

---

 IMPLÉMENTATION DE L'ALGORITHME D.E.S
 

---

 Sebastien.Varrette@imag.fr
 

---

## 1 Objectifs

L'objectif de ce projet est la réalisation en binôme d'une implémentation du cryptosystème à clé secrète DES en C. Un rapport devra être réalisé, justifiant les choix d'implémentation et répondant aux quelques questions qui sont posés au cours de l'énoncé. Ce rapport sera envoyé par mail à l'adresse ci-dessus avec l'archive qui contiendra les sources du programme réalisé. Ensuite, une soutenance sera organisée où le binôme présentera le travail réalisé et les problèmes rencontrés.

Une grande importance sera apportée à la lisibilité des programmes et à leur documentation.

## 2 Présentation exhaustive du système D.E.S

Ce système de chiffrement à clef privée est le plus connu. Un cryptosystème permet à deux protagonistes, appelés traditionnellement Alice et Bob, de communiquer ensemble sur un canal peu sûr lorsqu'un opposant, Oscar, souhaite espionner cette conversation. Evidemment, Oscar ne doit pas comprendre les informations qui sont échangées.

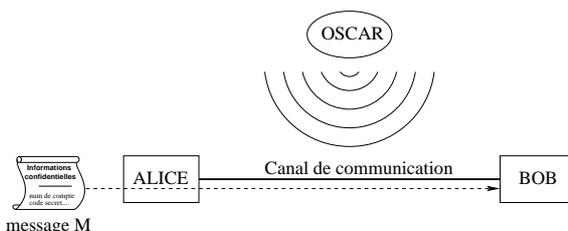


FIGURE 1 – Les protagonistes d'un cryptosystème

Pour un cryptosystème, on définit les expressions suivantes :

- **Texte clair** : information qu'Alice souhaite transmettre à Bob (Ex : texte en français, donnée numérique etc...)
- **Chiffrement** : processus de transformation d'un message  $M$  de telle manière à le rendre incompréhensible. Ce processus est basé sur une *fonction de chiffrement*  $E$  et permet de générer ainsi un **message chiffré**  $C = E(M)$
- **Déchiffrement** : processus de reconstruction du message clair à partir du message chiffré, basé sur une fonction de déchiffrement  $D$ .

On a donc  $D(C) = D(E(M)) = M$  ( $D$  et  $E$  sont injectives).

En pratique :  $E$  et  $D$  sont paramétrées par des clefs  $K_e$  et  $K_d$  (comme illustré

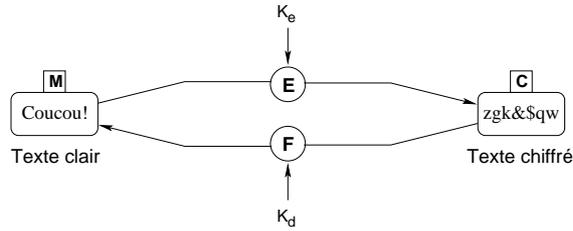


FIGURE 2 – Illustration du chiffrement/déchiffrement d’un texte clair

dans la figure 2) et sont liées par l’équation 1 :

$$\begin{cases} E_{K_e}(M) = C \\ D_{K_d}(C) = M \end{cases} \quad (1)$$

Le lien qui unit  $K_e$  et  $K_d$  définit deux grandes catégories de systèmes cryptographiques :

1. les systèmes à clef secrète (ou symétriques) ( $K_e = K_d = K$ ). C’est le cas du système D.E.S.
2. les systèmes à clef publique (ou asymétriques) ( $K_e \neq K_d$ ) comme par exemple le système R.S.A.

DES (Data Encryption Standard) fut adopté comme standard de chiffrement à clef secrète en 1977. Ce projet a donc pour objectif d’implémenter le chiffrement d’un bloc de texte clair de 64 bits par l’algorithme DES.

## 2.1 Algorithme général

Ce système fonctionne par blocs de texte clair de 64 bits en utilisant une clef de 56 bits<sup>1</sup>. Il permet d’obtenir ainsi des blocs de texte chiffré de 64 bits. D’une manière générale, l’algorithme se déroule en trois étapes :

1. soit  $x$  un bloc de texte clair de 64 bits. On lui applique une permutation initiale  $IP$  fixée pour obtenir une chaîne  $x_0$ . On a donc :

$$x_0 = IP(x) = L_0.R_0$$

où  $L_0$  contient les 32 premiers bits de  $x_0$  et  $R_0$  les 32 restants.

