DES

Monty pour Zenk-Security

Juin 2011

Table des matières

Chapitre 1

Introduction

1.1 DES

DES est un algorithme de chiffrement symétrique à bloc.
Il existe deux grandes familles d'algorithme : symétrique ou asymétrique.
Les algorithmes symétriques utilisent la même clé pour chiffrer et déchiffrer, alors que les algorithmes asymétriques fonctionnent à l'aide d'une clé publique et d'une clé privée.
Les algorithmes symétriques se décomposent en deux sous-familles les algorithmes de chiffrement à flot et les algorithmes à bloc.
DES fait donc partie de cette dernière partie.

Le DES est aussi connu sous le nom DEA (Data Encryption Algorithm) pour la norme ANSI et DEA-1 pour la norme ISO.

1.2 Histoire

Le DES a été conçu par IBM à la fin des années 70, il est issu de LUCIFER, un autre algorithme conçu par IBM, qui a été modifié suite à une demande de la NSA. Les modifications venant de cette demande ont fait couler beaucoup d'encre à l'époque, selon les rumeurs, la NSA aurait caché une faille secrète dans le DES. Ces rumeurs étaient alimentées en partie par les tables-S (que nous verrons plus loin dans la description de l'algorithme) dont les valeurs peuvent sembler sortir de nulle part. Cependant avec du recul, on se rend compte que DES résiste bien à des attaques inconnues par le grand publique à l'époque, alors que LUCIFER lui y est plus sensible.

On ne sait toujours pas la réelle implication de la NSA dans la création de DES, mais il semblerait qu'elle ait renforcé l'algorithme et si une faille a été introduite, cette faille est toujours cachée.
De nos jours le DES n'est plus vraiment utilisé sous sa forme d'origine. La variante la plus repandue encore utilisée, est le Triple DES (3DES) que nous verrons plus tard.

Le DES a été le standard de chiffrement pour les organisations du gouvernement américain pendant un peu plus de 20 ans, il a été remplacé par AES en 2000.

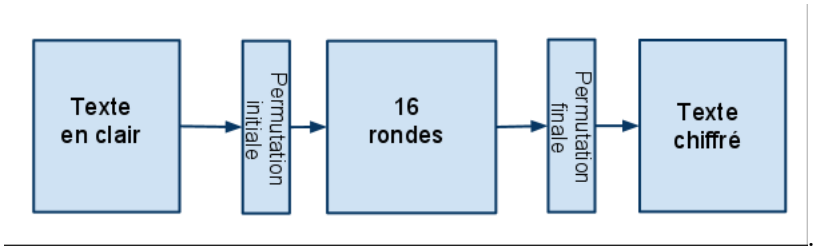
Chapitre 2

Vue d'ensemble

2.1 Plan général

Le DES est un algorithme de chiffrement par bloc de 64 bits.
Ainsi il prend en entrée un bloc de 64 bits de texte clair et ressort un texte chiffré de 64 bits.
C'est un algorithme symétrique, ainsi la clé de chiffrement est la même que la clé de déchiffrement.
La clé utilisée est une clé de 64 bits (mais seulement 56 bits sont utilisés, car on ignore un bit sur huit).

