

# TP CRYPTO

Vous allez dans un premier temps implémenter un DES simplifié :

- une seule ronde
- un seul tableau S

## Classe DES

constantes :

**taille\_bloc** = 64  
**taille\_sous\_bloc** = 32  
**nb\_ronde** = 1 (au départ 16 ensuite ...)  
**tab\_decalage** = table des décalages pour création de clé (diapo 27)  
**perm\_initiale** = permutation initiale (diapo 26) :  
il suffit de stoker le tableau PI (Permutation Initiale) : **attention dans le diaporama c'est une table d'indices commençant à 1**  
**S** = table de la fonction S (diapo 30) (tous les  $S_i$  seront identiques dans un premier temps ...)  
**E** = table diapo 28 : **attention dans le diaporama c'est une table d'indices commençant à 1**  
(dans les versions ultérieures cela pourra devenir un tableau de tableaux du genre diapo 8)

attributs :

**masterKey** = tableau de 64 éléments pris au hasard dans {0,1}  
**tab\_cles** : tableau, liste ... de tableaux, listes, ... stockant l'ensemble des clés calculées à chaque ronde

méthodes :

**Des()** : le constructeur , initialise la masterKey et crée tab\_cles

**int[] crypte** (String message\_clair) : message\_code transforme un message chaîne de caractères, en un tableau d'entiers (0 ou 1) résultat du cryptage

**String decrypte**( int[] messageCodé) : décrypte un tableau d'entiers (0 ou 1) résultat d'un cryptage en une chaîne de caractères donnat le message clair.

**int[] stringToBits**(String message) : transforme une chaîne de caractères en un tableau d'entiers : 0 et 1

**String bitsToString**(int[] blocs) : message\_clair : transforme un tableau d'entiers (0 ou 1) en chaîne de caractères.

**int[] generePermutation**( int taille) : génère une table de permutation de taille éléments.

**permutation**(int[] tab\_permutation, int[] bloc) : retourne un bloc qui subi la permutation contenue dans tab\_permutation

**invPermutation**(int[] tab\_permutation, int[] bloc) : retourne un bloc qui subi la permutation inverse de celle contenue dans tab\_permutation

**int[][] decoupage**(int[] bloc, int nbBlocs) : découpe bloc en nbBlocs ...

**int[] recollage\_bloc**(int[][] blocs) : recolle tous les blocs ...

**génèreClé**(int n) : calcule la clé de la n ième ronde, la stocke aussi dans tab\_clés (pour le décryptage ...)

**int[] decalle\_gauche**(int[] bloc, int nbCran) : décalage vers la gauche de nbCran de bloc

**int[] xor** (int[] tab1, int[] tab2 ) réalise le xor entre tab1 et tab2, vous n'aurez peut être pas besoin de mettre des paramètres.

**int[] fonction\_S** ( int[] tab) : fonction S

**int[] fonction\_F**( int[] uneCle, int[] unD) : fonction F, uneCle est une cle Kn stohée dans tabCles : donc pas besoin de ce paramètre

Faire une classe TestDes pour tester votre classe Des.

**deuxième version** : faire les 16 rondes avec les 16 clés, et en faisant varier le tableau S de façon aléatoire.

**Troisième version** : Triple DES

**Quatrième version** : interface graphique

**Indications :**

**1. Faites une classe TestDes dans laquelle vous mettrez le main testant au fur et à mesure vos méthodes.**

**2.** Pour les méthodes `int[] stringToBits(String message)` et `String bitsToString(int[] blocs)` il faudra vous intéresser aux classes :

String  
Integer  
Byte

vous pourrez ainsi utiliser tous les caractères de l'alphabet en minuscules et majuscules ainsi que « \_ » pour séparer les mots.

Si vous voulez utiliser tous les caractères possibles (avec les accents, ponctuation etc....) il faudra chercher encore davantage.

**3.** Ordre dans lequel vous pouvez implémenter et tester les méthodes de la classe DES :

- 1) `Des()`
- 2) `int[] stringToBits(String message)`
- 3) `String bitsToString(int[] blocs)`
- 4) `int[] generePermutation( int taille)`
- 5) `permutation(int[] tab_permutation, int[] bloc)`
- 6) `invPermutation(tab_permutation, bloc)`
- 7) `int[][] decoupage(int[] bloc, int nbBlocs)`
- 8) `int[] recollage_bloc(int[][] blocs)`
- 9) `int[] decalle_gauche(int[] bloc, int nbCran)`
- 10) `int[] xor ( )` on enlève les paramètres .
- 11) `génèreClé(int n)`
- 12) `int[] fonction_S ( int[] tab)`
- 13) `int[] fonction_F( int[] unD)`
- 14) `int[] crypte (String message_clair)`
- 15) `String decrypte( int[] messageCodé)`