UN EXEMPLE D'ALGORITHME MIXTE : LE "DES"

(dernière version définie dans FIPS[1] 46-3 de 1999)

Le "DES" ou « **Data Encryption Standard** » combine des transpositions et des substitutions pour former un "code produit" ("product cipher") nommé également "**algorithme mixte**". Transpositions et substitutions utilisées seules se révèlent fragiles c'est-à-dire "facilement brisables". Associées au sein d'un algorithme, elles peuvent toutefois réaliser un code relativement sûr. C'est ce que démontra R. SHANNON qui introduisit en 1949 cette idée de « mixage » de différents algorithmes symétriques.

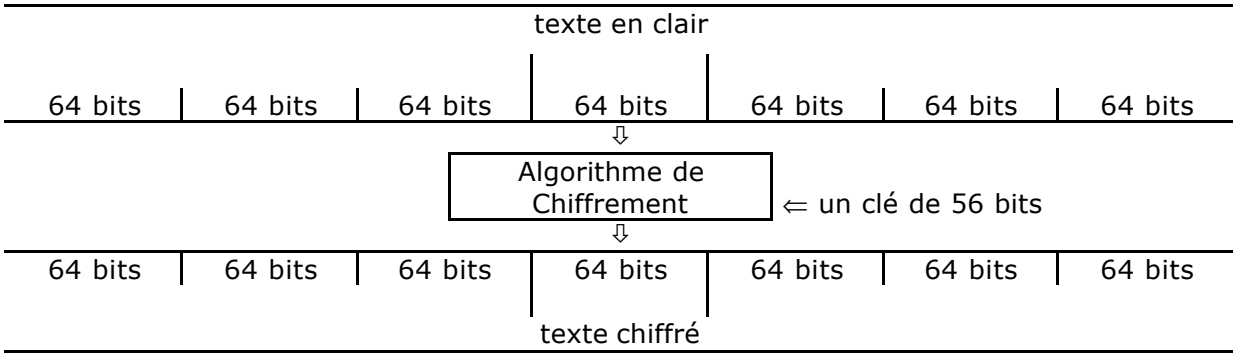
> *"Le DES est le résultat d'un travail d'amélioration d'algorithmes de codage conventionnels et en tant que tel s'inscrit directement dans une tradition historique"*[2].

Les « DES » est également connu sous le nom de « **DEA** » ou « **Data Encryption Algorithm** » norme de l'ANSI (ANSI X3.92) et sous le nom de « **DEA-1** » pour la norme « ISO ».

Description

Le message à chiffrer est **transcrit en binaire** grâce à un code convenu (code "ASCII" ou "EBCDIC", par exemple).

Il est ensuite **découpé en blocs de 64 bits**. Chacun de ces blocs subira le même traitement qui consiste en un chiffrement utilisant une **clé de 56 bits**, algorithme débouchant sur un **bloc chiffré de 64 bits**.



Le DES finalement n'effectue qu'une simple opération de transposition dans l'ensemble des 2^{64} messages possibles. Le nombre de permutations possibles s'élèvent tout de même à **(2^{64}) !** ...

1 FIPS : « Federal Information Processing Standards »
2 « Introduction aux méthodes de la cryptologie », B. Beckett, Masson 1990.

La première opération de ce traitement est une **transposition**, dite "**transposition ou permutation initiale**" connue sous le nom de « **IP** » (« Initial Permutation »), transposition qui mélange suivant une liste de transposition les 64 bits du bloc traité.

Permutation « 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**

Cette permutation initiale déplace le bit 58 à la position 1, le bit 50 à la position 2, …, le bit 1 à la position 40…