2. 16 itérations (ou 16 tours) d’une certaine fonction sont effectuées. On calcule  $L_i$  et  $R_i$ ,  $1 \leq i \leq 16$  suivant la règle :

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \end{cases} \quad (2)$$

La figure 3 présente un tour de chiffrement. La fonction  $f$  est une fonction à deux variables, l’une de 32 bits (correspondant à  $R_{i-1}$  à la  $i^{eme}$  itération) et l’autre de 48 bits (il s’agit de  $K_i$ ) dont le calcul sera explicité dans le §2.2. Les éléments  $K_i$  sont obtenus par diversification des bits de la clef initiale  $K$  (de 56 bits). Cette opération sera présentée dans le §2.3.

1. en fait, la clef compte 64 bits mais parmi lesquels 1 bit sur 8 est utilisé comme bit de parité

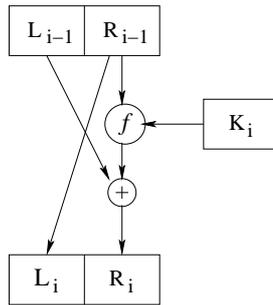


FIGURE 3 – Un tour de DES

3. La permutation inverse  $IP^{-1}$  est appliquée à  $R_{16}L_{16}$  pour obtenir un bloc de texte chiffré  $y = IP^{-1}(R_{16}L_{16})$  (à noter l'inversement de  $L_{16}$  et de  $R_{16}$ ).

Le même algorithme (à quelques légères nuances près) est utilisé pour déchiffrer. La sûreté du DES vient de la fonction  $f$  qui est en fait une substitution suivie d'une permutations. De plus, après 16 tours, le résultat est statistiquement "plat", c'est-à-dire que les caractéristiques générales du message source (la fréquence des caractères, le nombre d'espaces, ...) seront indétectables. De plus il dispose d'une caractéristique très importante pour éviter les attaques par analyse différentielle : une légère modification de la clef ou du texte à chiffrer provoque des changements importants dans le texte chiffré.

Les détails de la permutation initiale IP sont fournis dans le tableau 1

IP							
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

TABLE 1 – Permutation initiale IP

En numérotant les bits de 1 (bit de poids **fort**) à 64 (bit de poids **faible**) dans le calcul de  $y = IP(x)$ , le 58<sup>me</sup> bit de  $x$  est le premier de  $y$ , le 50<sup>me</sup> bit de  $x$  est le second de  $y$  etc...

Ce formalisme sera utilisé dans toute la suite de ce document.

De même, la permutation inverse  $IP^{-1}$  est donnée dans le tableau 2

Le gros avantage du DES est qu'il repose, tant pour le chiffrement que pour le déchiffrement, sur des opérations facilement implantables au niveau matériel, il est donc possible d'obtenir des taux de chiffrement très élevés, de l'ordre de 40Mo/s il y a une dizaine d'années, avec du matériel spécifique. Aujourd'hui,

IP <sup>-1</sup>							
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

TABLE 2 – Permutation finale IP<sup>-1</sup>

avec un puce spécifique bas de gamme<sup>2</sup> (de l'ordre de 60 euros), on arrive à des débits de l'ordre de 190 Mo/s. A noter que dans le cadre de ce projet à but pédagogique, l'efficacité n'est pas un critère primordial.

## 2.2 Détails de la fonction $f(A, J)$

On a vu que l'algorithme DES faisait appel à une fonction  $f$  prenant deux arguments :

1. une chaîne  $A$  de 32 bits (la partie droite du bloc à chiffrer) ;
2. une chaîne  $J$  de 48 bits (une clef diversifiée).

Le résultat de cette fonction est une chaîne de 32 bits.

Le calcul de  $f(A, J)$  se déroule en plusieurs étapes, illustrées dans la figure 4 :

1.  $A$  est augmentée en une chaîne de 48 bits par une *fonction d'expansion*  $E$ .  $E(A)$  est composé de tous les bits de  $A$  dans un certain ordre, et 16 d'entre eux apparaissent deux fois. La fonction d'expansion  $E$  est décrite dans la table 3

E						
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	

TABLE 3 – Fonction d'expansion E

2.  $B = E(A) \oplus J$  est calculé.  $B$  est ensuite découpé en 8 sous-chaînes consécutives de 6 bits chacune :  $B = B_1.B_2 \dots B_8$ .
3. L'étape suivante utilise 8 "boîtes-S"  $S_1, S_2, \dots, S_8$  qui sont des tableaux 4\*16 d'entiers compris entre 0 et 15.

---

2. voir par exemple <http://hifn.com/products/7955-7956.html>

Soit une sous-chaîne de six bits  $B_i = b_1.b_2.b_3.b_4.b_5.b_6$ .

