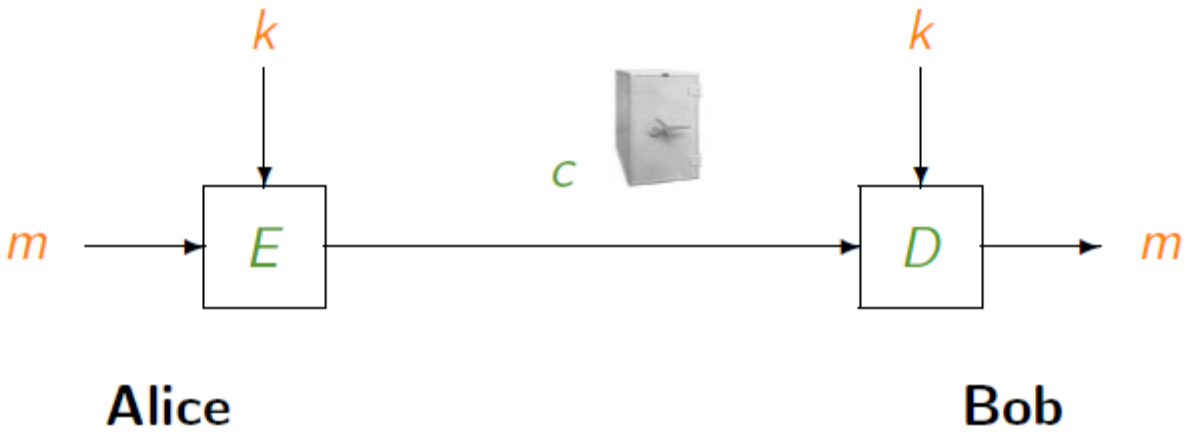
Chiffrement symétrique



Chiffrement : Alice chiffre son message m à l'aide de la clé secrète k,$c = E_k(m)$, et envoie le chiffré c à Bob.
Déchiffrement : Bob utilise la même clé k pour déchiffrer c, $m = D_k(c)$.
Principe de Kerkhoffs : les algorithmes de chiffrement E et déchiffrement D sont connus. Seule la clé k est secrète et nécessite un échange préalable.

Clé

Rôle : la clé est un « petit » secret partagé par Alice et Bob, qui leur permet de chiffrer de « grands » messages.

Constitution d'une clé : aujourd'hui, le plus souvent, les clés sont des chaînes de bits.

Condition nécessaire de sécurité :

- le nombre de clés doit être suffisamment grand pour échapper à la recherche exhaustive.
  Actuellement, on estime que la longueur minimum d'une clé d'un système de chiffrement symétrique doit être de 80 bits. L'espace des clés comprend alors au moins $2^{80} \approx 1,2 \times 10^{24}$ clés différentes.
- le choix de la clé secrète partagée par Alice et Bob doit être parfaitement aléatoire (chaque bit de la clé doit avoir une chance sur deux d'être un 1).

Caractéristique de chiffrement symétrique.

Avantages

- Systèmes rapides (implantation matérielle)
- Clés relativement courtes (128 ou 256 bits)

Inconvénients

- Gestion des clés difficiles (nombreuses clés)
- Point faible = échange d'un secret

Deux grandes familles de chiffrement

2 grandes familles: blocs et flots

Chiffrement par blocs: les messages sont découpés en blocs

- DES: blocs de 64 bits, clés de 56 bits
- IDEA: blocs de 64 bits, clés de 128 bits
- AES: blocs de 128 bits, clés de 128, 256 bits
- ...

Chiffrement par flots: les données sont traitées en flux

- Pseudo-Vernam : on « XOR » un pseudo-aléa au flux
- RC4 : chiffrement octet par octet
- ...

Chiffrement par blocs

Chiffrement par blocs.

Définition

On désigne par *chiffrement par blocs* (block-cipher en anglais), tout système de chiffrement (symétrique) dans lequel le message clair est découpé en blocs d'une taille fixée, et chacun de ces blocs est chiffré.

La longueur n des blocs et la taille l des clés sont deux caractéristiques des systèmes de chiffrement par blocs.

Découpage en blocs.

Le message $\mathbf{m}$ à chiffrer est découpé en blocs de n bits.

$$\mathbf{m} = m_1 m_2 \dots m_k .$$

Si la longueur du message n'est pas un multiple de la longueur d'un bloc, on le complète : c'est le *bourrage* ou *padding* en anglais. Plusieurs techniques de bourrage existent.

Technique de bourrage.

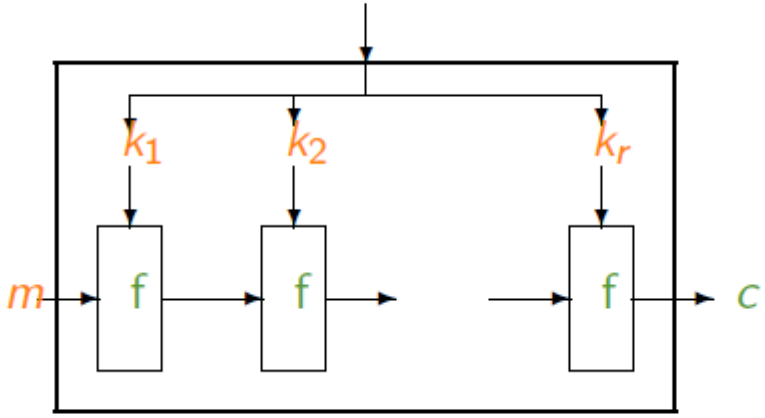
Une façon de bourrer (RFC2040) consiste à compléter le dernier bloc par autant d'octets que nécessaire, chaque octet ayant pour valeur le nombre d'octets ajoutés.

Par exemple, s'il manque trois octets au message $m = o_1 o_2 o_3 o_4 o_5$ pour obtenir un bloc de huit octets, on ajoute trois octets égaux à 3

o_1	o_2	o_3	o_4	o_5	03	03	03

Chiffrement itératif

Tous les sytèmes de chiffrement par blocs actuels suivent le schéma suivant

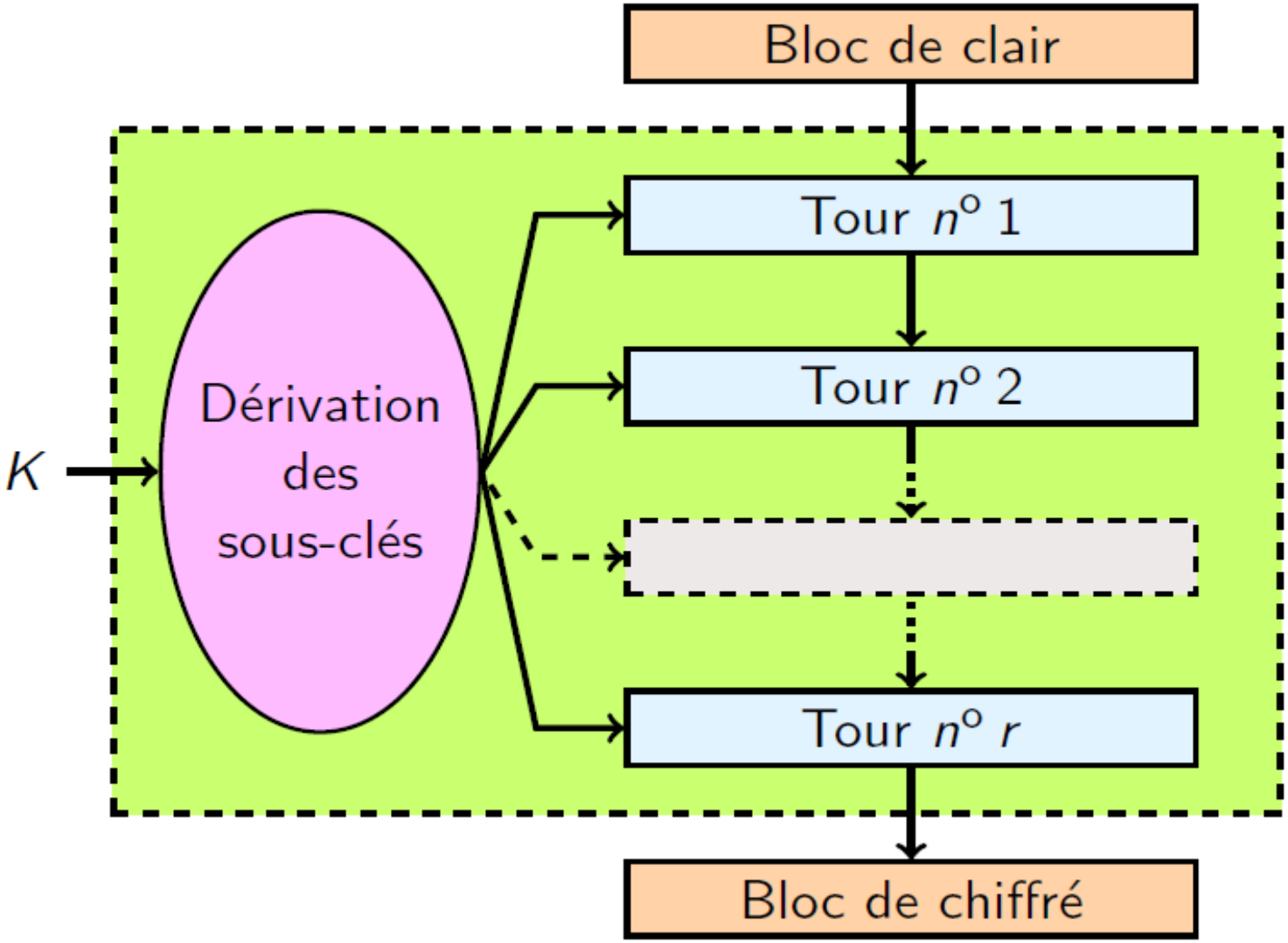


Le bloc clair m est transformé r fois successivement à l'aide d'une fonction f qui dépend d'une sous-clé k_i. Le chiffré c est le résultat de la dernière transformation.

$$c = f(\ldots f(f(m, k_1), k_2), \ldots), k_r).$$

r est appelé nombre de *tours* ou de *rondes*.