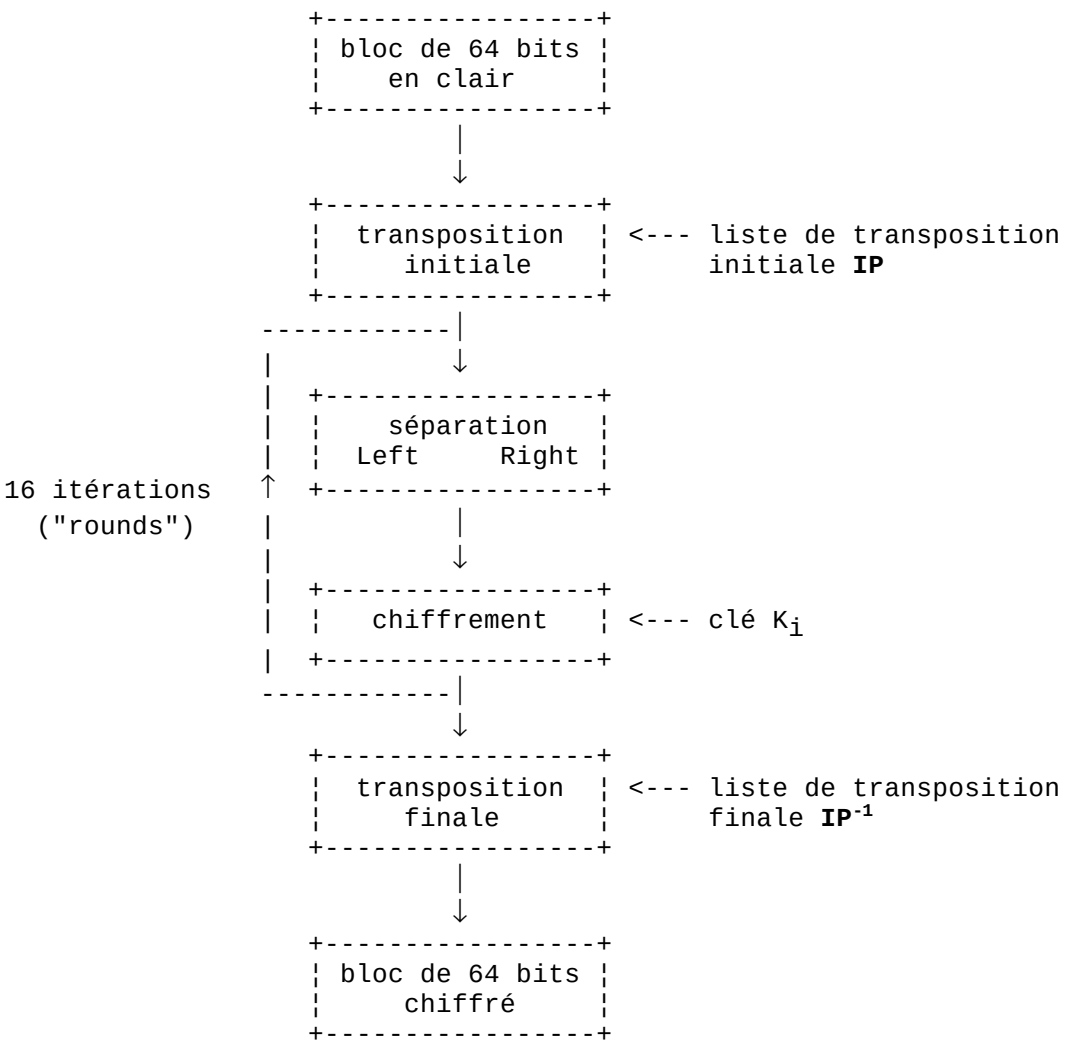
La nouvelle disposition des 64 bits est ensuite **partagée en deux parties**, une **partie gauche de 32 bits** et une **partie droite de 32 bits**. Ces deux parties subiront une **opération de chiffrement avec clé** et ce, **16 fois de suite**. Les 32 bits de gauche et les 32 bits de droite, refaisant une unité de 64 bits, subiront une **transposition finale** dite « $\mathbf{IP^{-1}}$ », nouveau mélange de ces 64 bits, en suivant une liste de transposition "inverse" de la liste initiale.

Permutation « IP^{-1} »

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**

Cette permutation finale déplace le bit 40 à la position 1, …, le bit 1 à la position 58, le bit 2 à la position 50...

> *"L'approche du DES est de combiner 'confusion' (substitution) et 'diffusion' (transposition) de façon à rendre la complexité maximale".*

