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 28)
perm_initiale = permutation initiale (diapo 27) :
il suffit de stoker le tableau PI (Permutation Initiale) :**attention dans le diaporama c'est une table d'indices commençant à 1**
PC1 et PC2 voir diapo 29.
S = table de la fonction S (diapo 32) (tous les S_i seront identiques dans un premier temps … d'autres S_i sont proposés dans les liens proposés dans le cours, vous pourrez en générer d'autres si vous voulez)
dans la deuxième version vous utiliserez 8 tables que vous pourrez stoker dans un tableau de tableaux ...
E = table diapo 30 : **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, mais normalement il est fixé)

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 créé 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[] **genereMasterKey**() : génère une clé aléatoire de 64 bits.

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

int[] **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 tailleBlocs) : découpe bloc en blocs de taille tailleBlocs ...

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) : décallage 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.

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[] **genereMasterKey**()

5) int[] **permutation**(int[] tab_permutation, int[] bloc)

6) int[] **invPermutation**(int[] tab_permutation, int[] bloc)

7) int[][] **decoupage**(int[] bloc, int tailleBlocs)

8) int[] **recollage_bloc**(int[][] blocs)

9) int[] **decalle_gauche**(int[] bloc, int nbCran)

10) int[] **xor (** int[] tab1, int[] tab2)

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)

Tables S1 à S8 (puis on réutilise dans cet ordre)

S_1	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0yyyy1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1yyyy0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1yyyy1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0yyyy1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1yyyy0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1yyyy1	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
0yyyy1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
1yyyy0	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
1yyyy1	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
0yyyy1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
1yyyy0	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
1yyyy1	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
0yyyy1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
1yyyy0	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
1yyyy1	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
0yyyy1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1yyyy0	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
1yyyy1	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
0yyyy1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
1yyyy0	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
1yyyy1	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11