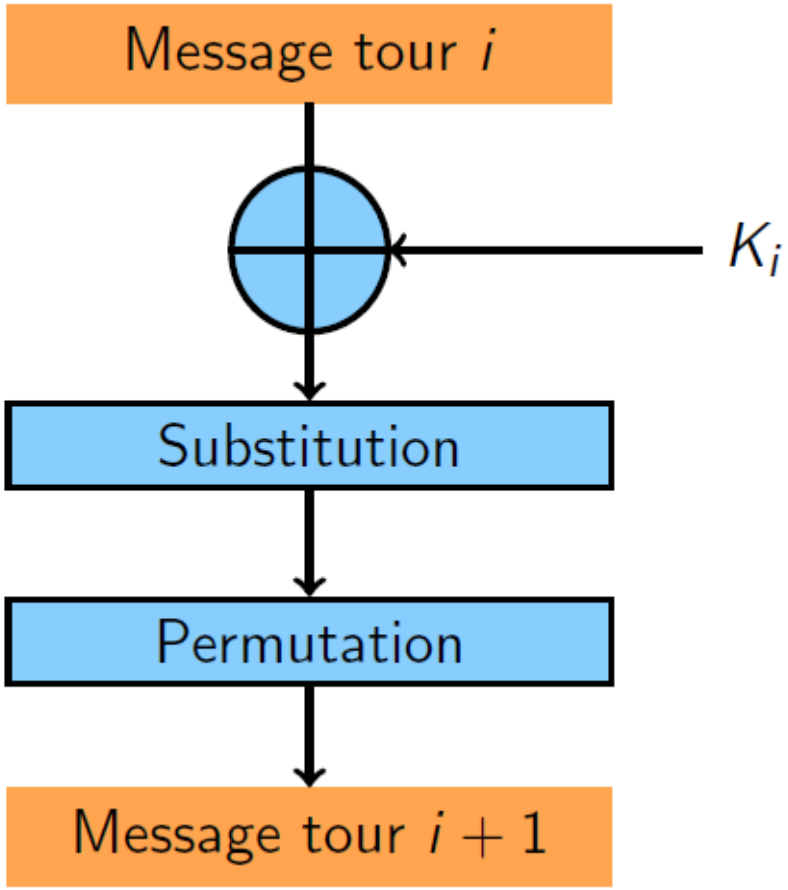
Construction de chiffrement par blocs.



Méthode 1: substitutions - permutations

Décomposition d'un tour:

- Intervention de la sous-clé
- Couche de substitution
- Couche de permutation



Méthode 2: schéma de Feistel

Dans une construction de Feistel, le bloc d'entrée d'un round est séparé en deux parties. La fonction de chiffrement est appliquée sur la première partie du bloc et l'opération binaire OU-Exclusif (+) est appliquée sur la partie sortante de la fonction et la deuxième partie. Ensuite les deux parties sont permutées et le prochain round commence.

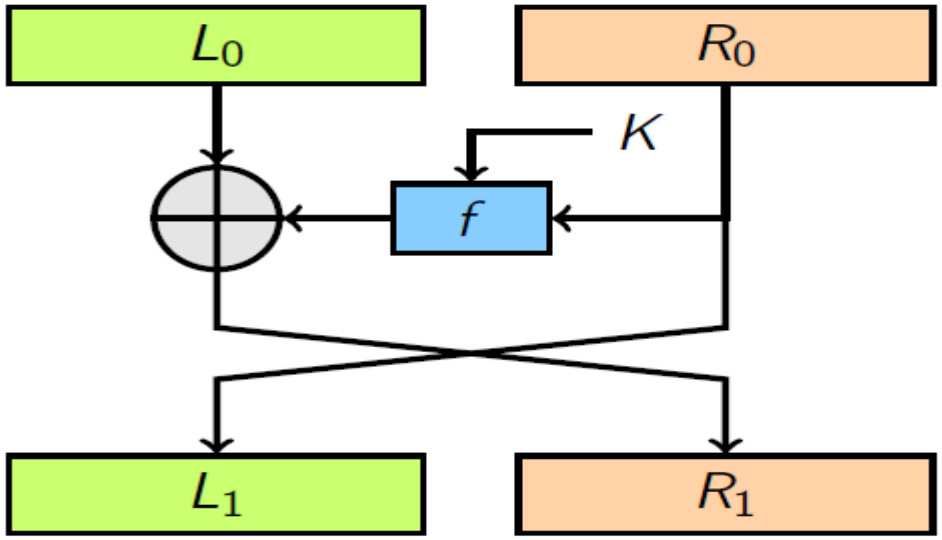
L'avantage est que la fonction de chiffrement et la fonction de déchiffrement sont identiques. Ainsi la fonction n'a pas à être inversible, c'est la structure qui l'est.

Chiffrement:

$$L_1 = R_0$$
$$R_1 = L_0 \oplus f(R_0)$$

Le déchiffrement est trivial :

$$R_0 = L_1$$
$$L_0 = R_1 \oplus f(R_0)$$

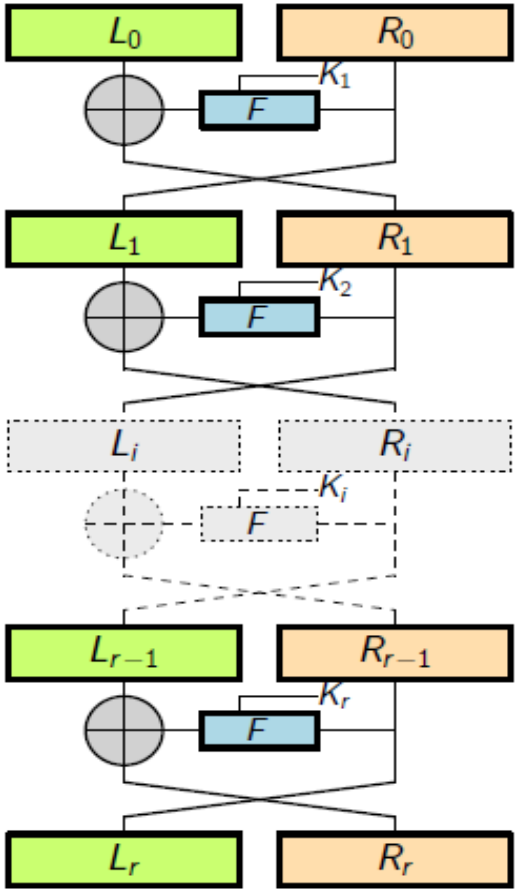


La fonction f est appelée la *fonction de confusion*

Schéma de Feistel.

Empilement de schéma de Feistel

Un tour isolé donne une bijection simpliste, la complexité croît de façon exponentielle avec le nombre de tours



Caractéristiques.

Bonnes performances, et sécurité bien étudiée

Avantages

Les *block ciphers* sont de façon générale:

- rapides (en matériel et en logiciel)
- bien conçus car théorie sous-jacente mature

Inconvénients

Principaux problèmes spécifiques aux blocs:

- utilisation des modes opératoires
- sécurité difficile à évaluer exactement

Choix des paramètres

La réalisation d'un tel chiffrement dépend des choix effectués pour les paramètres suivants :

- Taille du bloc : si elle augmente, la sécurité augmente également
- Taille de clé : si elle augmente, la sécurité aussi
- Nombre de cycle : plus il y en a, plus la sécurité est renforcée
- Algorithme de génération des sous-clés : plus il est complexe, plus la compréhension est rendue difficile.

Chiffrement par blocs: les messages sont découpés en blocs

- DES: blocs de 64 bits, clés de 56 bits
- IDEA: blocs de 64 bits, clés de 128 bits
- AES: blocs de 128 bits, clés de 128, 256 bits
- ...

Data Encryption Standard DES

DES simplifié (S-DES)

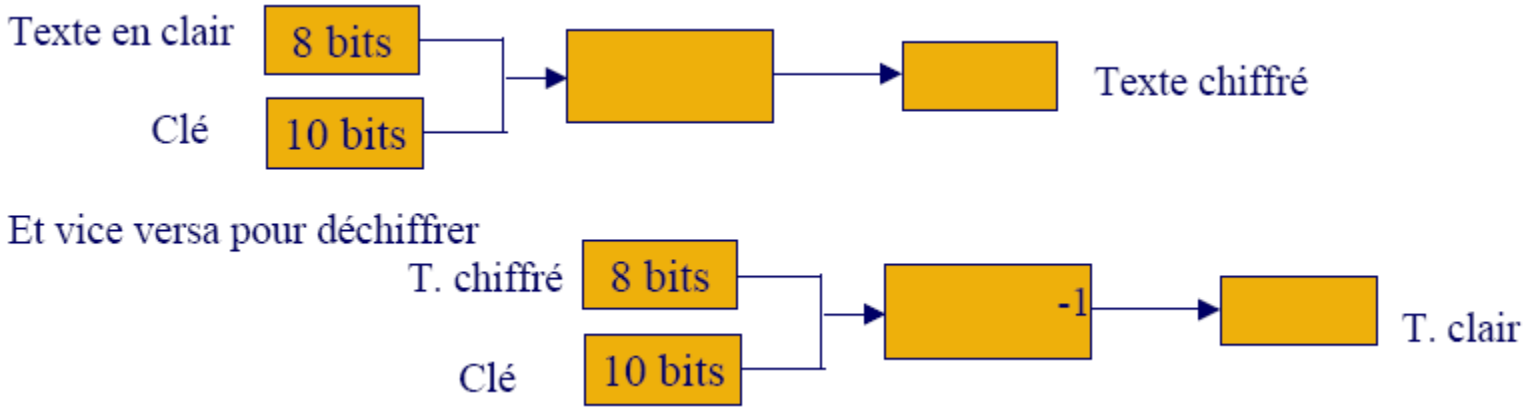
- La compréhension du DES et de ses principes aidera à comprendre les autres algorithmes conventionnels
- Le DES est un système de chiffrement par blocs.
- S-DES est une version simplifiée pour illustration des principes

DES-simplifié

Il a les mêmes propriétés et structure que le DES, avec des paramètres moins longs développé par le Prof. Schaefer de Santa Clara U.

DES simplifié (S-DES)

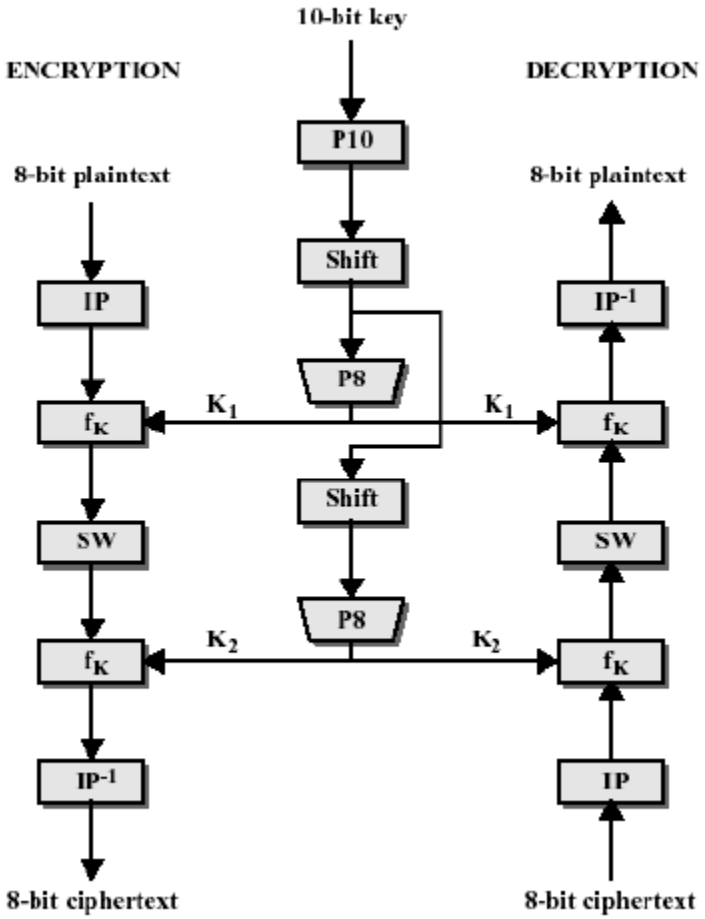
- Description: S-DES agit sur des blocs de 8 bits du texte en clair avec un clé de 10 bits