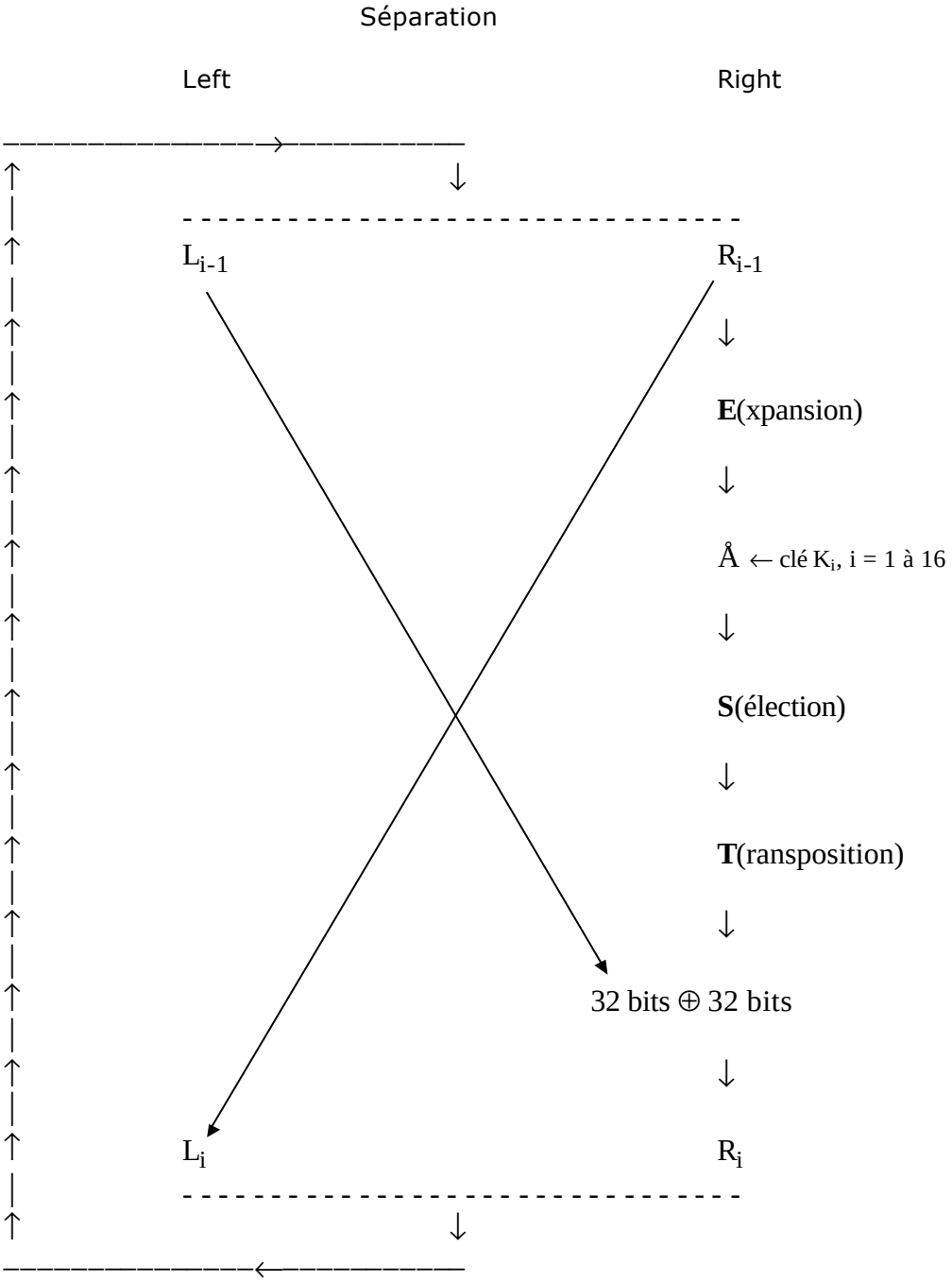


Les transpositions et les substitutions sont fixes et publiques. **La sécurité tout entière repose sur la clé**. Comportant 56 bits, celle-ci peut prendre 2^{56} valeurs possibles soit environ $7{,}2\ 10^{16}$ valeurs différentes. Une fois choisie cette clé est **utilisée à la fois au chiffrement et au déchiffrement**.

Le chiffrement central consistera en :

- une **fonction d'expansion** "**E**" qui est une transposition avec expansion,
- une **fonction de sélection** "**S**" qui est une substitution non linéaire,
- et en une **fonction de transposition** "**T**".

Voici résumé le processus de cette opération centrale de chiffrement :



Ce processus interne du DES fait de celui-ci un algorithme dit de "FEISTEL" ("feistel cipher") car il opère à chaque itération sur la moitié du texte.

La **Fonction d'Expansion**, « **E** », permutation expansive, utilise une liste de 48 valeurs et travaille sur un bloc de 32 bits. On obtient ainsi un bloc de 48 bits avec répétition de certains de ces bits (valeurs soulignées).

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**

Exemple [3]

Soit U, les 64 bits de données en clair après codage en binaire :
0101 1110 0111 0100 1111 1000 1010 0000 0111 1110 1100 0010 0001 1001 0011 0011
nous avons donc sur 32 bits, la partie droite : R_0 = 0010 1100 1001 1110 0101 0101 1011 0001
Comme la liste associée à la fonction d'expansion est :

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

nous obtenons sur 48 bits : X = 1001 0101 1001 0100 1111 1100 0010 1010 1011 1101 1010 0010

Calcul de la clé K_i

La clé de l'itération **i** est déduite de celle de l'itération **i-1** par un certain nombre d'opérations. La **"clé racine K'"** est composée de **56 bits**. En fait, elle est extraite d'un bloc de **64 bits** en enlevant les bits 8, 16, 24, 32, 40, 48, 56 et 64 (fins d'"octet"). Ceux-ci peuvent servir d'ailleurs de contrôle de parité pour les bits restants. Ces 56 bits restants sont partagés en 2 blocs, de 28 bits chacun et chaque bloc subit une permutation appelée « **PC - 1** ». En fait, on utilise une liste de permutation ne contenant pas les rangs 8, 16, 24, 32, 40, 48, 56 et 64. Une table de transposition **mélange les 64 bits totaux et en supprime 8**.

3 « La sécurité des réseaux, méthodes et techniques », J.-M. Lamère, Y. Leroux, J. Tourly, Dunod Informatique, Paris 1987, de p. 266 à p. 269.

Exemple [4]

Clé initiale, bloc de **64 bits** :

0000 000~~0~~	0000 000~~0~~	0000 000~~0~~	0000 000~~0~~
0000 000~~0~~	0000 001~~0~~	1000 000~~0~~	1000 000~~0~~

On enlève les 8 derniers bits (caractères barrés) en effectuant une permutation, basée sur deux listes de 28 nombres :

Permutation « PC - 1 »

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**
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**

Exemple [5]

On obtient alors une nouvelle clé sur **56 bits** :

1100 0000 0000 0000 0000 0000 0000 0010 0000 0000 0000 0000 0000 0000 0000

Une autre liste donne le **nombre de décalage circulaires à gauche** à effectuer sur ces deux blocs de 28 bits pour chaque itération. On décale de un bit pour les itérations de rang 1, 2, 9 ou 16 et de 2 bits pour les autres itérations. Cette liste donne le **nombre de décalage à gauche** pour les **16 itérations** (LS_i) :

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	**1**	**2**	**2**	**2**	**2**	**2**	**2**	**1**	**2**	**2**	**2**	**2**	**2**	**2**	**1**

Exemple [6]

Pour la première itération, nous obtenons donc après **1** décalage à gauche de chacune des deux parties :

1000 0000 0000 0000 0000 0000 0001 0100 0000 0000 0000 0000 0000 0000 0000

Une dernière permutation, dite « **PC - 2** », mélange et enlève 8 bits en fonction de la liste suivante :

4 idem, p. 266.
5 idem, p. 268.
6 idem, p. 268.