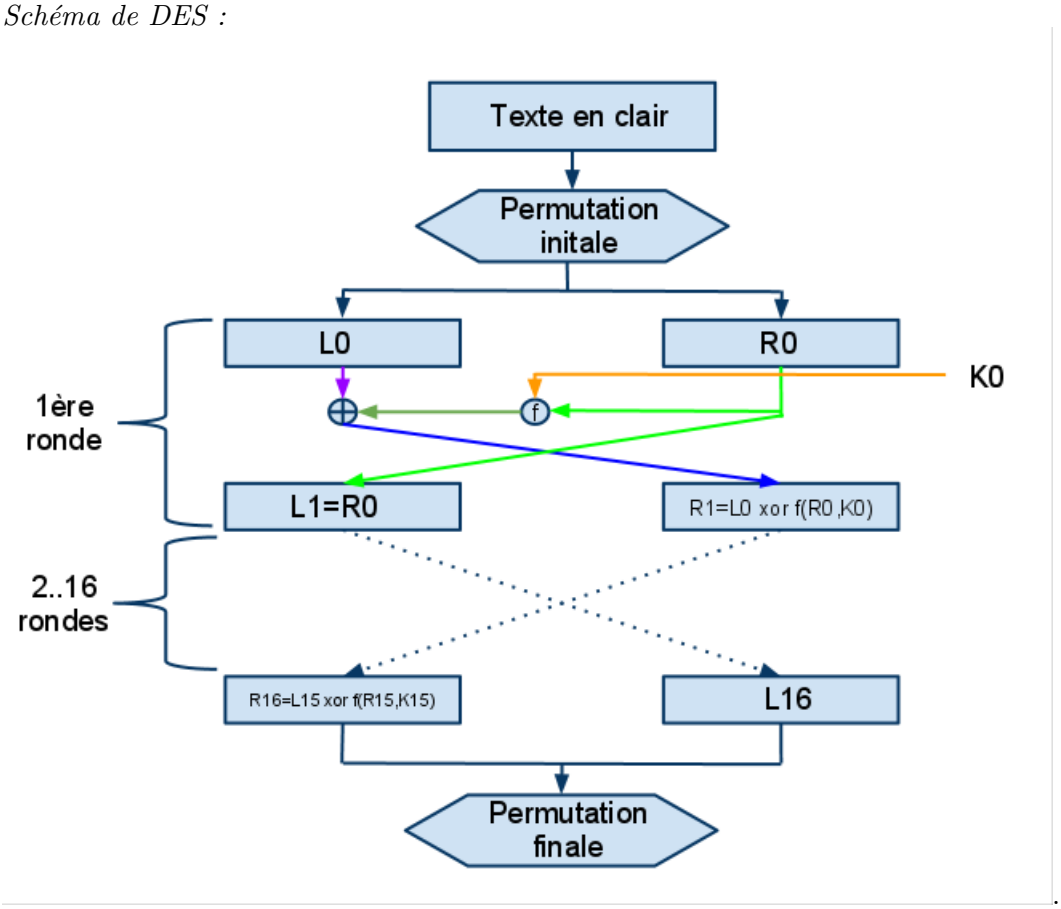
Sommairement l'algorithme se décompose comme ceci :
Tout d'abord on effectue une permutation initiale du texte, puis on effectue 16 "rondes" au texte, et pour finir une permutation finale est appliquée au texte.



Chaque "ronde" peut être représentée comme un schéma de feistel.
Au premier tour le texte permuté est découpé en deux partie (partie gauche/droite).
Puis à chaque tour on effectue une série d'opération.

On peut ainsi schématiser l'algorithme :

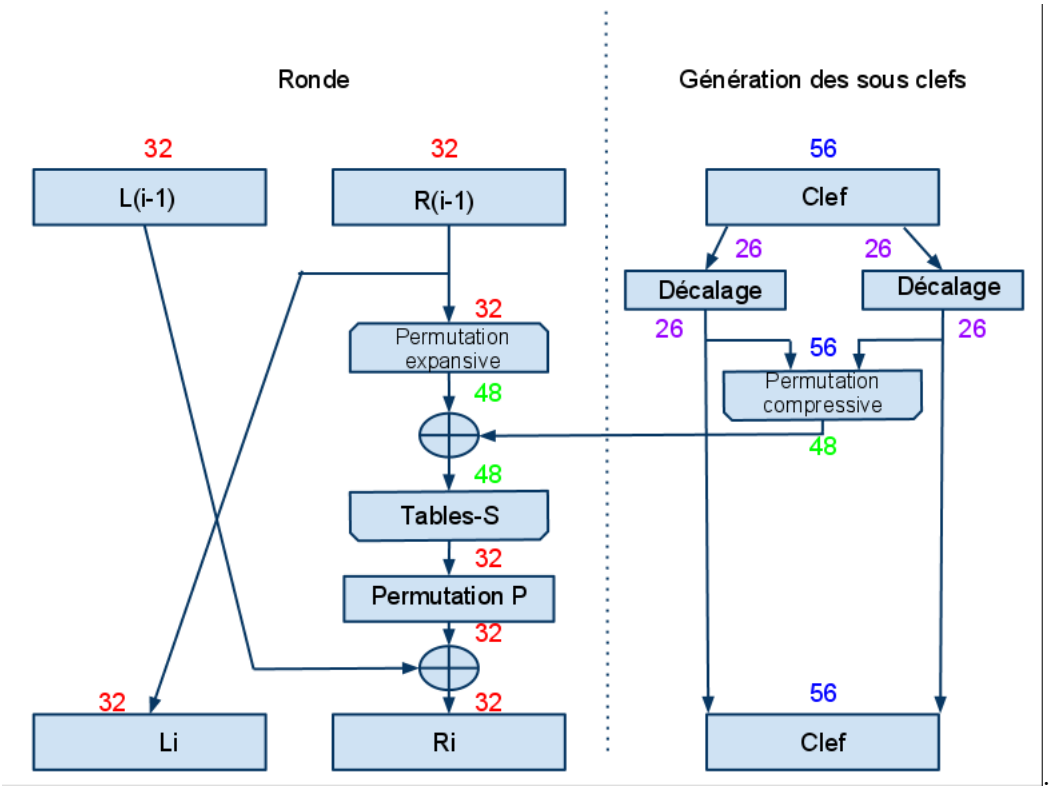
Schéma de DES :