DES simplifié (S-DES)

L'algorithme de chiffrement (5 fonctions)

1. Permutation initial (IP)
2. Fct complexe f_k → permutation + substitution + Clé
3. Permutation simple (SW) →permuté les 2 moitiés d'une donnée
4. Fct f_k (avec une autre clé)
5. Permutation inverse (IP^{-1})



DES simplifié (S-DES)

Plus le nombre de permutation et de substitution augmente plus l'algorithme est complexe et plus il est difficile à le casser avec une cryptanalyse.

initial (IP)

f_K: input {données+ clé 8 bits}

L'algorithme peut être conçu à utiliser des clés de 16 bits

- une fois une sous clé de 8 bits et une autre fois avec l'autre sous clé
- ou bien à la base d'une clé de 10 bits on génère les deux s/clés (compromis)

Génération des clés:

K: les opérations sur la clé de 10 bits

→ **Permutation P10**

→ Décalage (shift-1)

→ Permutation P8 → s/clé **K1**

→ Décalage(shift-2)

→ Permutation P8 → s/clé **K2**

DES simplifié (S-DES)

$K1 = P8(shift(P10(K)))$ et $K2 = P8(shift(P10(K)))$

Algorithme de chiffrement : composition de fonctions binaires

$$Tchiffré = IP^{-1}(fk2(SW(fk1(IP(tclair)))))$$

Déchiffrement est l'opération inverse, donnée par

$$Tclair = IP^{-1}(fk1(SW(fk2(IP(tchiffré)))))$$

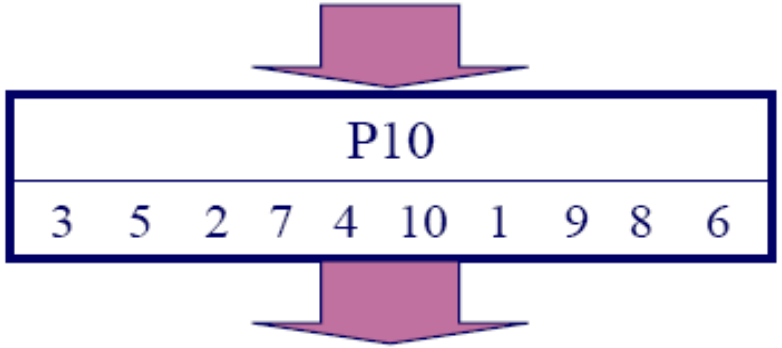
Seule différence → l'ordre des clés.

DES simplifié (S-DES) Génération des s/clés

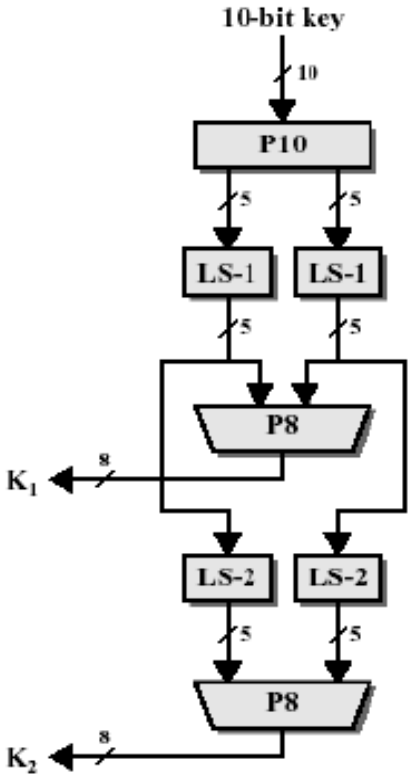
Clé de 10 bits $K = (k_1 , k_2 , k_3 , k_4 , k_5 , k_6 , k_7 , k_8 , k_9 , k_{10})$

1- Opération P10 : permute les bits selon la règle suivante

$P10(k_1 , k_2 , k_3 , k_4 , k_5 , k_6 , k_7 , k_8 , k_9 , k_{10})$

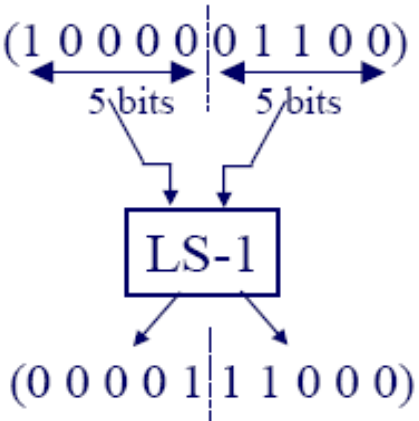


$(k_3 , k_5 , k_2 , k_7 , k_4 , k_{10} , k_1 , k_9 , k_8 , k_6)$



2- Opération LS-1: décalage à gauche circulaire d'un bit

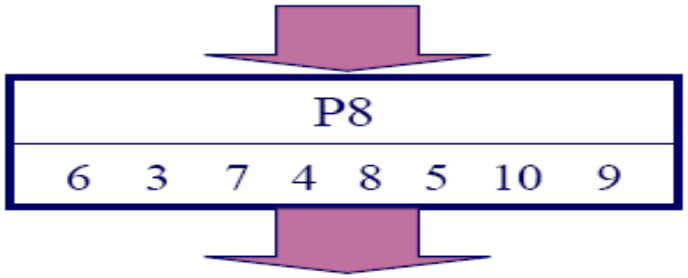
Exemple: P10(1 0 1 0 0 0 0 0 1 0) = (1 0 0 0 0 | 0 1 1 0 0)



DES simplifié (S-DES)

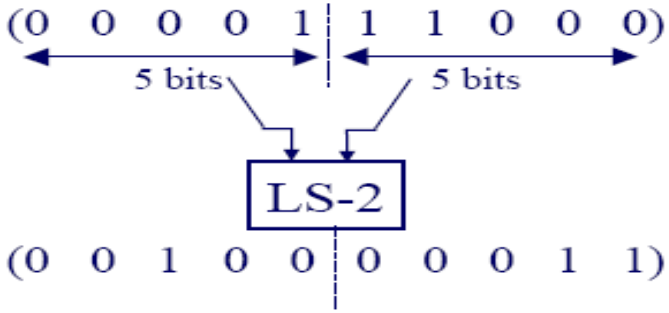
3- Opération P8 : permute 8 bits selon la règle suivante

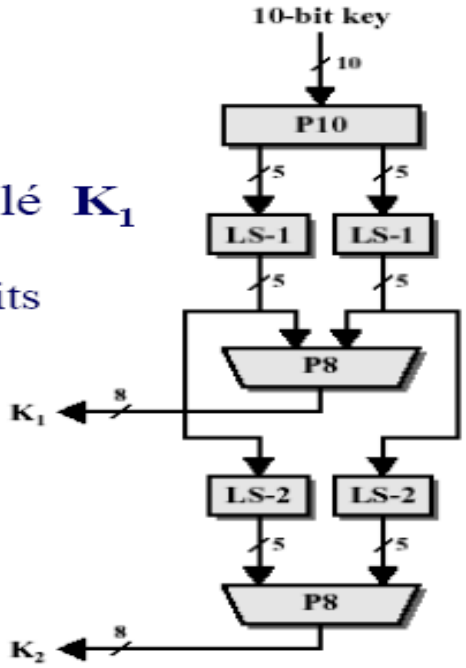
P8 (0 0 0 0 1 1 1 0 0 0)



(1 0 1 0 0 1 0 0) ➔ s/clé K_1

4- Opération LS-2: décalage à gauche circulaire de deux bits

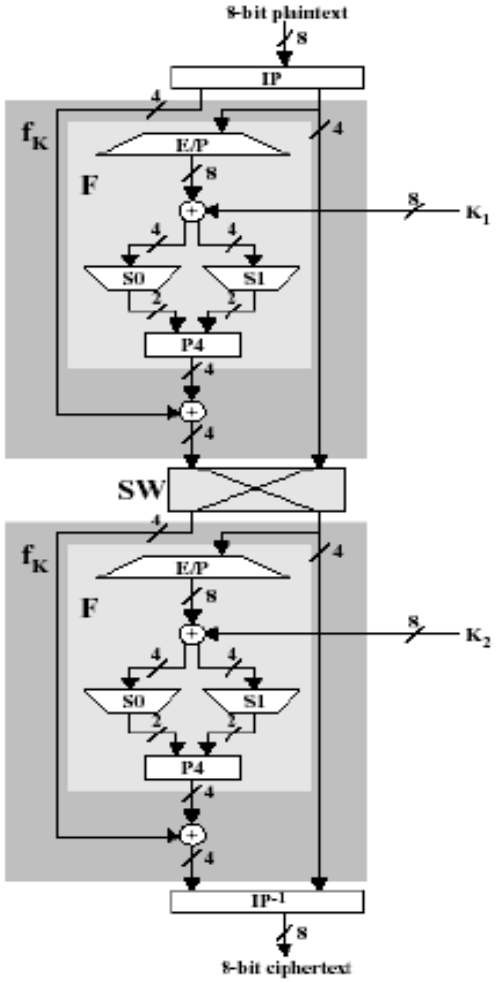




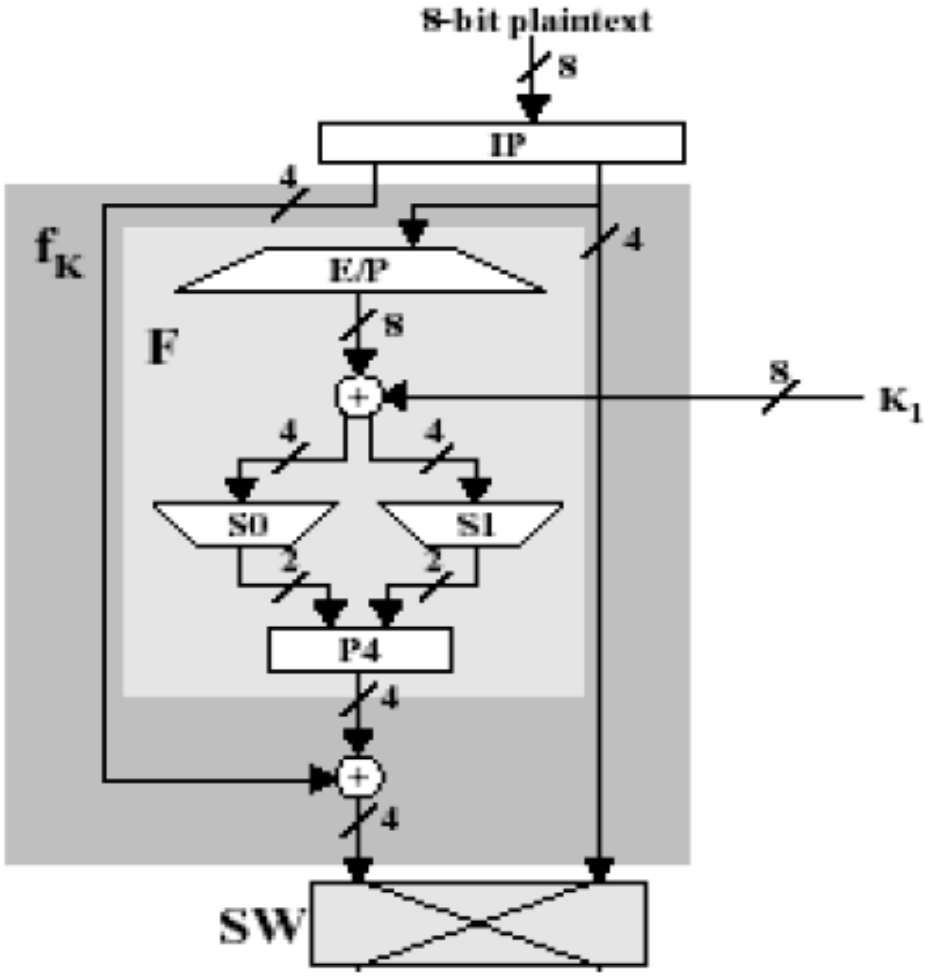
5- Opération P8: permute 8 bits selon la règle ci-dessus