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**

Exemple[7]

Nous obtenons donc la clé K_1 de 48 bits :

0000 1001 0000 0000 0000 0000 0000 0010 0000 0000 0000 0000

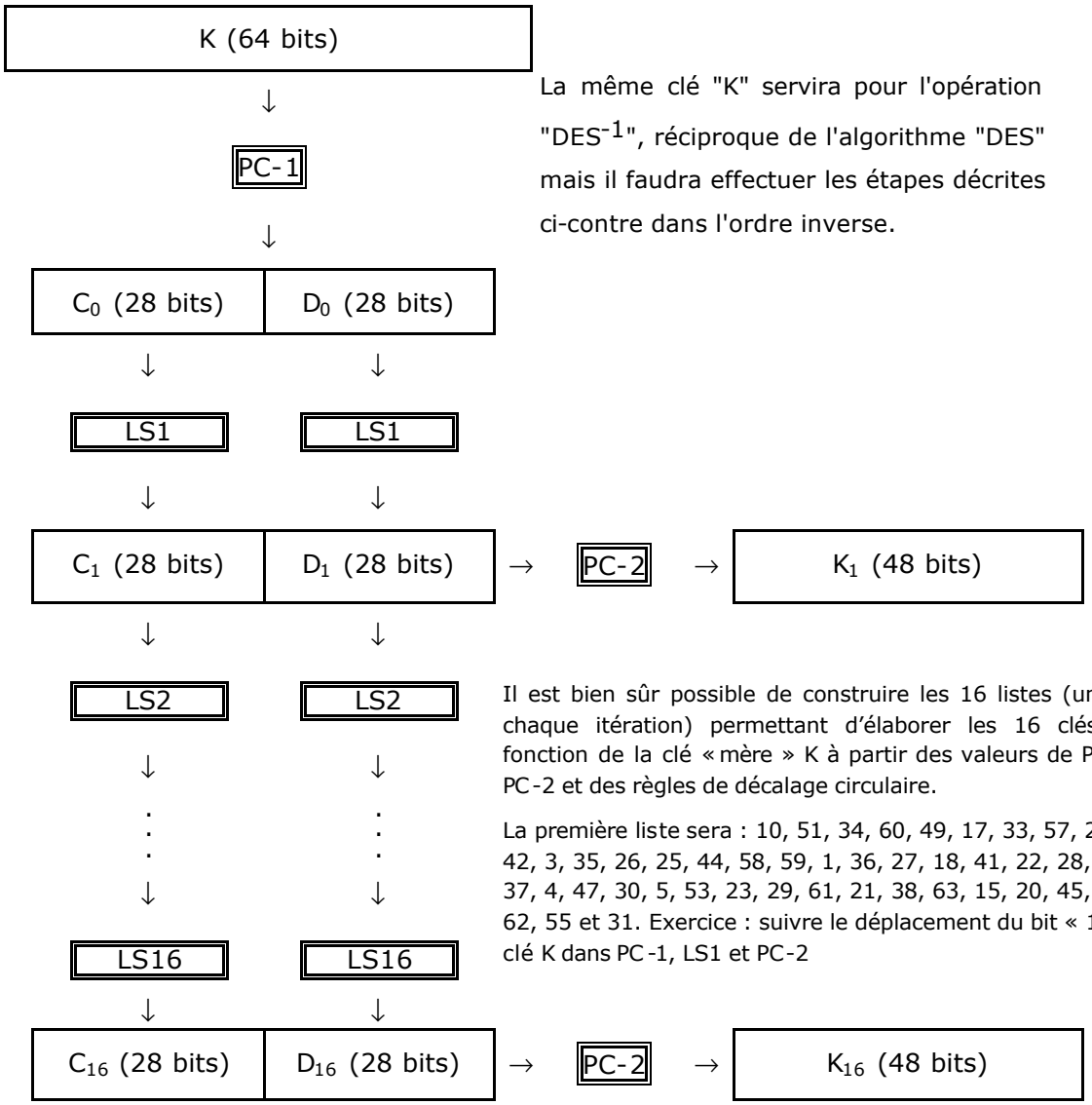
Cet algorithme est appliqué à chaque itération. Pour calculer K_2, nous repartons de la valeur :

1000 0000 0000 0000 0000 0000 0001 0100 0000 0000 0000 0000 0000

donc de la suite de 56 bits **avant la permutation**. C'est à cette valeur que sera appliqué le décalage circulaire suivant (itération 2 : décalage de 1 à gauche). Il en sera de même à chaque fois.

La clé est donc finalement composée de **48 bits**.

7 idem, p. 269..



La même clé "K" servira pour l'opération "DES^{-1}", réciproque de l'algorithme "DES" mais il faudra effectuer les étapes décrites ci-contre dans l'ordre inverse.

Il est bien sûr possible de construire les 16 listes (une pour chaque itération) permettant d'élaborer les 16 clés K_i en fonction de la clé « mère » K à partir des valeurs de PC-1, de PC-2 et des règles de décalage circulaire.

La première liste sera : 10, 51, 34, 60, 49, 17, 33, 57, 2, 9, 19, 42, 3, 35, 26, 25, 44, 58, 59, 1, 36, 27, 18, 41, 22, 28, 39, 54, 37, 4, 47, 30, 5, 53, 23, 29, 61, 21, 38, 63, 15, 20, 45, 14, 13, 62, 55 et 31. Exercice : suivre le déplacement du bit « 1 » de la clé K dans PC-1, LS1 et PC-2

x x x

La **Fonction de Sélection** "$S_{i,\ i = 1 \text{ à } 8}$" décomposera les **6 bits d'entrée** en **4 bits de positionnement "colonne"**, les bits 2 à 5 et en **2 bits de positionnement "ligne"**, les bits 1 et 6. Ces deux coordonnées permettront de cibler une valeur dans la matrice associée à "$S_{i,\ i = 1 \text{ à } 8}$", valeur qui sera codée en binaire sur **4 bits**.

L'analyse des 8 matrices des fonctions « S_i » permet d'appréhender cette opération comme une permutation avec un choix parmi quatre permutations, choix « décidé » par les deux autres bits.

Exemple [8]

En entrée de S, nous avons : 100111 001001 010011 111100 001010 001011 110110 100010
résultat du ⊕ entre X et K_1.