On calcule une chaîne de quatre bits  $S_i(B_j)$  de la façon suivante :

- les bits  $b_1.b_6$  fournissent la représentation binaire de l'indice  $l$  de la ligne de  $S_i$  à considérer.  $0 \leq l \leq 3$ .
- les bits  $b_2.b_3.b_4.b_5$  fournissent la représentation binaire de l'indice  $c$  de la colonne de  $S_i$  à utiliser.  $0 \leq c \leq 15$ .
- l'intersection de la ligne  $l$  et de la colonne  $c$  de  $S_i$  fournit un entiers compris entre 0 et 15, donc une chaîne de 4 bits, qui sera le résultat  $C_i = S_i(B_i)$ . On calcule ainsi  $\{C_j = S_j(B_j)\}_{1 \leq j \leq 8}$ .

4. La chaîne  $C = C_1.C_2 \dots C_8$ , de longueur 32 bits, est alors réordonnée suivant une permutation fixée  $P$ . Le résultat  $P(C)$  définit  $f(A, J)$ .

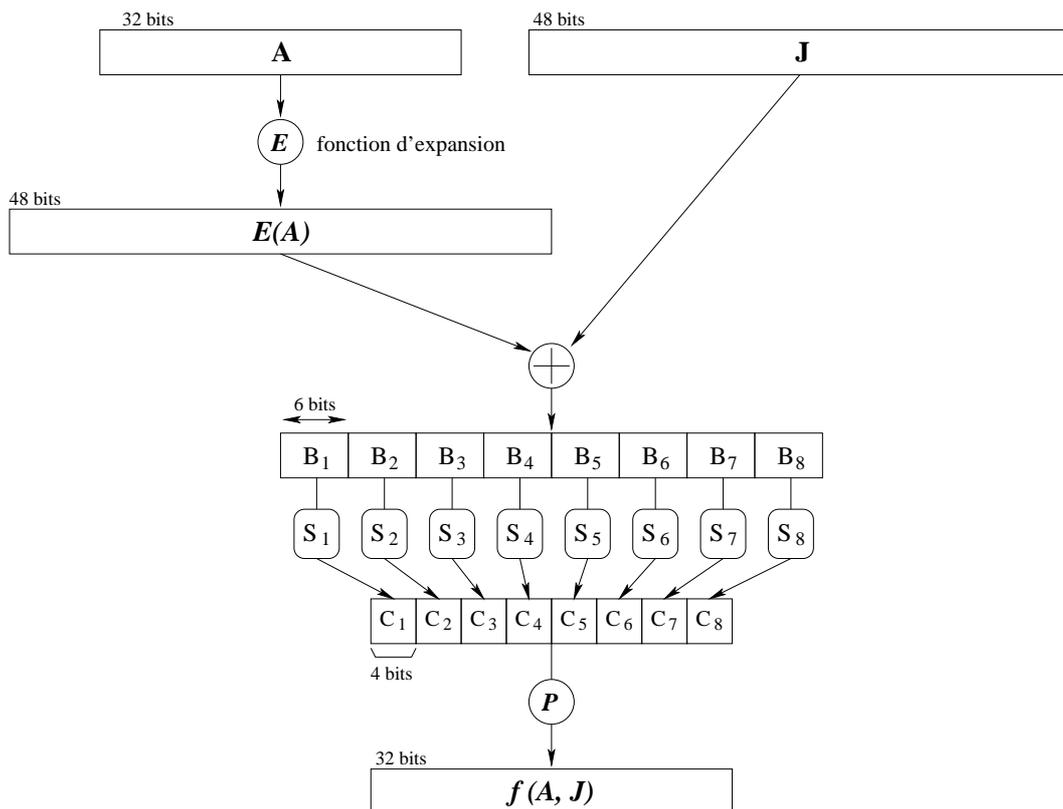


FIGURE 4 – La fonction  $f$  de DES

Nous présentons maintenant les détails des boîtes-S et de la permutation  $P$  :

$S_1$															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

$S_2$															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

$S_3$															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

$S_4$															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

$S_5$															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

$S_6$															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

$S_7$															
4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

$S_8$															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

$P$			
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

### 2.3 Détails de la diversification de la clef dans DES

On a vu que le second argument de la fonction  $f$  pour le tour  $i$  est une clef  $K_i$  extraite de la clef initiale  $K$ . Cette section détaille l'algorithme utilisé.

La clef  $K$  est une chaîne de 64 bits, dont 56 seulement définissent la clef. Les 8 autres bits, ceux situés en position 8,16,...,64 sont des bits de parité.

Donc si  $K = k_1 \dots k_8 \dots k_{64}$ , alors  $\forall i \in [1, 8]$  :

$$k_{8i} = \bigoplus_{j=1}^7 k_{8i-j}$$

(Selon les implémentations, on utilise alternative le complément à 1 de la formule précédente) Ainsi, une erreur (plus précisément un nombre impair d'erreurs) peut être détectée pour chaque groupe de 8 bits. Dans ce cas, le procédé de chiffrement ne peut être effectué. Ces bits de parités sont ignorés dans le procédé de diversification (la permutation PC-1 les éliminent directement). Vous pouvez si vous le souhaitez effectuer ce contrôle de parité mais on n'en tiendra pas compte dans la suite.