P8 (0 0 1 0 0 0 0 0 1 1) = (0 1 0 0 0 0 1 1) ➔ s/clé K_2

DES simplifié (S-DES)



L'algorithme S-DES



La 1ère rond du S-DES

L'algorithme S-DES (5 fonctions séquentielles)

Permutations Initiale et Finale

tclair

1- Fonction IP permute les 8 bits

IP							
2	6	3	1	4	8	5	7

→ 1 0 1 1 1 1 0 1 (ex)

Et à la fin de l'algorithme on utilise la permutation IP^{-1}

IP^{-1}							
4	1	3	5	7	2	8	6

D'ailleurs $IP^{-1}(IP(X)) = X$

2- Fonction f_K (la composante la plus complexe)

f_K = fcts{combinaison, permutation, substitution}

Input de f_K est de 8 bits (4 L | 4 R) ➔ $f_K(L,R)=(L \oplus F(R,SK), R)$

Fonction F fait correspondre 4 bits → s/clé SK → 4 bits

F(1 1 0 1, SK) = (1 1 1 0) (ex)

alors $f_K(10111101) = (1011 \oplus 1110 \,|\, 1101) = (01011101)$

3- Fonction F : agit sur 4 bits (n_1 , n_2 , n_3 , n_4) → expansion/permutation (E/P)

E/P							
4	1	2	3	2	3	4	1

n_4	n_1 n_2	n_3
n_2	n_3 n_4	n_1

8 bits de la s/clé (SK) K_1
(k_{11} , k_{12} , k_{13} , k_{14} , k_{15} , k_{16} , k_{17} , k_{18})

On retrouve :

$n_4 + k_{11}$	$n_1 + k_{12}$	$n_2 + k_{13}$	$n_3 + k_{14}$
$n_2 + k_{15}$	$n_3 + k_{16}$	$n_4 + k_{17}$	$n_1 + k_{18}$

On renomme les 8 bits par :

$p_{0,0}$	$p_{0,1}$	$p_{0,2}$	$p_{0,3}$
$p_{1,0}$	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$

4 bits

4 bits

Exemple:

($p_{0,0}$, $p_{0,3}$) → ligne (00)

($p_{0,1}$, $p_{0,2}$) → colonne (10)

S0 (ligne 0 , col. 2) = 3 = (1 1)

($p_{1,0}$, $p_{1,3}$) → ligne (01)

($p_{1,1}$, $p_{1,2}$) → colonne (01)

S1 (ligne 1 , col. 1) = 0 = (0 0)

S1 S-Box S0

2 bits 2 bits

S0 :

1	0	3	2
3	2	1	0
0	2	1	3
3	1	3	2

S1 :

0	1	2	3
2	0	1	3
3	0	1	0
2	1	0	3

Les 4 bits S0S1 subissent une permutation P4

P4			
2	4	3	1

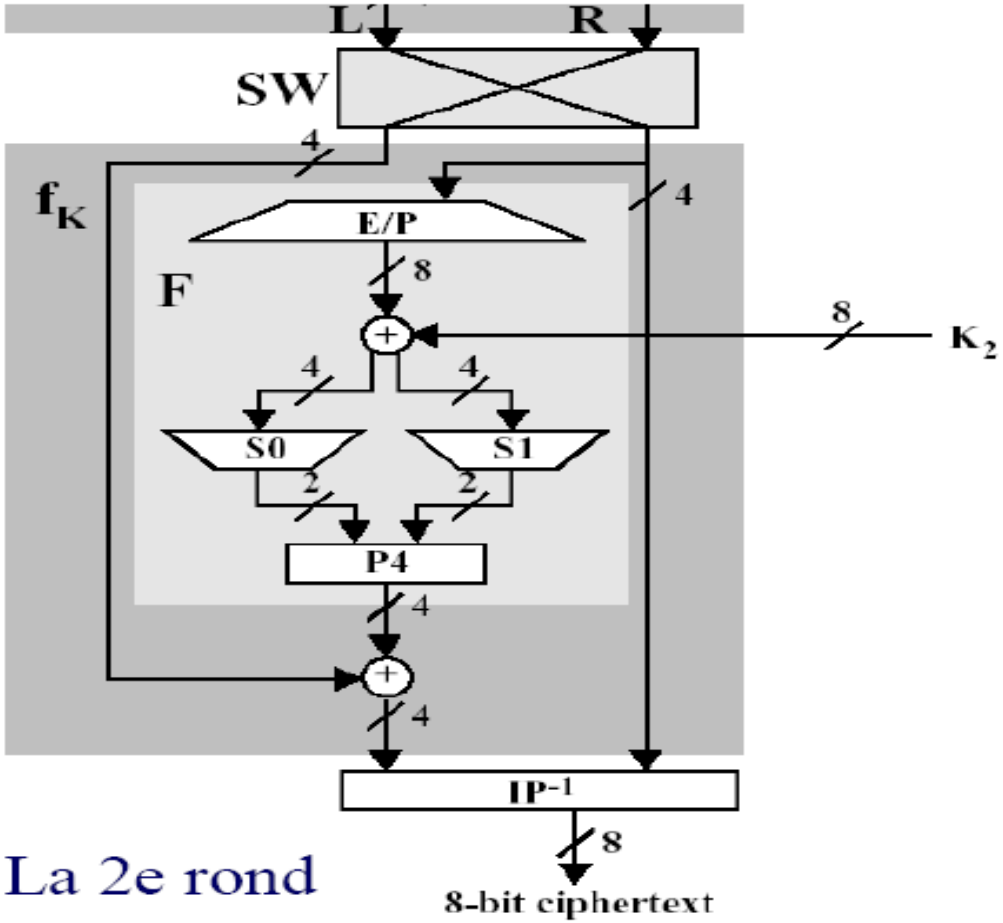
4 bits

Sortie de F

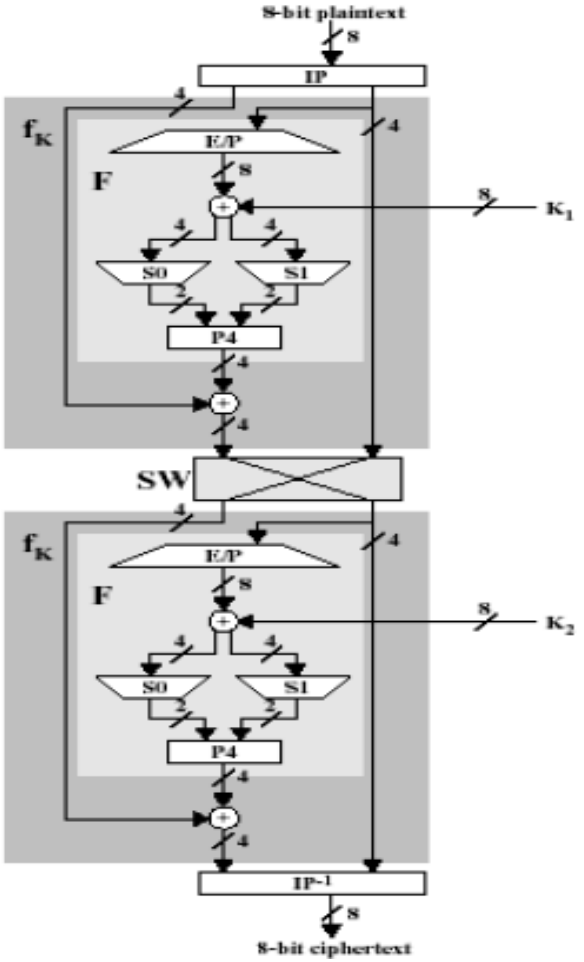
DES simplifié (S-DES)

4- Fonction SW :

- La fonction f_K altère seulement les 4 bits de gauche (L)
- La fonction SW interchange les obtenus L et R et elle est suivie de la même fonction F qui agira cette fois-ci sur les 4 bits R mais avec la s/clé K_2



La 2e rond



DES simplifié (S-DES)

Relation avec le DES

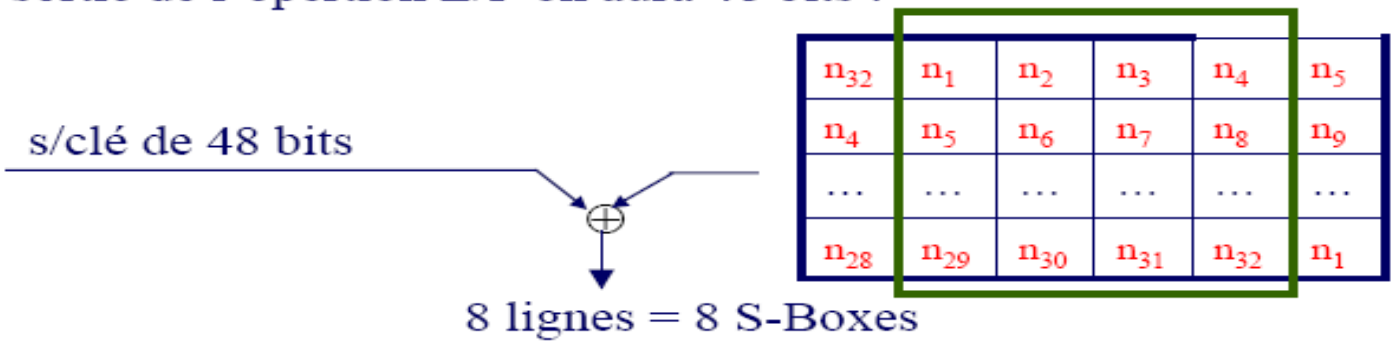
Le DES opère sur des blocs de 64 bits avec le schéma de chiffrement:

$$\mathbf{IP^{-1} \circ f_{K16} \circ SW \circ f_{K15} \circ SW \circ \ldots \circ f_{K2} \circ SW \circ f_{K1} \circ IP}$$

16 s/clés (16 rounds) de 48 bits obtenues d'une clé de 56 bits

Dans l'algorithme, la fonction F agit sur 32 bits (n_1 , n_2 , …, n_{32})

À la sortie de l'opértion E/P on aura 48 bits :