100111 : bloc d'entrée de S_1 et S_1 : b_1, b_6 ---> N_l

b_2, b_3, b_4, b_5 ---> N_c

$N_l = 3$, $N_c = 3$

matrice de S_1 (il existe 7 autres matrices analogues de S_2 à S_8) :

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

bloc de sortie de S_1 : 2 en binaire soit 0010

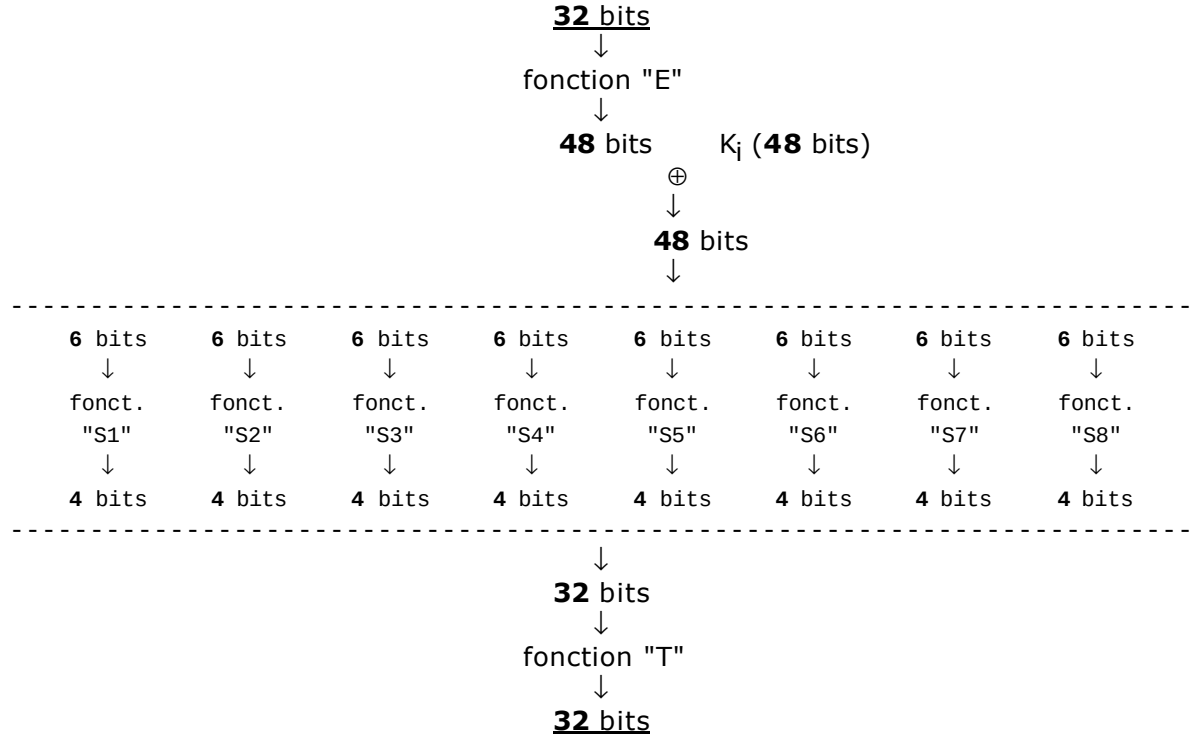
La **Fonction de Permutation** ou de Transposition, nommée « **P** », est une fonction classique de mélange des bits grâce à une liste de 32 valeurs :

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**

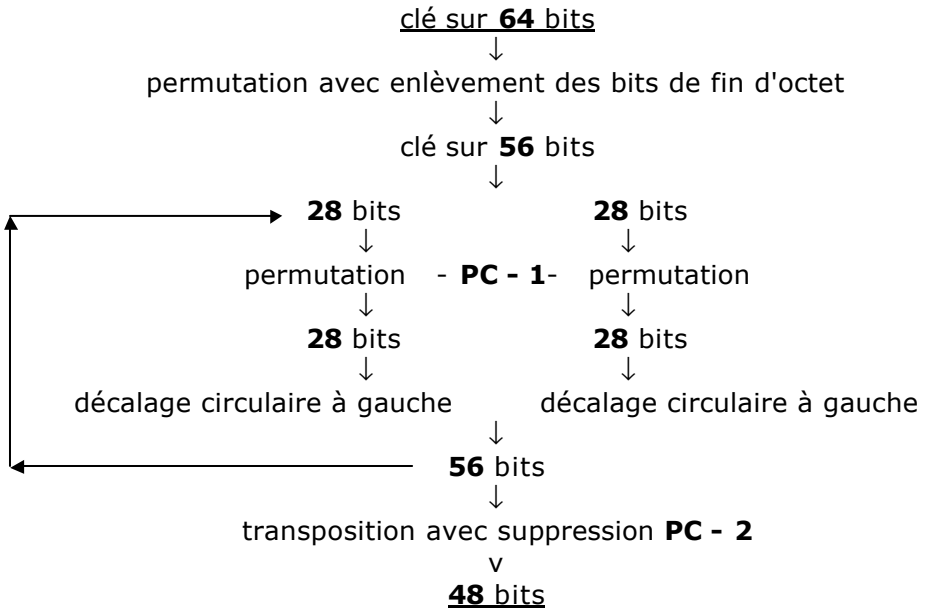
8 idem p. 269-270.

Synthèses sur les diverses longueurs…

1) Données



2) Clé(s)



Fonction S1

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

Fonction S2

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

Fonction S3

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	12	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

Fonction S4

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

Fonction S5

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

Fonction S6

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

Fonction S7

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

Fonction S8

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

Quelques réflexions...