Voici l'algorithme utilisé pour calculer les clefs  $K_i$  utilisés pour chaque ronde  $i$  :

1. A partir de la clef  $K$  de 64 bits, on enlève les bits de parités et on ordonne les bits restants selon une permutation initiale PC-1. On note  $PC-1(K) = C_0.D_0$ , où  $C_0$  est composée des 28 premiers bits de  $PC-1(K)$  et  $D_0$  des 28 restants.
2. On calcule  $C_i, D_i$  et  $K_i$  ( $1 \leq i \leq 16$ ) de la façon suivante :

$$\begin{cases} C_i = LS_i(C_{i-1}) \\ D_i = LS_i(D_{i-1}) \\ K_i = PC-2(C_i.D_i) \end{cases} \quad (3)$$

Où  $LS_i$  est une rotation circulaire vers la gauche d'une ou deux positions selon la valeur de  $i$  :

- on décale d'une position si  $i \in \{1, 2, 9, 16\}$  ;
- sinon, on décale de deux positions.

PC-2 est une autre permutation des bits de la chaîne.

Le calcul de la diversification des clefs est illustré dans la figure 5.

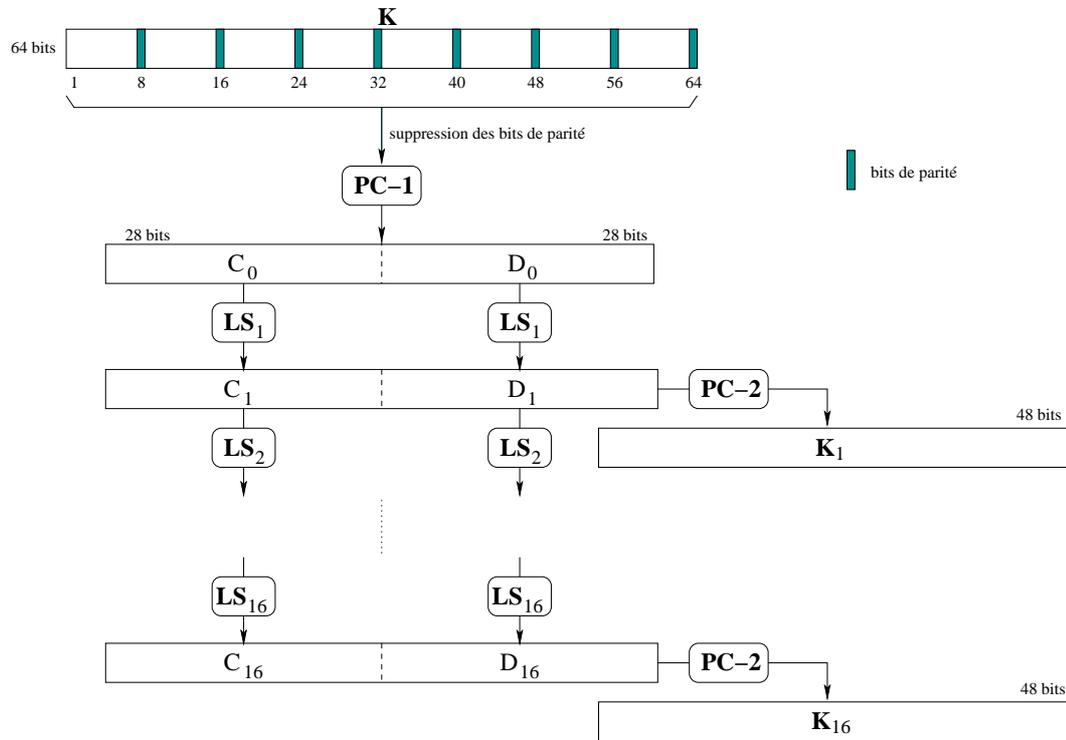


FIGURE 5 – La diversification de clef dans DES

Les permutations PC-1 et PC-2 utilisées dans la diversification des clefs sont :

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

### 3 Cahier des charges de l'exécutable à produire

La compilation des sources produites devra fournir un exécutable `des`. On se limitera au chiffrement d'un bloc de 64 bits. Les groupes qui le souhaitent pourront implémenter le chiffrement d'une chaîne de longueur quelconque mais ce n'est pas demandé.

Cet exécutable devra supporter les options suivantes :

- h : affichage de l'aide : nom des binômes, des options disponibles et de toute information jugée utile .