2.2 Une ronde

On peut décrire une ronde ainsi :
$L_i = R_{i-1}$
$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

Schéma d'une ronde : (les nombres correspondent au nombre de bits à chaque étape)



À chaque ronde la partie droite passe d'abord par une permutation expansive qui étend les 32 bits initiaux à 48 bits.
Puis un XOR est effectué avec la sous-clé correspondante au tour.
Le texte ainsi obtenu est soumis à une opération de substitution par des Tables-S qui réduit les 48 bits à 32 bits.
Puis vient une permutation-P qui pour chaque bit d'entrée affecte un bit de sortie (il n'y a donc aucun bit perdu, ni aucun bit doublé, on pourra parler de permutation pure).
Pour finir, on combine le texte engendré par un ou exclusif avec la moitié droite initiale.

Chaque ronde utilise une sous-clé déduite de la clé initiale.
La clé initiale de 64 bits subit une permutation (qui supprime un bit sur huit). Il y donc 56 bits qui sont extrait.
Puis pour chaque ronde un décalage de bit est opéré sur la clé.
De ce décalage une permutation compresive est opéré, ce qui donne la sous-clé.

2.3 Déchiffrement

Le déchiffrement se fait de la même manière que le chiffrement, on doit simplement inverser l'ordre des sous-clés.

2.4 Mode opératoire

DES est compatible avec 4 modes opératoires : ECB, CBC, CFB,OFB.

2.4.1 ECB

C'est le mode le plus utilisé, chaque bloc de 64 bits est chiffré avec la même clé :
$C_i = E_k(T_i)$ (E pour Encryption, C pour le texte chiffré, T le texte en clair)

2.4.2 CBC

Dans ce mode, chaque bloc depend du précédent :
$C_0 = E_k(T_0 \oplus IV)$ (IV étant un vecteur d'initialisation)
$C_i = E_k(T_i \oplus C_{i-1})$

2.4.3 CFB

Ce mode est adapté pour les blocs de 8 bits.
$C_0 = T_0 \oplus E_k(IV)$
$C_i = T_i \oplus E_k(C_{i-1})$

2.4.4 OFB

Dans le dernier mode, on obtient la clé en chiffrant le précédent flux de clé.
$F_0 = IV$
$F_i = E_k(F_{i-1})$
$C_i = T_i \oplus F_i$

Chapitre 3

Détail des opérations sur le texte

3.1 Permutation initiale/finale

La permutation initiale est la permutation qui est effectuée avant la première ronde, chaque bit d'entrée est affecté à une position de sortie suivant le schéma suivant :

Permutation initiale :

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

Ainsi le premier bit en sortie est le bit 58ème bit en entrée, le second le bit 50 etc (attention ce tableau, comme ceux que l'on rencontre souvent compte les bits de 1 à 64 et non pas de 0 à 63).
Par exemple le code :
0101000000110000011011100011 0011
0111100101010010001100000111 1000
Devient :
1011010111111011000001000001 1000
0000000011011110100101000010 1100

La permutation finale est la permutation effectuée à la fin des 16 rondes, c'est l'inverse de la permutation initiale. Il est à noté que les parties gauches et droites ne sont pas échangées lors de la dernière ronde, on utilise donc $R_{16}L_{16}$ comme entrée pour la permutation finale.

Permutation finale :

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

3.2 Permutation expansive

Cette opération comme son nom l'indique augmente le nombre de bits, qui passe de 32 à 48.

