Introduction à la cryptographie - TP 3

1 Data Encryption Standard

Le **DES** est un dérivé d'un algorithme de chiffrement par bloc répondant au (doux) nom de Lucifer utilisé dans le début des années 70 par **IBM**. Ce dernier fut modifié par la **NSA** avant d'être publié et admis comme standard en 1976.
Le **DES** opère sur des blocs de 64 bits. C'est à dire que le fichier à chiffrer est découpé en de multiples parties de 64 bits surlesquels le DES agira indépendamment . Si la taille du fichier n'est pas multiple de 64, on peut ajouter, à la fin de celui-ci, autant de 0 que nécessaire. Cette opération s'appelle « padding » ou, en français (et beaucoup moins ... poétique), du *bourrage*.
L'algorithme se découpe en 3 étapes :

1. Une permutation initiale **P** des 64 bits à chiffrer.
2. Seize itérations successives basées sur un **réseau de Feistel**.
3. Une permutation finale $\mathbf{P^{-1}}$ (qui est en fait, comme son nom l'indique, la permutation inverse de **P**) sur les 64 bits de la dernière itération.

Les clefs que l'algorithme **DES** utilise sont également sur 64 bits mais seuls 56 sont considérés. Les 8 bits restants sont des bits de « contrôle » (de parité).

1.1 La permutation P initiale

Les 64 bits du bloc à traiter sont mélangés selon la permutation suivante :

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Il faut ainsi comprendre que l'on met le 58 ième bit en première position, le 50 ième bit en deuxième position ... et le 7 bits en 64 ème position. On remarquera aussi que la première moitié $\mathbf{G_0}$ (sur 32 bits) contient tous les bits de rangs pairs et la seconde partie $\mathbf{D_0}$ (sur 32 bits) tous les bits de rangs impairs.

1.2 Les rondes

Au tour de boucle i (allant de 1 à 16) nous manipulerons les objets G_{i-1} et D_{i-1}. Notez que G_0 et D_0 ont bien été définies précédemment.
Au tour de boucle i, on laissera G_{i-1} intact mais on étendra, en revanche, D_{i-1} avec la fonction d'extension **E** suivante :

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Ce tableau se lit comme la permutation **P** auparavant définie. Mais cette fonction d'extension ne se limite pas à "mélanger", elle augmente le nombre de bits. On passe ainsi de 32 bits à 48 bits. À chaque tour de boucle, une nouvelle clef K_i est dérivée de la clef initiale K. Sa génération sera décrite un peu plus tard dans le document. K_i est un élément de 48 bits que l'on va venir « superposer » à $\mathbf{E(D_{i-1})}$ à l'aide d'un **XOR** effectué bit à bit. On appellera le résultat de l'opération **F**.
F est ensuite coupé en 8 morceaux de 6 bits. Chacun de ces morceaux subira une transformation dictée par les **SBOXES** que l'on trouvera en annexe. Une fois cette opération achevée, on se retrouvera de nouveau avec un vecteur de 32 bits.
Enfin, ces 32 bits seront mélangés à l'aide de la permutation suivante :

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

On notera le résultat de l'ensemble de ces opérations β.
Finalement, $G_i = D_{i-1}$ et $D_i = \beta \oplus G_{i-1}$.

1.3 Génération des K_i

Le **DES** opère à l'aide d'une clef K de 64 bits. Une première partie de la clef K_i (que l'on nommera GK_i) s'établira avec la permutation sur K suivante :

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

Une seconde partie de la clef K_i (que l'on nommera DK_i) s'établira avec la permutation sur K suivante :

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Ensuite GK_i et DK_i seront décalés (vers la gauche) d'un certain nombre de bit, dépendant de l'itération courante. Dans l'ordre, les décalages seront les suivants : $1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1$. Il y a bien 16 décalages différents correspondants à chacun des tours de boucle. On recolle les deux "bouts" obtenus et appliquons (de nouveau !) la permutation suivante :

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

1.4 La permutation P^{-1} finale

Une fois que les seize tours de boucle ont été effectuées, il suffit d'appliquer la permutation finale au vecteur de 64 bits que l'on vient de calculer.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

1.5 Exemple de génération des K_i

Imaginons la clef **K** dont tous les bits sont égaux à 0 sauf les bits 57 et 63 (le premier bit est à l'indice 1, le second à l'indice 2 etc ...). **K** = "0000000000000000000000000000000000000000000000000000000010000010"
On applique à K les deux premières permutations. On obtient alors deux morceaux de 32 bits que je vais appeler GB et DB :

DB = "10000000000000000000000000000" (28 bits)
GB = "10000000000000000000000000000" (28 bits)

A la première ronde, DB et GB sont décalés (circulairement) de 1 bits vers la gauche.
On obtient après décalage :

DB1 = "00000000000000000000000000001"
GB1 = "00000000000000000000000000001"

Nous appliquons la dernière permutation sur
GB1 || DB1 = "00000000000000000000000000010000000000000000000000000001" (56 bits)
qui va nous permettre de passer d'une longueur de 56 bits à 48 bits.