- k <key> : utilisation de la cle <key> donnee sous forme hexadecimale sur 64 bits.  
*Valeur par defaut* : 13345779 9BBCDF1 (c'est celle qui sera utilisée comme exemple tout au long de document)
- t <text> : le texte a chiffrer sous forme d'un bloc de 64 bits ecrit en hexadecimal.  
*Valeur par defaut* : 01234567 89ABCDEF
- r <nb\_ronde> : définit le nombre de rondes à effectuer.  
*Valeur par defaut* : 16.
- m <mode> : mode de chiffrement ('des' ou 'dea'). Dans le mode 'dea', on applique pas la permutation initiale IP, ni la permutation finale  $IP^{-1}$  (l'inversion finale est également omise).  
*Valeur par defaut* : **des**

(très utile) -v : verbose, affiche des informations supplementaires

Libre à vous également d'implémenter d'autres options (pour permettre par exemple le déchiffrement).

Pour vous faciliter la tâche, une première archive peut être récupérée à l'adresse <http://www-id.imag.fr/~svarrett/enseignements.html> dans la rubrique consacrée à ce cours. Pour décompresser l'archive :

```
> tar xvzf archive.tgz
```

On veillera à conserver l'organisation des répertoires :

- la racine contiendra notamment les fichiers sources .c.
- les fichiers d'en-tête (.h) seront à placer dans le répertoire **Include/**.
- le répertoire .obj/ contiendra les fichiers objets générés.

Initialement, cette archive est fonctionnelle. A vous de compléter cette archive à l'aide de vos nouveau fichiers.

### 3.1 Description des fichiers initialement fournis

Outre le fichier principal (**main.c**) qui vous montre comment gérer les arguments de la ligne de commande avec la librairie **getopt**, vous trouverez les fichiers suivants :

- **Include/ressources.h** : ce fichier contient toutes les définitions des différentes boites-S, des permutations statiques (IP,  $IP^{-1}$  etc...). On dit merci qui? :-)
- **Makefile** : fichier de configuration de la commande 'make' qui gère automatiquement la compilation des fichiers source. Vous pouvez ajouter autant de fichiers que vous le souhaitez, vous ne devriez pas avoir à retoucher ce Makefile normalement.
- **README** : fichier texte qui donne quelques informations.
- **TAGS** : ce fichier est généré à la compilation et est utilisé par certains éditeurs de texte avancés (ici emacs) pour naviguer dans les fichiers sources (par exemple depuis une définition du prototype d'une fonction dans **toto.h** au source de cette fontion dans **toto.c**). man **etags** pour plus d'informations.

## 4 Quelques conseils pratiques

- Pensez à bien commenter vos programmes. **Des fichiers non commentés ne seront pas corrigés!!!**
- Choisissez des noms explicites pour vos variables.
- Définissez rapidement les conventions de programmation que vous utiliserez : représentation des blocs à chiffrer, architecture des fichiers (n'hésiter pas à regrouper vos fonctions par module). N'oubliez pas de mentionner tout cela dans le rapport final, à remettre avec vos sources.
- **Vous travaillez en binôme!** Cela implique que vous devez vous répartir équitablement les tâches et que le travail se fait à deux. Vous pouvez bien entendu échanger vos idées entre groupes mais éviter de copier les programmes du groupe voisin la veille de la deadline!
- **Pensez au debuggage!** Rien n'est parfait du premier coup! Alors pensez à vous faciliter la tâche en créant une batterie de routines pour le débogage, notamment l'affichage de blocs en binaire. Vous pouvez aussi utiliser des debugger comme `gdb` (mode ligne de commande) ou encore `xxgdb` et `ddd` qui sont des outils graphiques. Une bonne note pour `valgrind` qui permet de détecter les fuites de mémoire<sup>3</sup>
- Organisations d'un travail efficace sur ce projet :
  1. Décider de la représentation des blocs (qui ont au maximum 64 bits) :
    - tableau de  $n$  bits? (préférer la convention ou `tab[0]` correspond au bit de poids fort)
    - tableau de 2 `unsigned long` (facilite la découpe en 2 sous-blocs)
    - un `unsigned long long` (si le compilateur le permet)
  2. Ecrire les routines de manipulation de blocs (notamment l'affichage binaire ou hexa et la création d'un bloc à partir d'une chaîne hexa).
  3. Ecrire la fonction de permutation et la tester en implémentant la diversification de clés.
  4. Ecrire la fonctions calculant la sortie des S-Box puis  $f(A, J)$
  5. Enfin, réaliser la fonction de chiffrement DES.

L'archive fournie n'est pas une bible : si son organisation ou la programmation ne vous convenait pas, libre à vous de l'organiser comme vous le souhaitez. Il faut simplement respecter le cahier des charges défini au §3.

## 5 Aide au débogage

Afin de vous aidez dans le débogage de votre programme, ce paragraphe détaille toutes les étapes du chiffrement du bloc de texte clair (hétéradécimal) 01234567 89ABCDEF avec la clef (hétéradécimale également) 13345779 9BBCDFF1.

