TP CRYPTO

Vous allez implémenter un DES simplifié.

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)
S = table de la fonction S (diapo 30) (tous les S_i seront identiques dans un premier temps ...)
E = table diapo 28

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 une liste de binaires cryptés
String **decrypte**(int[] messageCodé) : décrypte une suite de binaires en une chaîne de caractères.

int[] **stringToBytes**(String message) : transforme une chaîne de caractères en tableaux de tableaux de 64 entiers
String bytesToString(int[] blocs) : message_clair : transforme une liste de bits en chaîne de caractères.

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

permutation(tab_permutation, bloc) : retourne un bloc qui subi la permutation contenue dans tab_permutation

invPermutation(tab_permutation, bloc) : retourne un bloc qui subi la permutation inverse de celle contenue dans tab_permutation

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

int[] **recollage_bloc**(int[][] blocs) : recolle tous les blocs …

génèreClé(int n) : retourne 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

int[] **fonction_S** (int[] tab) : fonction S

(deuxième version : faire les 16 rondes en faisant varier le tableau S de façon aléatoire.)