K_0 = "000000010000000000000000000000000000000100000000" (48 bits)

Et c'est avec ce K_0 que nous allons chiffrer lors de la première ronde de l'algorithme.

Pour générer la clef K_1, GB1 et DB1 seront décalés de 1 bits vers la gauche (circulairement) et nous obtiendrons

DB2 = "00000000000000000000000000010" GB2 = "00000000000000000000000000010"

On applique la permutation sur GB2||DB2 pour trouver la clef K_2.

On remarquera que le décalage n'est pas toujours de 1 bit vers la gauche, mais dépend du tour de ronde.

Au 1er tour de ronde, on décale de 1.
Au 2eme tour de ronde, on décale de 1.
Au 3eme tour de ronde, on décale de 2.
Au 4eme tour de ronde, on décale de 2.
Au 5eme tour de ronde, on décale de 2.
Au 6eme tour de ronde, on décale de 2.
Au 7eme tour de ronde, on décale de 2.
Au 8eme tour de ronde, on décale de 2.
Au 9eme tour de ronde, on décale de 1.
Au 10eme tour de ronde, on décale de 2.
Au 11eme tour de ronde, on décale de 2.
Au 12eme tour de ronde, on décale de 2.
Au 13eme tour de ronde, on décale de 2.
Au 14eme tour de ronde, on décale de 2.
Au 15eme tour de ronde, on décale de 2.
Au 16eme tour de ronde, on décale de 1.

Ainsi, à la ronde 3, GB2 et DB2 seront décalés de 2.

DB3 = "00000000000000000000000001000" GB3 = "00000000000000000000000001000"

Ainsi de suite ...

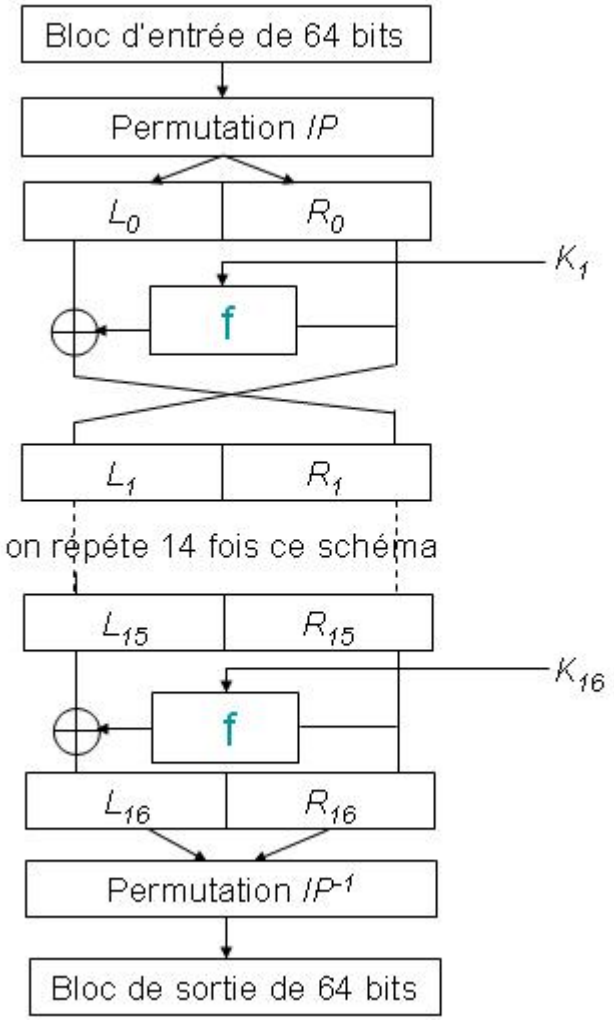
Voici un exemple pour tester votre programme :

M est le message que l'on veut chiffrer avec la Clef K

M = "0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111" (64 bits)

K = "00010011 00110100 01010111 01111001 10011011 10111100 11011111 1111000" (64 bits)

Chiffrement de M par K = "10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101"


Bloc d'entrée de 64 bits
Permutation IP
L0
R0
K1
f
L1
R1
on répéte 14 fois ce schéma
L15
R15
K16
f
L16
R16
Permutation IP-1
Bloc de sortie de 64 bits

1.6 Les SBOXES

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	