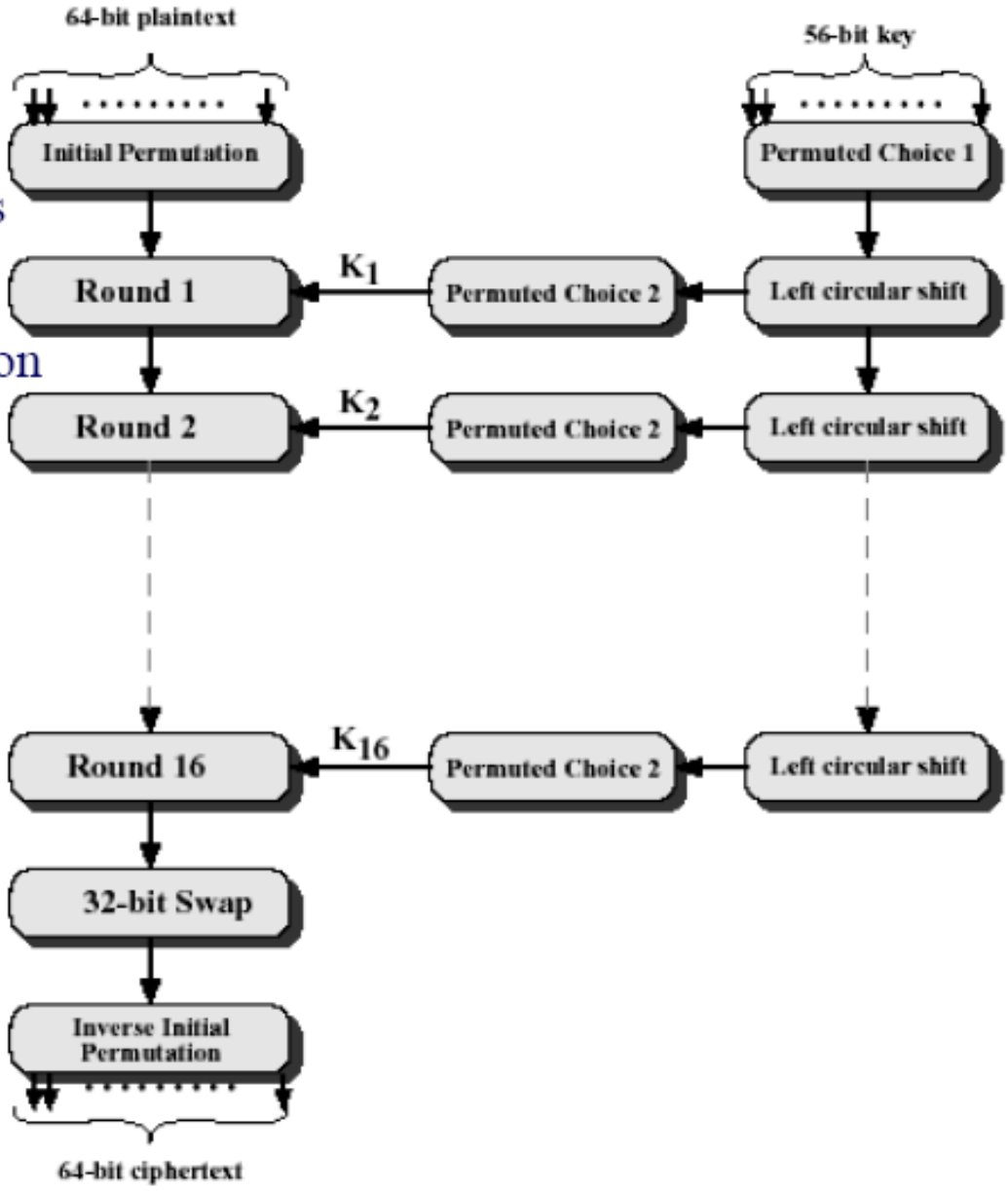
Chaque S-Box → 4 lignes & 16 colonnes

→ 1er bit et le dernier bit → ligne

→ 4 bits au milieu → colonne

Étape de l'algorithme

- Permutation initiale (IP)
- Bloc divisé en 2 moitiés de 32 bits chaque
- Application de 16 ronds d'opération identiques
- Après la 16e ronde permtation 32 bits gauche avec 32 bits droite
- Permutation inverse (IP^{-1})



Opérations IP et IP^{-1}

Transposition du bloc, l'une est linverse de l'autre

$X=IP(M) \Leftrightarrow Y=IP^{-1}(X)=IP^{-1}(IP(M)=M$

(a) Initial Permutation (IP)

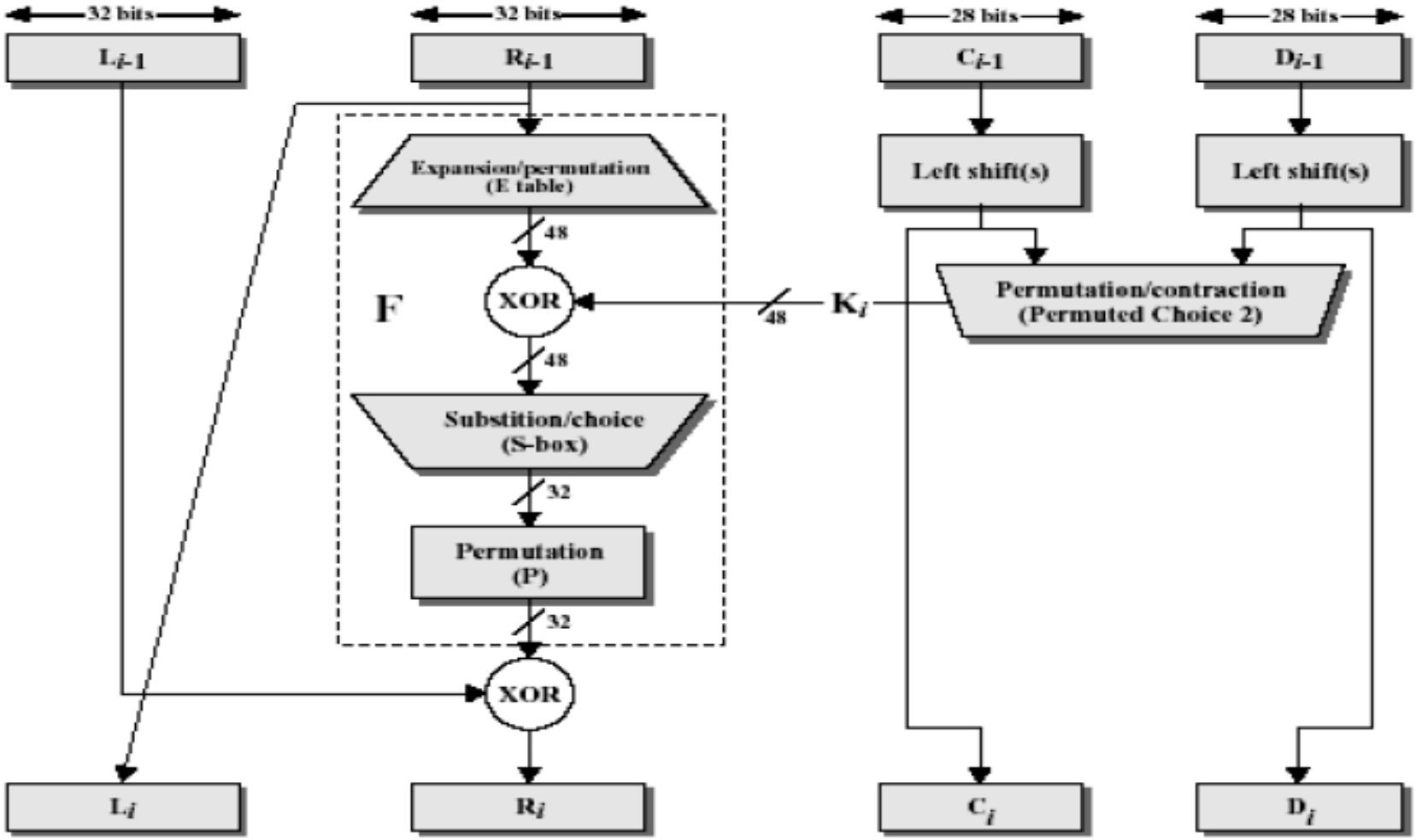
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial 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

Une ronde

$B_{i-1} = [L_{i-1} , R_{i-1}]$ → les deux parties de (32bits) sont traitées séparément



$L_i = R_{i-1}$ $\quad R_i = L_{i-1} \oplus F(R_{i-1} , K_i)$

R subit une permutation expansive à l'aide de la table E
Elle étend les 32 bits sur 48 bits
→ bloc divisé en blocs de 4 bits et ajout de 2 bits latéraux

Le but:
Avoir une longueur = sous-clé
Permettre une compression lors de la substitution

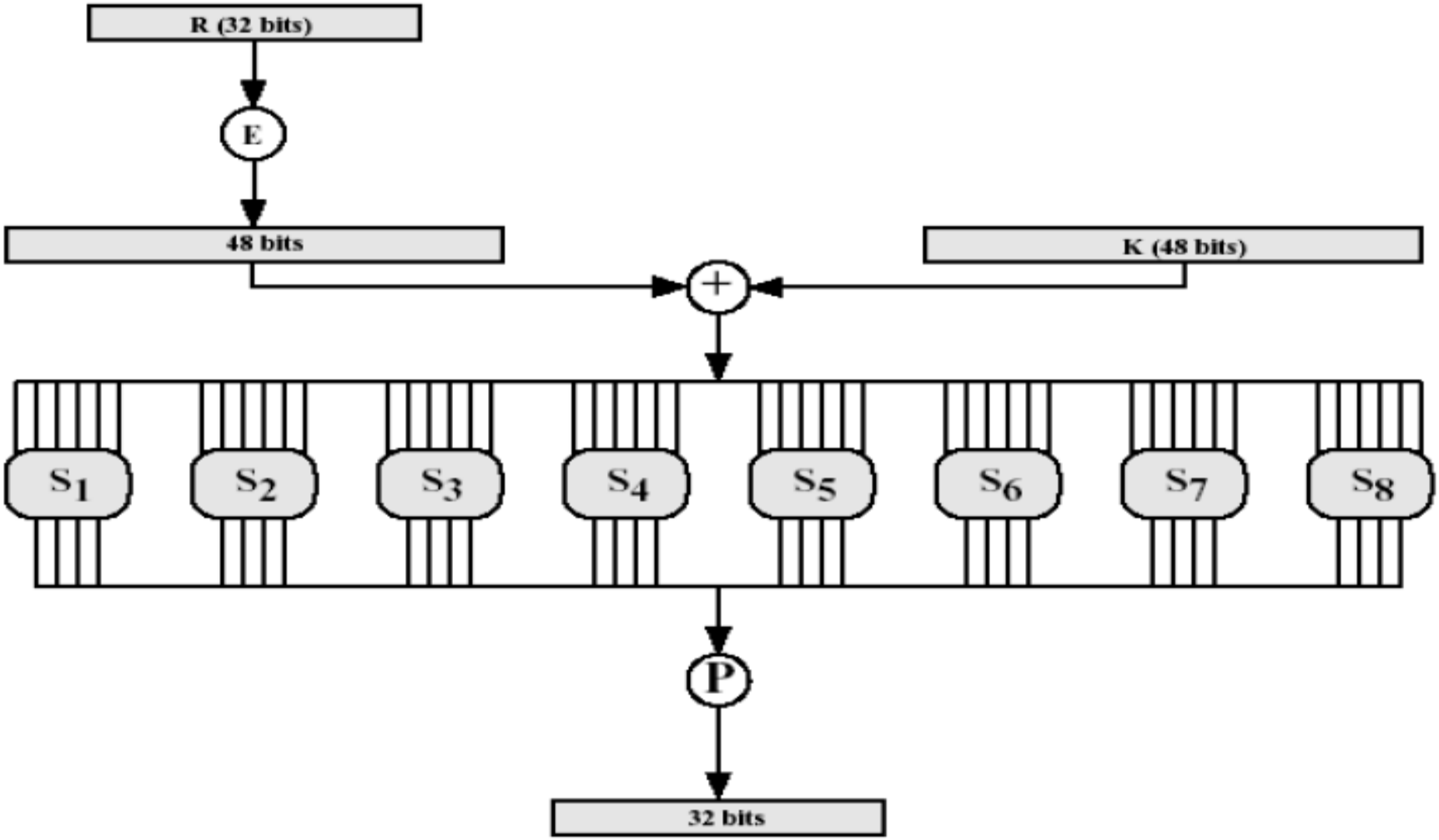
(c) Expansion Permutation (E)