### 5.1 Etape de diversification des clefs

$C_0$	=	1111 00001100 11001010 10101111
$D_0$	=	0101 01010110 01100111 10001111

3. Voir <http://developer.kde.org/~sewardj/docs-2.1.2/manual.html>

$C_1$	= 1110 00011001 10010101 01011111
$D_1$	= 1010 10101100 11001111 00011110
$K_1$	= 00011011 00000010 11101111 11111100 01110000 01110010 = 1B02EF FC7072
$C_2$	= 1100 00110011 00101010 10111111
$D_2$	= 0101 01011001 10011110 00111101
$K_2$	= 01111001 10101110 11011001 11011011 11001001 11100101 = 79AED9 DBC9E5
$C_3$	= 0000 11001100 10101010 11111111
$D_3$	= 0101 01100110 01111000 11110101
$K_3$	= 01010101 11111100 10001010 01000010 11001111 10011001 = 55FC8A 42CF99
$C_4$	= 0011 00110010 10101011 11111100
$D_4$	= 0101 10011001 11100011 11010101
$K_4$	= 01110010 10101101 11010110 11011011 00110101 00011101 = 72ADD6 DB351D
$C_5$	= 1100 11001010 10101111 1111 0000
$D_5$	= 0110 01100111 10001111 0101 0101
$K_5$	= 01111100 11101100 00000111 11101011 01010011 10101000 = 7CEC07 EB53A8
$C_6$	= 0011 00101010 10111111 1100 0011
$D_6$	= 1001 10011110 00111101 0101 0101
$K_6$	= 01100011 10100101 00111110 01010000 01111011 00101111 = 63A53E 507B2F
$C_7$	= 1100 10101010 11111111 0000 1100
$D_7$	= 0110 01111000 11110101 0101 0110
$K_7$	= 11101100 10000100 10110111 11110110 00011000 10111100 = EC84B7 F618BC
$C_8$	= 0010 10101011 11111100 0011 0011
$D_8$	= 1001 11100011 11010101 0101 1001
$K_8$	= 11110111 10001010 00111010 11000001 00111011 11111011 = F78A3A C13BFB
$C_9$	= 0101 01010111 1111 1000 0110 0110
$D_9$	= 0011 11000111 1010 1010 1011 0011
$K_9$	= 11100000 11011011 11101011 11101101 11100111 10000001 = E0DBEB EDE781
$C_{10}$	= 0101 01011111 11100001 10011001
$D_{10}$	= 1111 00011110 10101010 11001100
$K_{10}$	= 10110001 11110011 01000111 10111010 01000110 01001111 = B1F347 BA464F
$C_{11}$	= 0101 01111111 10000110 01100101
$D_{11}$	= 1100 01111010 10101011 00110011
$K_{11}$	= 00100001 01011111 11010011 11011110 11010011 10000110 = 215FD3 DED386
$C_{12}$	= 0101 11111110 00011001 10010101
$D_{12}$	= 0001 11101010 10101100 11001111
$K_{12}$	= 01110101 01110001 11110101 10010100 01100111 11101001 = 7571F5 9467E9
$C_{13}$	= 0111 11111000 01100110 01010101
$D_{13}$	= 0111 10101010 10110011 00111100
$K_{13}$	= 10010111 11000101 11010001 11111010 10111010 01000001 = 97C5D1 FABA41
$C_{14}$	= 1111 11100001 10011001 01010101
$D_{14}$	= 1110 10101010 11001100 11110001
$K_{14}$	= 01011111 01000011 10110111 11110010 11100111 00111010 = 5F43B7 F2E73A
$C_{15}$	= 1111 10000110 01100101 01010111
$D_{15}$	= 1010 10101011 00110011 11000111
$K_{15}$	= 10111111 10010001 10001101 00111101 00111111 00001010 = BF918D 3D3F0A

$C_{16}$	=	1111 00001100 11001010 10101111
$D_{16}$	=	0101 01010110 01100111 10001111
$K_{16}$	=	11001011 00111101 10001011 00001110 00010111 11110101 = CB3D8B 0E17F5