La fonction de Sélection est la seule fonction non-linéaire du DES. C'est donc elle qui est le cœur du système puisque la cryptanalyse nous démontre qu'il est plus facile de « casser » des chiffrements linéaires que ceux qui ne le sont pas. Les critères de construction de cette fonction c'est-à-dire l'établissement des 7 tables, ne semblent pas tous connus. Les tables originelles proposées par IBM ne furent pas retenues par l'administration américaine. Certains affirmèrent ainsi que la « NSA » (« National Security Agency ») pouvait briser les textes chiffrés grâce à des « **trappes** » cachées dans cette fonction. Il est possible que la NSA redoutait de la part de IBM des trappes dans les fonctions "S"...

Une autre critique du DES porte sur la taille trop petite du champ de la clé, $\mathbf{2^{56}}$ nombre égal tout de même à **72 057 594 037 927 936**... Des machines actuelles, par une recherche systématique, en disposant du texte en clair et du texte chiffré pourraient découvrir en quelques jours la valeur de la clé utilisée.

Une des particularité notable du DES est sa **propriété de complémentation**. Celle-ci peut se traduire par :

$$C = E_K(T) \Leftrightarrow \overline{C} = E_{\overline{K}}(\overline{T})$$

Cette particularité peut être une faiblesse dans le cadre d'une attaque à texte clair choisi (voir le chapitre consacré à la cryptanalyse).

Il faut noter également l'existence de **six clés faibles** :

"00 00 00 00 00 00 00 00", "01 01 01 01 01 01 01 01", "FE FE FE FE FE FE FE FE",
"1F 1F 1F 1F 1F 1F 1F 1F", "E0 E0 E0 E0 E0 E0 E0 E0" et "FF FF FF FF FF FF FF FF".

pour lesquelles le DES est une **involution** c'est-à-dire que le chiffrement et le déchiffrement sont dans ce cas là strictement équivalents, les clés K_1 à K_{16} étant identiques dans les deux processus autrement dit que : $\mathbf{E_K(E_K(T)) = T}$. Ces clés sont également nommées « **clés palindromes** ».

Qui plus est, chacune de ces clés présentent $\mathbf{2^{32}}$ **points fixes**, c'est-à-dire des blocs de 64 bits tels que $\mathbf{E_K(T_i) = T_i}$ (mis en évidence par COPPERSMITH en 1985).

Il est également possible d'isoler **6 paires de clés** (les « dual keys » du NBS) qui seront dites « **semi-faibles** » qui présentent, pour chaque couple, un ensemble de clés

dérivées identiques mais d'ordre inverse, autrement dit 6 paires (K,L) telles que $\mathbf{E_K(E_L(T) = T}$. Ce sont les paires suivantes :

01 FE 01 FE 01 FE 01 FE et FE 01 FE 01 FE 01 FE 01,

1F E0 1F E0 1F E0 1F E0 et E0 1F E0 1F E0 1F E0 1F

01 E0 01 E0 01 E0 01 E0 et E0 01 E0 01 E0 01 E0 01,

1F FE 1F FE 1F FE 1F FE et FE 1F FE 1F FE 1F FE 1F

01 1F 01 1F 01 1F 01 1F et 1F 01 1F 01 1F 01 1F 01,

E0 FE E0 FE E0 FE E0 FE et FE E0 FE E0 FE E0 FE E0

4 de ces 12 clés sont dites également « **antipalindromes** » c'est-à-dire présentant des clés dérivées complémentaires les unes des autres, K_1 de K_{16}, K_2 de K_{15}... Ces 4 clés présentent l'inconvénient d'avoir 2^{32} points dits « **anti-fixes** », c'est-à-dire des blocs de 64 bits dont le chiffrement est le complément binaire d'eux-mêmes.

Un article paru dans la revue « Science » en 1983 soulevait le danger d'utiliser des clés présentant des « régularités ». L'auteur cite 48 clés dites potentiellement faibles.

En 1992, DEC (« Digital Equipement Corporation ») annonça au CRYPTO92 la fabrication d'une puce capable de chiffrer à un taux de 1 Go/s et ce pour un prix d'environ 1 500 FF.

Mise en œuvre du DES

Il existe quatre modes principaux de mise en œuvre du DES :

- le « **Electronic Code Book** », « **ECB** » ou « Mode du carnet de codage électronique »
- le « **Cipher Block Chaining** » , « **CBC** » ou « Mode de chiffrement avec chaînage de blocs »,
- le « **Output FeedBack** mode», « **OFB** » ou « Mode de rétroaction de sortie »,
- et le « **Cipher FeedBack mode** », « **CFB** » ou « Chiffrement à rétroaction ».

Ces mises en œuvre ne sont, en fait, pas propres au DES et peuvent être utilisées pour tout algorithme de chiffrement par blocs. Ils sont standards car décrits dans « US Department of Commerce Federal Information Processing Standard n° 81 ou « FIPS P 81» publié en 1980. Ces standards sont publiés par « NIST ».

L' « Electronic CodeBook mode » ou « mode dictionnaire »

C'est le mode le « plus évident ». Chaque bloc de 8 octets du message est chiffré avec la même clé. Ce mode opératoire est celui que nous avons décrit jusqu'à présent. C'est le mode natif ou « naturel » du DES. C'est un mode très pratique pour chiffrer tout texte en clair structuré en blocs de 64 bits. Nous avons :

$$\mathbf{C_n = E_K(T_n)} \text{ et } \mathbf{T_n = D_K(C_n).}$$

Toutefois, ce mode permet à l'attaquant de réaliser un **livre de codes** (« code book »), contenant, s'il dispose du texte en clair, les associations «clair/chiffré » d'un certain nombre de blocs de 8 octets. Ce peut être le cas dans l'utilisation du mode «ECB » au sein d'un réseau dont les protocoles normalisés présentent des répétitions de valeurs constantes à des positions « fixes » (blocs dit « stéréotypés »).