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

Substitution par tables S-Box → partie cruciale de l'algorithme qui lui confère la sécurité (partie non-linéaire)

Les 48 bits sont divisés en blocs de 6 bits → [1 2 3 4 5 6]

[1 6] → numéro de la ligne et [2 3 4 5] → numéro de la colonne



Les 8 blocs se recombinent et forme 32 bits

S-Box

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

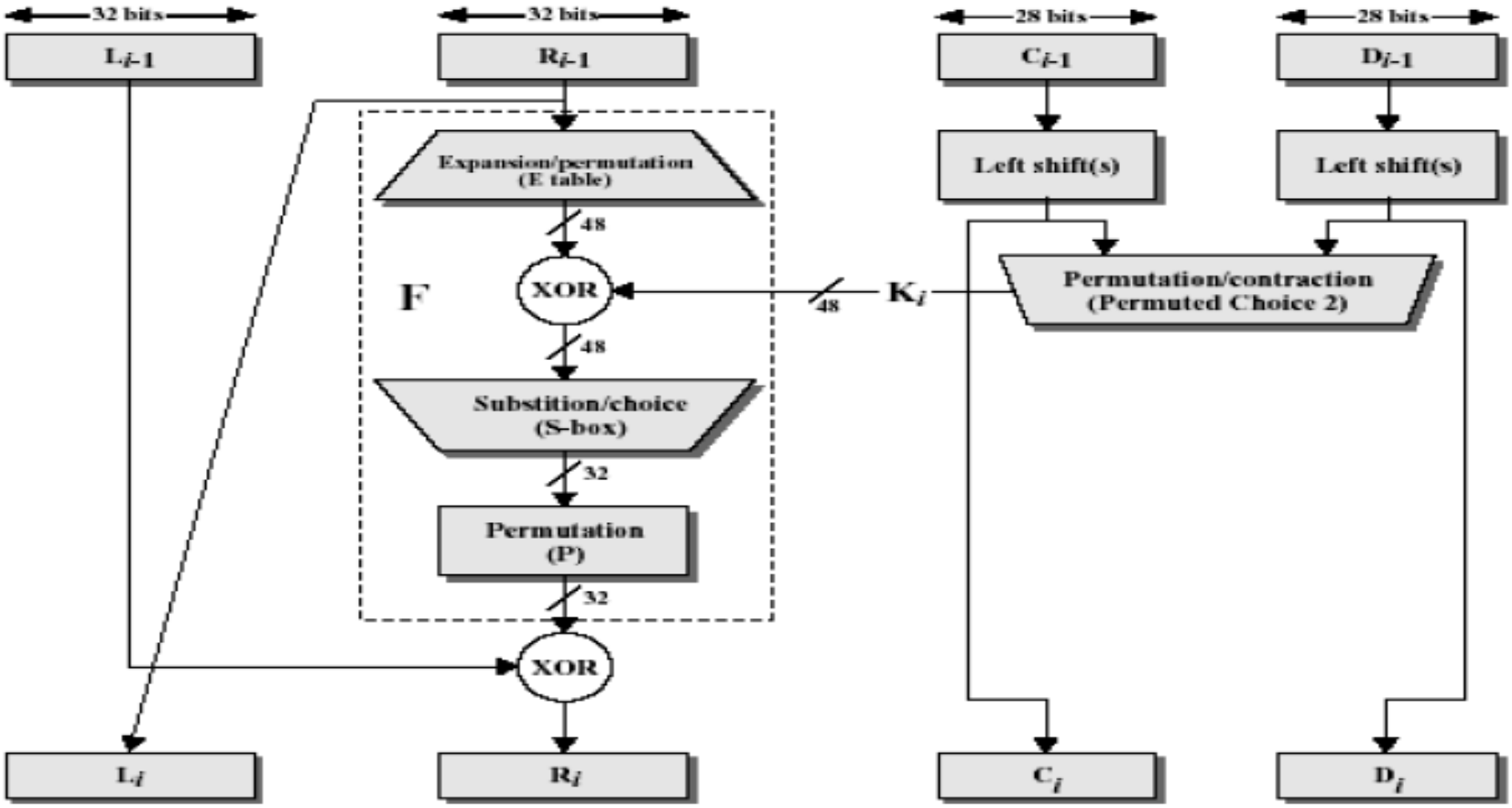
S_8

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

Les 32 bits obtenus vont être permutés, comme à l'étape 1, avec la table 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

56 bits sont extraits des 64 bits (ingorant les bits de parité)
Permutation Choice 1
Division des 56 bits en 2 moitiés de 28 bits
Chaque moitié subit un décalage LS (1 ou 2 bits)
une permutation Choice 2



(c) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Permutation des 56 bits ensuite division en 2 blocs de 28 bits chque

(a) Permuted Choice One (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

Permutation compressive Sur 56 bits permuter une sous-clé de 48 bits est sélectionnée

(b) Permuted Choice Two (PC-2)

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

(c) Schedule of Left Shifts

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Clés faibles

Dans la méthode d'extraire les sous-clés à partir de la clé initiale utilesées à chaque ronde, il se peut qu'on retrouve des clés faibles.

Exemple: Si après le décalage de chaque moitié de la clé initiale on se retrouve devant le cas où:

tous les bits = 1
ou tous les bits = 0
ou si une moitié des 1 et l'autre des 0 → clé utilisé pour toutes les rondes est la même

Selon la littérature le nombre de clés faibles environ 64 clés qui est négligeable devant le nombre total de clé possible : $2^{56} = 72057594037927936$

Exemple:
0101 0101 0101 0101 → clé faible
01FE 01FE 01FE 01FE → clé semi-faible
1F1F 0101 0E0E 0101 → potentiellement faible

L'insécurité de DES simple

- Une clé de 56 bits n'est pas suffisante pour s'assurer qu'une recherche exhaustive des clés n'est pas possible. C'est probablement pourquoi ce système de chiffrement a été adopté comme tel (72,057,594,037,927,936 clés) :
  - De nombreux processeurs peuvent déchiffrer plus de 92 milliards de clés/sec. Un grand organisme peut accomplir une recherche exhaustive en déchiffrant en parallèle à l'aide de plusieurs machines.
  - Des machines dédiées peuvent briser DES pour moins de 100 000 $ en quelques heures.
- Des versions plus fortes ont été proposées. Triple-DES est un système de chiffrement avec 112 bits de clé très utilisé dans le monde bancaire:
  - $3DES_{KK'}(M) = DES_K(DES_{K'}^{-1}(DES_K(M)))$.

Chiffrement par Blocs: modes d'opération

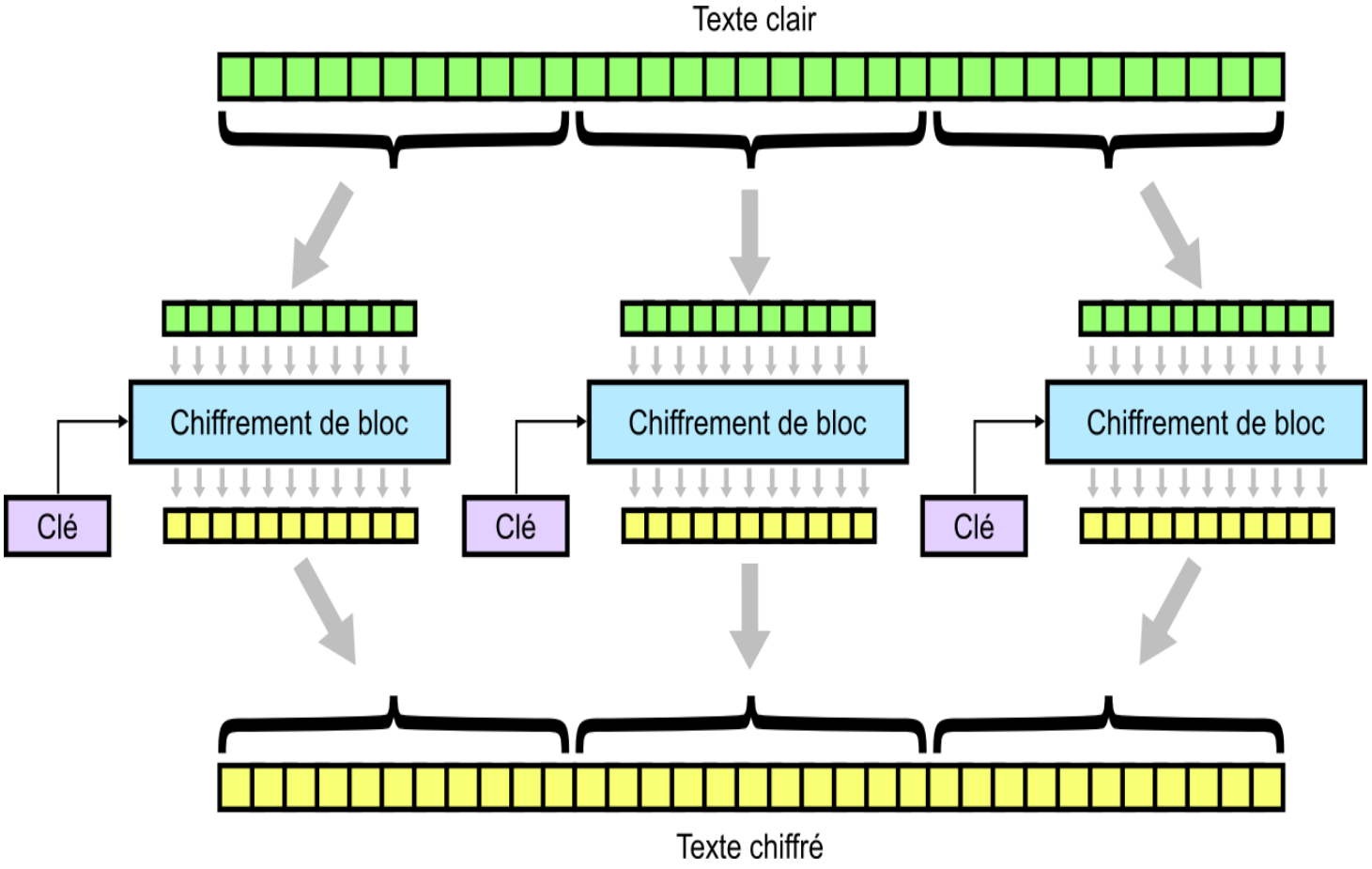
Chiffrement par Blocs

Il y a différentes méthodes d'utiliser les chiffrements par blocs, elles sont appelées « modes d'opération » (Anglais : modes of operation).
Les quatres types de chiffrement par blocs:

- Dictionnaire de codes : « Electronic codebook » (ECB)
- Enchaînement des blocs : « Cipher Block Chaining » (CBC)
- Chiffrement à rétroaction : « Cipher Feedback » (CFB)
- Chiffrement à rétroaction de sortie : « Output Feedback » (OFB)

Le chiffrement par blocs DES par exemple s'utilise avec ces modes d'opération. À noter qu'il y a d'autres modes d'opération existants.