## 5.2 Détail des rondes

L0	=	11001100 00000000 11001100 11111111	=	CC00CCFF	
R0	=	11110000 10101010 11110000 10101010	=	F0AAF0AA	
	E(A)	=	01111010 00010101 01010101 01111010 00010101 01010101	=	7A15 557A1555
	J	=	00011011 00000010 11101111 11111100 01110000 01110010	=	1B02 EFFC7072
	E(A) <sup>~</sup> J	=	01100001 00010111 10111010 10000110 01100101 00100111	=	6117 BA866527
	C	=	01011100 10000010 10110101 10010111	=	5C82B597
L1	=	11110000 10101010 11110000 10101010	=	F0AAF0AA	
R1	=	11101111 01001010 01100101 01000100	=	EF4A6544	
	E(A)	=	01110101 11101010 01010100 00110000 10101010 00001001	=	75EA 5430AA09
	J	=	01111001 10101110 11011001 11011011 11001001 11100101	=	79AE D9DBC9E5
	E(A) <sup>~</sup> J	=	00001100 01000100 10001101 11101011 01100011 11101100	=	0C44 8DEB63EC
	C	=	11111000 11010000 00111010 10101110	=	F8D03AAE
L2	=	11101111 01001010 01100101 01000100	=	EF4A6544	
R2	=	11001100 00000001 01110111 00001001	=	CC017709	
	E(A)	=	11100101 10000000 00000010 10111010 11101000 01010011	=	E580 02BAE853
	J	=	01010101 11111100 10001010 01000010 11001111 10011001	=	55FC 8A42CF99
	E(A) <sup>~</sup> J	=	10110000 01111100 10001000 11111000 00100111 11001010	=	B07C 88F827CA
	C	=	00100111 00010000 11100001 01101111	=	2710E16F
L3	=	11001100 00000001 01110111 00001001	=	CC017709	
R3	=	10100010 01011100 00001011 11110100	=	A25C0BF4	
	E(A)	=	01010000 01000010 11111000 00000101 01111111 10101001	=	5042 F8057FA9
	J	=	01110010 10101101 11010110 11011011 00110101 00011101	=	72AD D6DB351D
	E(A) <sup>~</sup> J	=	00100010 11101111 00101110 11011110 01001010 10110100	=	22EF 2EDE4AB4
	C	=	00100001 11101101 10011111 00111010	=	21ED9F3A
L4	=	10100010 01011100 00001011 11110100	=	A25C0BF4	
R4	=	01110111 00100010 00000000 01000101	=	77220045	
	E(A)	=	10111010 11101001 00000100 00000000 00000010 00001010	=	BAE9 040020A
	J	=	01111100 11101100 00000111 11101011 01010011 10101000	=	7CEC 07EB53A8
	E(A) <sup>~</sup> J	=	11000110 00000101 00000011 11101011 01010001 10100010	=	C605 03EB51A2
	C	=	01010000 11001000 00110001 11101011	=	50C831EB
L5	=	01110111 00100010 00000000 01000101	=	77220045	
R5	=	10001010 01001111 10100110 00110111	=	8A4FA637	
	E(A)	=	11000101 01000010 01011111 11010000 11000001 10101111	=	C542 5FD0C1AF
	J	=	01100011 10100101 00111110 01010000 01111011 00101111	=	63A5 3E507B2F
	E(A) <sup>~</sup> J	=	10100110 11100111 01100001 10000000 10111010 10000000	=	A6E7 6180BA80
	C	=	01000001 11110011 01001100 00111101	=	41F34C3D
L6	=	10001010 01001111 10100110 00110111	=	8A4FA637	
R6	=	11101001 01100111 11001101 01101001	=	E967CD69	
	E(A)	=	11110101 00101011 00001111 11100101 10101011 01010011	=	F52B OFE5AB53
	J	=	11101100 10000100 10110111 11110110 00011000 10111100	=	EC84 B7F618BC
	E(A) <sup>~</sup> J	=	00011001 10101111 10111000 00010011 10110011 11101111	=	19AF B813B3EF
	C	=	00010000 01110101 01000000 10101101	=	107540AD
L7	=	11101001 01100111 11001101 01101001	=	E967CD69	
R7	=	00000110 01001010 10111010 00010000	=	064ABA10	