Le «Cipher Block Chaining mode » (« CBC ») ou «mode chaînage de blocs chiffrés »

Le bloc chiffré dépend non seulement du bloc en clair mais également de tous les blocs en clair qui précèdent ce dernier. Ce mode utilise une méthode de rétroaction.

Dans ce mode, nous avons :

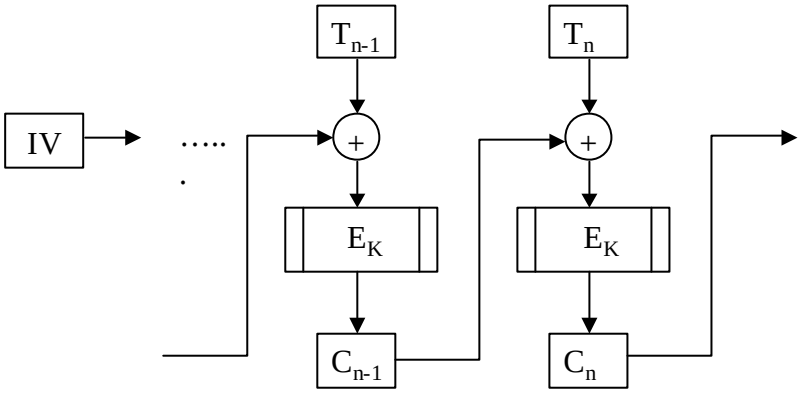
$$\mathbf{C_n = E_K(T_n \text{ Å } C_{n-1})} \text{ et } \mathbf{T_n = D_K(C_n) \text{ Å } C_{n-1}}$$

avec $\mathbf{C_1 = E_K(T_1 \text{ Å } IV)}$ et $\mathbf{T_1 = D_K(C_1) \text{ Å } IV}$

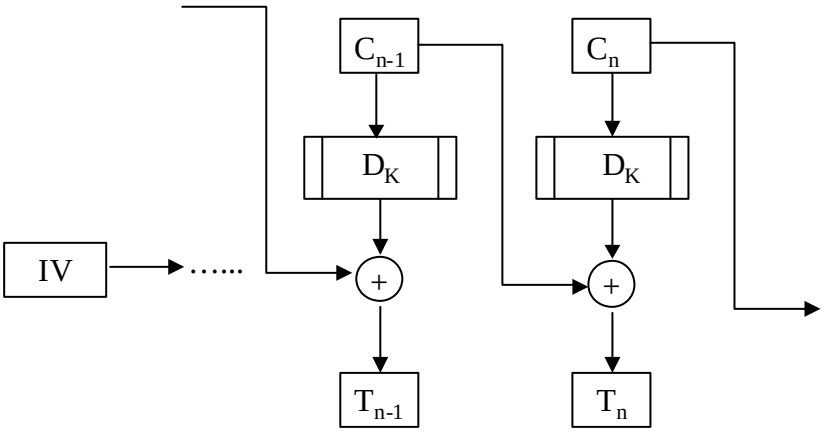
Il se pose, en effet, le problème du chiffrement du premier bloc et du dernier, si la longueur du texte n'est pas un multiple de 8 octets. On utilise alors un **vecteur d'initialisation** ou « **IV** » (« Initialisation Vector ») dont le choix devra être aléatoire, vecteur transmis avant l'opération de chiffrement, en mode « ECB ». Il devra être changé pour chaque message différent en lui associant, par exemple, un numéro de séquence à chaque émission.

Ce mode de chiffrement est dit « **auto-régulateur** ». En effet, une erreur de transmission, ou une malversation volontaire, sur un bit chiffré n'entraîne une erreur que sur 2 blocs consécutifs. La propagation des erreurs est donc finie.

Une erreur de «synchronisation », par contre, un bit de perdu ou d'ajouté lors de la transmission par exemple, entraîne une suite infinie d'erreur.



Déchiffrement en mode CBC



Le « Cipher FeedBack mode » (« CFB ») ou « mode rebouclage de texte chiffré »

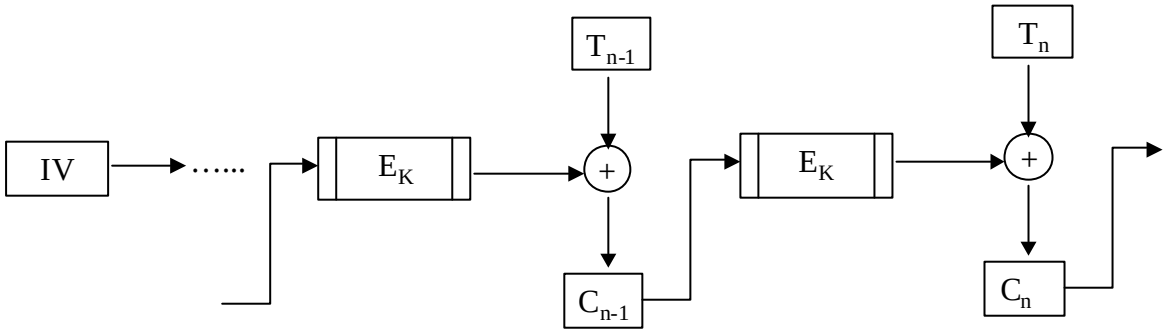
Le mode précédent convient bien pour des blocs sur 64 bits. Il est moins adapté pour des blocs de 8 bits. Ce mode, destiné aux messages transmis octet par octet ou bit par bit, a pour expression symbolique :

