();

			SDES.charToShort(ch, testShort);
			sdes.cryptageSDES(testShort, out);
			System.out.println(SDES.shortToInt(out));


		}
		sr.close();

		System.out.println(fichierLu);

		System.out.println(aCrypter);
```

Bon courage !

Si le fichier « texte.txt » contient « fichier à crypter » vous devriez avoir à l'écran :

```
95
42
121
255
42
196
171
69
115
69
121
171
```

137
218
159
196
171
220