« Electronic codebook » (ECB)



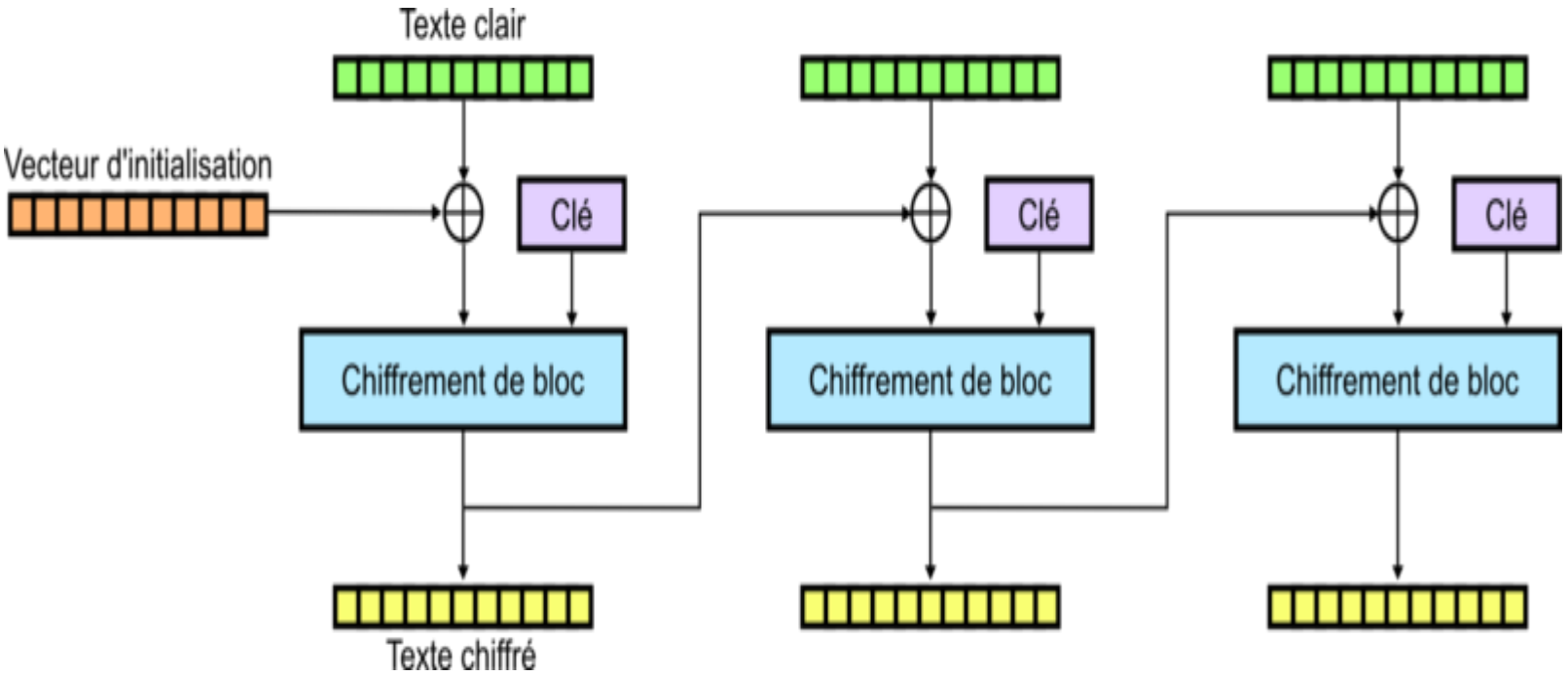
« Electronic codebook » (ECB)

Le message à chiffrer est subdivisé en plusieurs blocs qui sont chiffrés séparément les uns après les autres.

P[n]=nième bloc de texte clair.
C[n]=nième bloc de texte chiffré.
E(m)=fonction de chiffrement du bloc m.
D(m)=fonction de déchiffrement du bloc m.
Chiffrement :C[n] = E(P[n])
Déchiffrement :P[n] = D(C[n])
P et C sont d'une longueur fixe.

l'inconvénient de cette méthode est que deux blocs avec le même contenu seront chiffrés de la même manière, on peut donc tirer des informations à partir du texte chiffré en cherchant les séquences identiques. On obtient dès lors un « dictionnaire de codes » avec les correspondances entre le clair et le chiffré d'où le terme codebook.

« Cipher Block Chaining » (CBC)

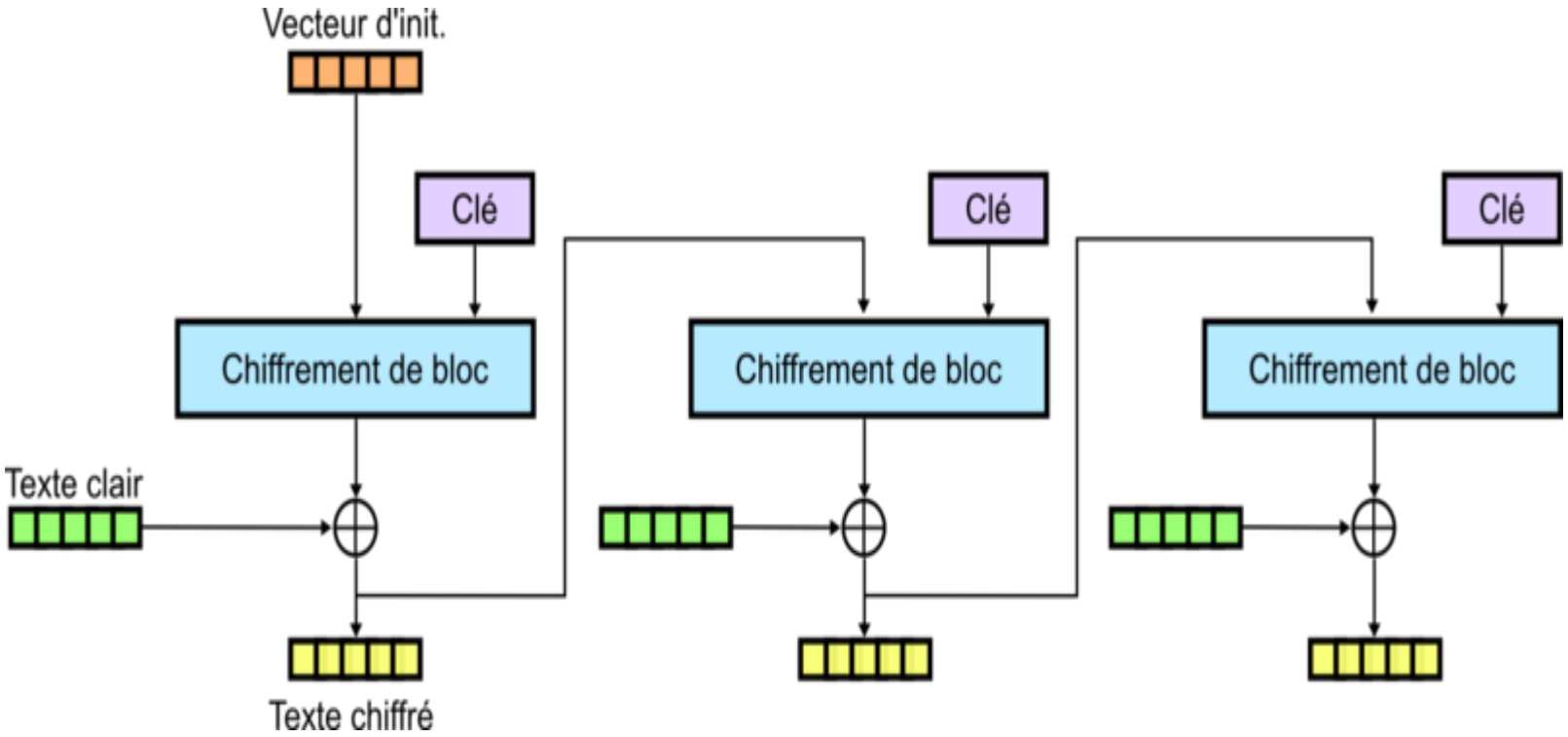


« Cipher Block Chaining » (CBC)

C'est un des modes les plus populaires. Il est une solution au premier problème du mode ECB. Avant d'être chiffré, l'opération binaire XOR est appliquée entre le bloc actuel de texte clair et le bloc précédent de texte chiffré. Pour le tout premier bloc, un bloc ayant un contenu aléatoire est généré et utilisé pour l'application de l'opération XOR, appelé « vecteur d'initialisation » (initialization vector). Ce premier bloc est envoyé tel quel avec le message chiffré.