$$C_1 = T_1 \text{ Å } E_K(IV)$$

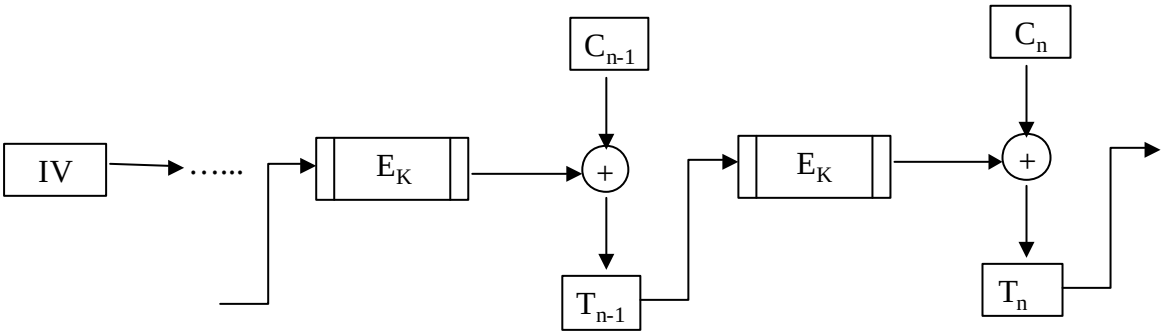
$$C_n = T_n \text{ Å } E_K(C_{n-1}) \text{ et } T_n = C_n \text{ Å } E_K(C_{n-1})$$

Ce mode opératoire est de la famille des chiffres de « **VERNAM** » dont les clés chiffrantes, suites de nombres aléatoires pouvant être périodiques, étaient aussi longues que les messages chiffrés. Ce type de chiffre fut sans doute utilisé pour protéger la ligne « Maison Blanche - Kremlin » plus connue sous le nom de « Téléphone Rouge ».

Ce mode est utilisé pour protéger les messages de type «caractères » existant, par exemple, au sein de transmissions asynchrones (téléimprimeurs, télex, terminal asynchrone...). Il permet en effet de chiffrer chaque caractère sans attendre qu'un nombre suffisant de caractères (8) soit arrivé.



Déchiffrement en mode « CFB »

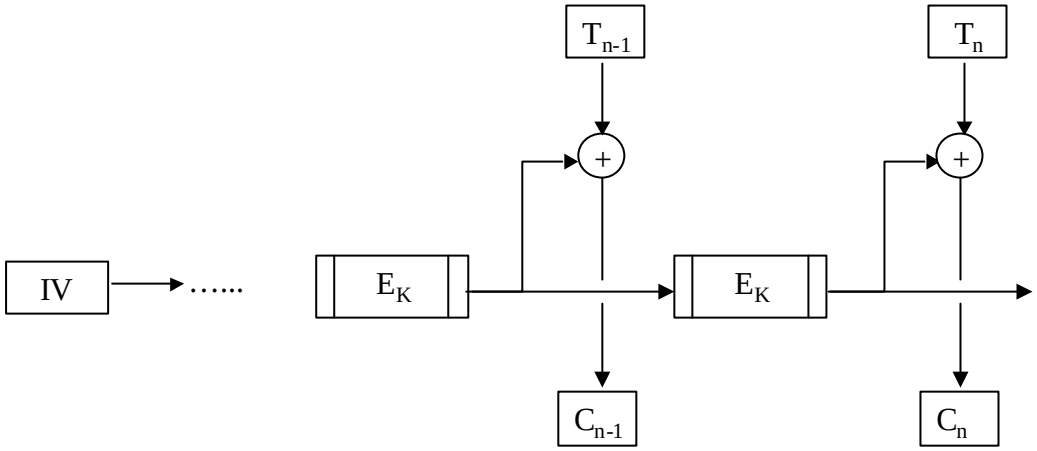


L' « Output FeedBack mode » (« OFB ») ou « mode rebouclage sur la sortie »

Ce mode est également utilisé dans les transmissions asynchrones comme les entrées au clavier par exemple. Nous avons :

$$\mathbf{DES_n = E_K(DES_{n-1})} \text{ et } \mathbf{C_n = T_n} \text{ Å } \mathbf{DES_n}$$

Dans ce mode, le mécanisme de rétroaction est indépendant à la fois des flots de texte en clair et des textes chiffrés.



Ce mode utilise le DES comme générateur de nombre aléatoire en bouclant sa sortie sur son entrée et en effectuant un « ou exclusif » sur cette sortie et sur le texte en clair.

Il n'y a pas de propagation de l'erreur sur un bit du texte en clair.

x x x

Le standard bancaire de l'ANSI spécifie les modes « ECB » et « CBC » pour le chiffrement et les modes « CBC » et « CFB » à n bits pour l'authentification.

Le Triple DES

Le « Triple DES » appartient à la famille des surchiffrements triples avec deux voire trois clefs.

Le « Triple DES » est un algorithme qui travaille sur des blocs de données de 64 bits en utilisant trois fois de suite l'algorithme du « DES », soit dans son utilisation en « chiffrement » soit dans son utilisation en «déchiffrement ». On utilise soit deux soit trois clés de 56 bits ce qui peut s'écrire sous les différentes formes suivantes :

$$E\left[K_1, D\left(K_2, E\left(K_1, T\right)\right)\right],\ E\left[K_1, D\left(K_2, E\left(K_3, T\right)\right)\right] \text{ et } E\left[K_1, E\left(K_2, E\left(K_1, T\right)\right)\right]$$

Les deux premières formes sont parfois nommées mode « chiffre-déchiffre-chiffre ».

Il semble raisonnable de croire, d'après un certain nombre d'études, qu'il n'existe pas K_3 telle que $E_{K_2}\left[E_{K_1}(T)\right] = E_{K_3}(T)$.