Permutation expansive :

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

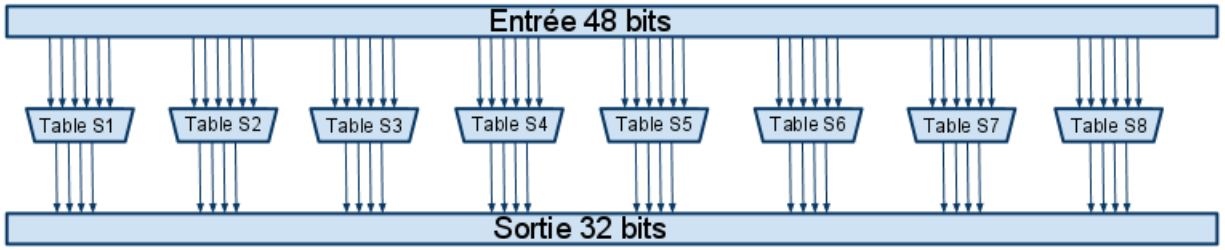
Il est intéréssant de constater que pour un bloc de sortie donné, un seul bloc d'entrée est possible, malgré le fait que le bloc de sortie soit plus grand.

3.3 Substitution par tables-S

C'est cette partie qui avait fait couler de l'encre lors de la publication de DES. En effet c'est l'origine un peu mystérieuse des valeurs des tables-S qui avait laissé penser que la NSA avait introduit une faille dans l'algorithme. De plus, on peut considérer que cette partie est la plus importante, car comparé à toutes les autres opérations, elle est non linéaire.

Les 48 bits issuent du ou exclusif entre la clé et le texte ayant subit la permutation expansive sont donc soumis à cette opération de substistution par les tables-S.

Il y a 8 tables-S et chaqu'une prend 6 bits en entrée et a 4 bits en sortie. On a donc 8 blocs de 6 bits en entrée (48bits) et 8 de 4 bits en sortie (32 bits).
Le premier bloc est traité par la 1er table-S, le second par la seconde etc.



La manière dont s'effectue l'opération est un peu particulière :
Soit X le groupe d'entrée constitué de 6 bits,
$x_1, x_2, x_3, x_4, x_5, x_6$
On forme un nombre de deux bits en combinant x_1 et x_6 : x_1x_6 (entre 0 et 3)
Ce nombre correspondra à la ligne.

Puis on forme un nombre de 4 bits en combinant x_2 à x_5 : $x_2x_3x_4x_5$ (entre 0 et 15)
Ce nombre correspondra à la colonne.

Ainsi pour un bloc de 6 bits en entrée et une table-S donné, la sortie correspondra à la valeur inscrite dans l'intersection de la ligne et de la colonne de la table-S.

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

Table 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

Table 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	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

Table 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

Table 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

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

Table 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

Table 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

Par exemple, si en entrée pour la table S1 nous avons :
000111

La ligne sera 01 = 2
La colonne sera 0011=3

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

La valeur en sortie sera donc **8 / 1000** :

3.4 Permutation P

Cette permutation est comme on la remarqué précédemment une permutation pure.

Permutation 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

Chapitre 4

Détail des opérations sur la clé

Comme nous l'avons vu précedemment, pour chaque ronde il y a une sous clé qui est calculé

4.1 Permutation clé

Tout d'abord avant la premiere ronde une permutation est effectuée sur la clé. Cette derniere réduit les 64 bits en 56.
On peut utilisé les 8 bits non utilisé pour effectuer une vérification sur la saisie de la clé.

Permutation clé :

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

4.2 Décalage

La clé est divisé en deux partie, gauche et droite. Puis on décale les bits de chaque partie vers la gauche (sans perte de bit, lorsqu'un bit dépasse par la gauche, il est inséré à droite).
Selon le tour de ronde, le décalage n'est pas le même :
Pour les rondes 1,2, 9 et 16 on ne décale que d'un bit, pour les autres rondes

on décale de deux bits.

4.3 Permutation compressive

Cette permutation renvoit 48 bits des 56 en entrée selon le schéma suivant :

Permutation compressive :

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

On peut remarquer que chaque décalage rend les clés différentes en modifiant les sous-ensembles utilisés pour leur création.

Chapitre 5

Sécurité de DES

5.1 BruteForce

La première méthode pour attaquer le DES qui vient à l'esprit est la méthode forte, qui consiste à tester toutes les possibilités.
Dans le cas du DES il y a 2^{56} clés différentes.
Lors de la sortie de DES, cette recherche semblait presque impossible, cependant avec les moyens techniques actuels, cette attaque peut être envisageable.