	E(A)	= 00000000	11000010	01010101	01011111	01000000	10100000	= 00C2	555F40A0
	J	= 11110111	10001010	00111010	11000001	00111011	11111011	= F78A	3AC13BFB
	E(A)^J	= 11110111	01001000	01101111	10011110	01111011	01011011	= F748	6F9E7B5B
	C	= 01101100	00011000	01111100	10101110			= 6C187CAE	
L8	=	00000110	01001010	10111010	00010000			= 064ABA10	
R8	=	11010101	01101001	01001011	10010000			= D5694B90	
	E(A)	= 01101010	10101011	01010010	10100101	01111100	10100001	= 6AAB	52A57CA1
	J	= 11100000	11011011	11101011	11101101	11100111	10000001	= E0DB	EBEDE781
	E(A)^J	= 10001010	01110000	10111001	01001000	10011011	00100000	= 8A70	B9489B20
	C	= 00010001	00001100	01010111	01110111			= 110C5777	
L9	=	11010101	01101001	01001011	10010000			= D5694B90	
R9	=	00100100	01111100	11000110	01111010			= 247CC67A	
	E(A)	= 00010000	10000011	11111001	01100000	11000011	11110100	= 1083	F960C3F4
	J	= 10110001	11110011	01000111	10111010	01000110	01001111	= B1F3	47BA464F
	E(A)^J	= 10100001	01110000	10111110	11011010	10000101	10111011	= A170	BEDA85BB
	C	= 11011010	00000100	01010010	01110101			= DA045275	
L10	=	00100100	01111100	11000110	01111010			= 247CC67A	
R10	=	10110111	11010101	11010111	10110010			= B7D5D7B2	
	E(A)	= 01011010	11111110	10101011	11101010	11111101	10100101	= 5AFE	ABEAFDA5
	J	= 00100001	01011111	11010011	11011110	11010011	10000110	= 215F	D3DED386
	E(A)^J	= 01111011	10100001	01111000	00110100	00101110	00100011	= 7BA1	78342E23
	C	= 01110011	00000101	11010001	00000001			= 7305D101	
L11	=	10110111	11010101	11010111	10110010			= B7D5D7B2	
R11	=	11000101	01111000	00111100	01111000			= C5783C78	
	E(A)	= 01100000	10101011	11110000	00011111	10000011	11110001	= 60AB	F01F83F1
	J	= 01110101	01110001	11110101	10010100	01100111	11101001	= 7571	F59467E9
	E(A)^J	= 00010101	11011010	00000101	10001011	11100100	00011000	= 15DA	058BE418
	C	= 01111011	10001011	00100110	00110101			= 7B8B2635	
L12	=	11000101	01111000	00111100	01111000			= C5783C78	
R12	=	01110101	10111101	00011000	01011000			= 75BD1858	
	E(A)	= 00111010	10111101	11111010	10001111	00000010	11110000	= 3ABD	FA8F02F0
	J	= 10010111	11000101	11010001	11111010	10111010	01000001	= 97C5	D1FABA41
	E(A)^J	= 10101101	01111000	00101011	01110101	10111000	10110001	= AD78	2B75B8B1
	C	= 10011010	11010001	10001011	01001111			= 9AD18B4F	
L13	=	01110101	10111101	00011000	01011000			= 75BD1858	
R13	=	00011000	11000011	00010101	01011010			= 18C3155A	
	E(A)	= 00001111	00010110	00000110	10001010	10101010	11110100	= 0F16	068AAAF4
	J	= 01011111	01000011	10110111	11110010	11100111	00111010	= 5F43	B7F2E73A
	E(A)^J	= 01010000	01010101	10110001	01111000	01001101	11001110	= 5055	B1784DCE
	C	= 01100100	01111001	10011010	11110001			= 64799AF1	
L14	=	00011000	11000011	00010101	01011010			= 18C3155A	
R14	=	11000010	10001100	10010110	00001101			= C28C960D	
	E(A)	= 11100000	01010100	01011001	01001010	11000000	01011011	= E054	594AC05B
	J	= 10111111	10010001	10001101	00111101	00111111	00001010	= BF91	8D3D3F0A
	E(A)^J	= 01011111	11000101	11010100	01110111	11111111	01010001	= 5FC5	D477FF51
	C	= 10110010	11101000	10001101	00111100			= B2E88D3C	
L15	=	11000010	10001100	10010110	00001101			= C28C960D	
R15	=	01000011	01000010	00110010	00110100			= 43423234	
	E(A)	= 00100000	01101010	00000100	00011010	01000001	10101000	= 206A	041A41A8
	J	= 11001011	00111101	10001011	00001110	00010111	11110101	= CB3D	8B0E17F5
	E(A)^J	= 11101011	01010111	10001111	00010100	01010110	01011101	= EB57	8F14565D
	C	= 10100111	10000011	00100100	00101001			= A7832429	

---

L16	=	01000011	01000010	00110010	00110100	=	43423234
R16	=	00001010	01001100	11011001	10010101	=	0A4CD995

---

### 5.3 Texte chiffré (mode DES)

On obtient pour ce texte le bloc résultat 85E81354 0F0AB405

### 5.4 Texte chiffré (mode DEA)

On obtient pour ce texte le bloc résultat 8E5907DC 0C465F03

### 5.5 Autres exemples de chiffrement

En utilisant toujours la clef K = 13345779 9BBCDFF1

Texte Clair	--	Texte chiffré
748502CD38451097	->	D7F1A01A2E0B7AB7
3874756438451097	->	87449A10B5DFF4E9
486911026ACDFF31	->	6E3BAA414F29713B

Avec la clef K = 1234567890ABCDEF (attention : cette clé ne passera pas le contrôle de parité!) :

FFFFFFFFFFFFFFFF -> EB90BD2A6F9D3F12