5.2 Cryptanalyse différentielle

La cryptanalyse différentielle consiste à étudier comment évoluent les données en sortie en fonction des données en entrée. Grossièrement il suffit de donner plusieurs textes connus en entrée, avec une différence fixée commune et entre eux. Puis, on étudie les résultats en sortie pour connaître la probabilité de certaine clés. À force de donner des textes connus en entrée on finit par découvrir la bonne clé.
Cette technique a été introduite après la création de DES, cependant ce dernier résiste plutôt bien à cette attaque, ce qui laisse penser que la NSA connaissait déjà cette attaque et qu'elle a modifié DES pour lui permettre d'y résister.

À l'aide d'une cryptanalyse différentielle, le nombre de clés est réduit à 2^{47}, seulement le nombre de textes clairs choisis est de 2^{47} (2^{55} si les textes clairs sont connus mais non choisis), ce qui rend cette attaque peu réalisable.

5.3 Cryptanalyse linéaire

La cryptanalyse linéaire consiste à créer une approximation linéaire de l'algorithme de chiffrement.
Avec cette attaque, on a besoin de 2^{43} texte clair connu.

À noter qu'il existe d'autre types d'attaques sur le DES plus ou moins efficaces (cryptanalyse différentielle-linéaire, compromis temps-mémoire...)

5.4 Clés faibles

Il existe certaines clés pour lesquelles DES est spécialement faible.
On dit qu'une clé K est faible si pour un message M :
$E_k(E_k(M)) = M$

La génération des sous-clés part du principe qu'à chaque tour la partie droite et la partie gauche de la clé sont décalées, or si tous les bits de chaque moitié sont soit à 1 soit à 0, le décalage n'aura aucun effet, et la sous-clé sera toujours la même.

Clef faible :

Valeur de la clef avec bits de parité				Clef	
0101	0101	0101	0101	00000000	00000000
1F1F	1F1F	0E0E	0E0E	00000000	FFFFFFFF
E0E0	E0E0	F1F1	F1F1	FFFFFFFF	00000000
FEFE	FEFE	FEFE	FEFE	FFFFFFFF	FFFFFFFF

(rappelez vous qu'une permutation est effectué sur la clef et qu'elle supprime un bit sur 8)

Il existe aussi des couples de clés dit semi-faible, ce sont des clés qui chiffrent un texte clair en un texte même chiffré.
K1 et K2 sont des clés semi faible si :
$E_{k1}(E_{k2}(M)) = M$

Cette particularité est du au fait qu'au lieu d'engendrer 16 sous clés différentes lors du décalage de bit, elles n'engendrent que 2 sous-clés, qui

seront utilisé 8 fois chacune. Les deux-sous clés de K1 correspondent au deux-sous clés de K2.

De même manière il existe des clés potentiellement faibles qui ne produisent que 4 sous-clés différentes.

Cependant il n'existe en tout que 64 clés faibles/semi faible ou potentiellement faibles, ce qui est négligeable par rapport au 2^{56} clés possibles. De plus, il est toujours possible de mettre en place une detection de ces clés faibles lors du chiffrement pour empécher leur utilisation.

Chapitre 6

Variante DES

6.1 Triple DES

La variante la plus courante de DES est le triple DES (ou 3DES).

DES n'est pas un groupe car il est faux de dire :
$\forall(K_1, K_2), \exists K_3 | E_{k1}(E_{K2}(x)) = E_{K3}(x)$

Ainsi appliquer avec plusieurs clés, DES renforce la sécurité, c'est ce que fais 3DES.
Le principe est simple, il applique successivement 3 fois l'algorithme DES avec 1, 2 ou 3 clés différentes.

Le standard du Triple DES exige que l'on utilise 3DES en mode EDE (Encryption, Decryption, Encryption), c'est à dire que l'on chiffre avec une première clé, puis que l'on déchiffre avec une seconde clé, puis que l'on chiffre à nouveau avec une troisième clé :
$C = E_{k3}(D_{k2}(E_{k1}(M)))$

Ceci a pour effet d'augmenter la sécurité de 3DES.

Le standard défini trois options pour les clés :

1. Trois clés indépendantes
2. K1 et K2 indépendants, et K3=K1
3. K1=K2=K3 (ce mode permet la compatibilité avec le DES simple)

6.2 Autres variantes

3DES est la variante la plus utilisée, mais il en existe d'autres tels que DESX, crypt(3) (utilisé dans les systèmes UNIX), GDES...