Chiffrement: C[0] = E(P[0] xor VI), C[n] = E(P[n] xor C[n-1])
Déchiffrement: P[0] = D(C[n]) xor VIP[n] = D[C[n]) xor C[n-1] ,

« Cipher Feedback » (CFB)



« Cipher Feedback » (CFB)

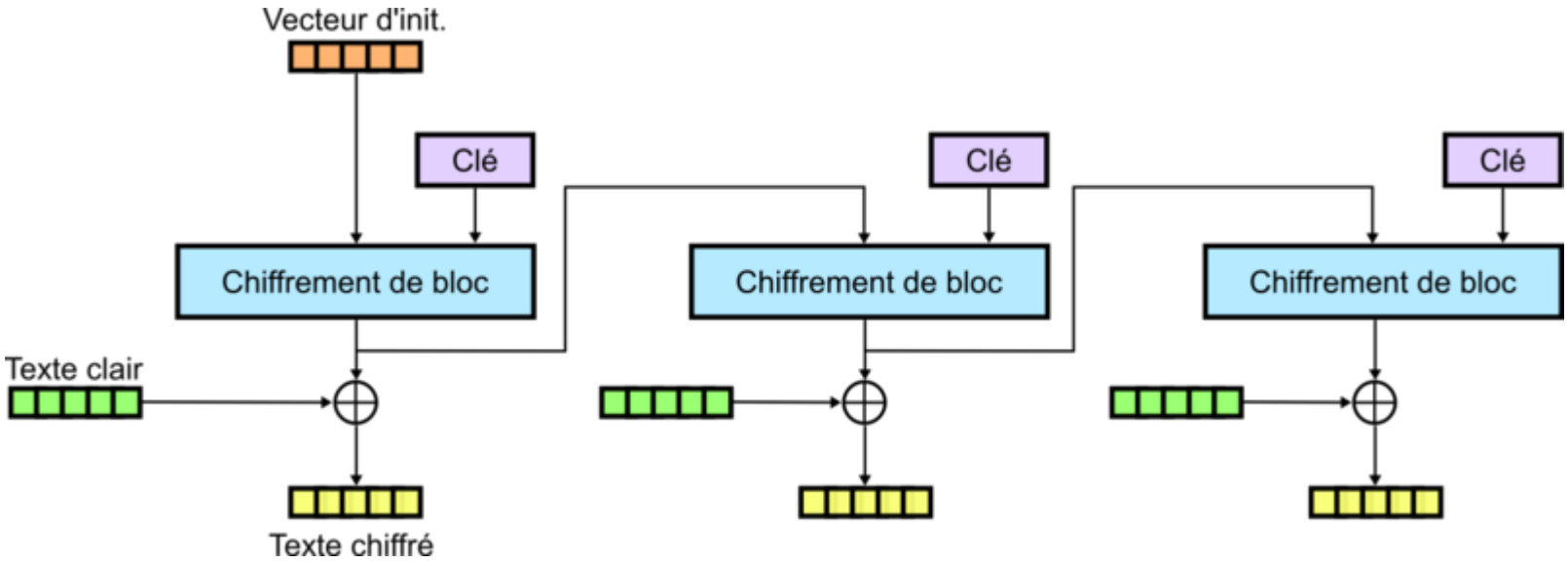
Dans ce mode l'opération XOR est appliquée entre le bloc de texte clair et le résultat précédent chiffré à nouveau par la fonction de chiffrement. Pour le premier bloc de texte clair, on génère un vecteur d'initialisation.

P[n]=nième bloc de texte clair.
C[n]=nième bloc de texte chiffré.
I[n]=nième bloc temporaire
E(m)=fonction de chiffrement et de déchiffrement du bloc m
VI=vecteur d'initialisation
^ = XOR

Chiffrement :I[0] = VI
I[n] = C[n-1] , si (n > 0)
C[n] = P[n] ^ E(I[n])

Déchiffrement :I[0] = VI
I[n] = C[n-1], si (n > 0)
P[n] = C[n] ^ E(I[n])

« Output Feedback » OFB



« Output Feedback » (OFB)

Au départ un vecteur d'initialisation est généré. Ce bloc est chiffré à plusieurs reprises et chacun des résultats est utilisé successivement dans l'application de l'opération XOR avec un bloc de texte clair. Le vecteur d'initialisation est envoyé tel quel avec le message chiffré.

P[n]=nième bloc de texte clair.
C[n]=nième bloc de texte chiffré.
I[n]=nième bloc temporaire
R[n]=nième bloc temporaire second
E(m)=fonction de chiffrement et de déchiffrement du bloc m
VI=vecteur d'initialisation
^=OU-Exclusive

Chiffrement :I[0] = VI
I[n] = R[n-1] , si (n > 0)
R[n] = E(I[n])
C[n] = P[n] ^ R[n]

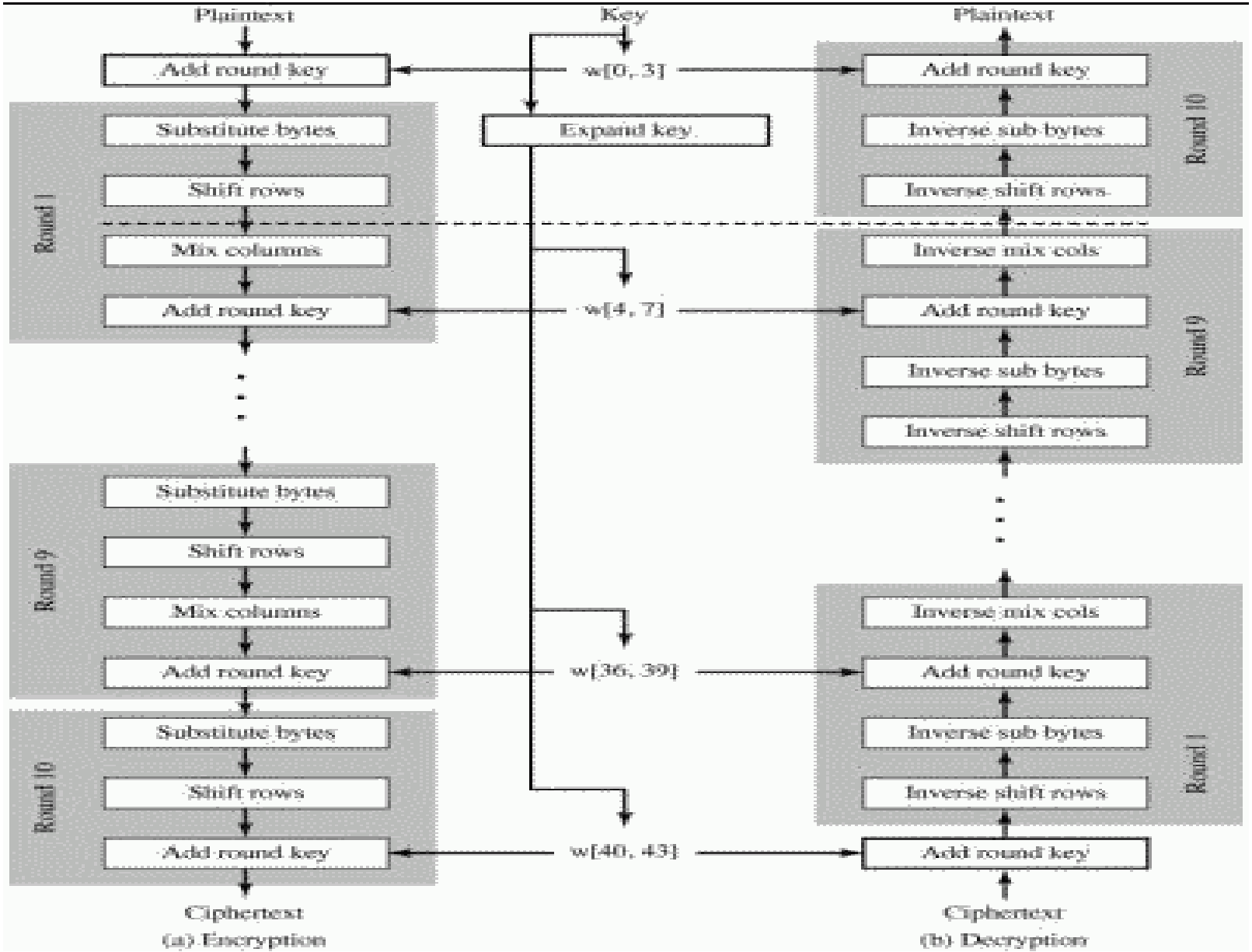
Déchiffrement :I[0] = VI
I[n] = R[n-1] , si (n > 0)
R[n] = E(I[n])
P[n] = C[n] ^ R[n]

Advanced Encryption Standard : **AES**

Advanced Encryption Standard : **AES**

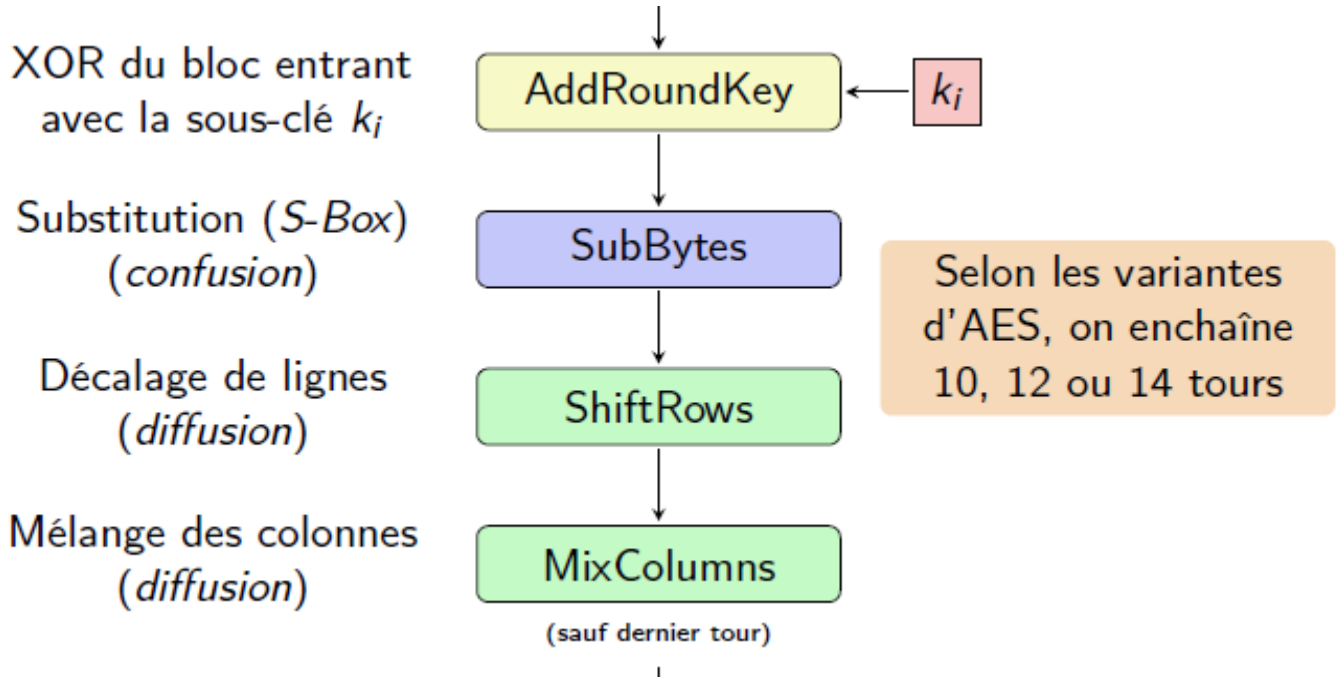
- AES est un algorithme de chiffrement symétrique par bloc.
- Il a remporté en 2000 le concours lancé par le NIST en 1997.
- Il y avait 15 participants à ce concours et c'est la proposition Rijndael, du nom de ses concepteurs belges Joan Daemen et Vincent Rijmen, qui a été retenue pour succéder au DES.
- AES est un sous-ensemble de la proposition initiale Rijndael : la taille des blocs est fixée à 128 bits (au lieu d'une taille variable multiple de 32 bits et comprise entre 128 bits et 256 bits).
- AES se décline en trois versions :
  - AES-128 ⇒ clé de 128 bits, 10 tours
  - AES-192 ⇒ clé de 192 bits, 12 tours
  - AES-256 ⇒ clé de 256 bits, 14 tours

Advanced Encryption Standard : **AES**



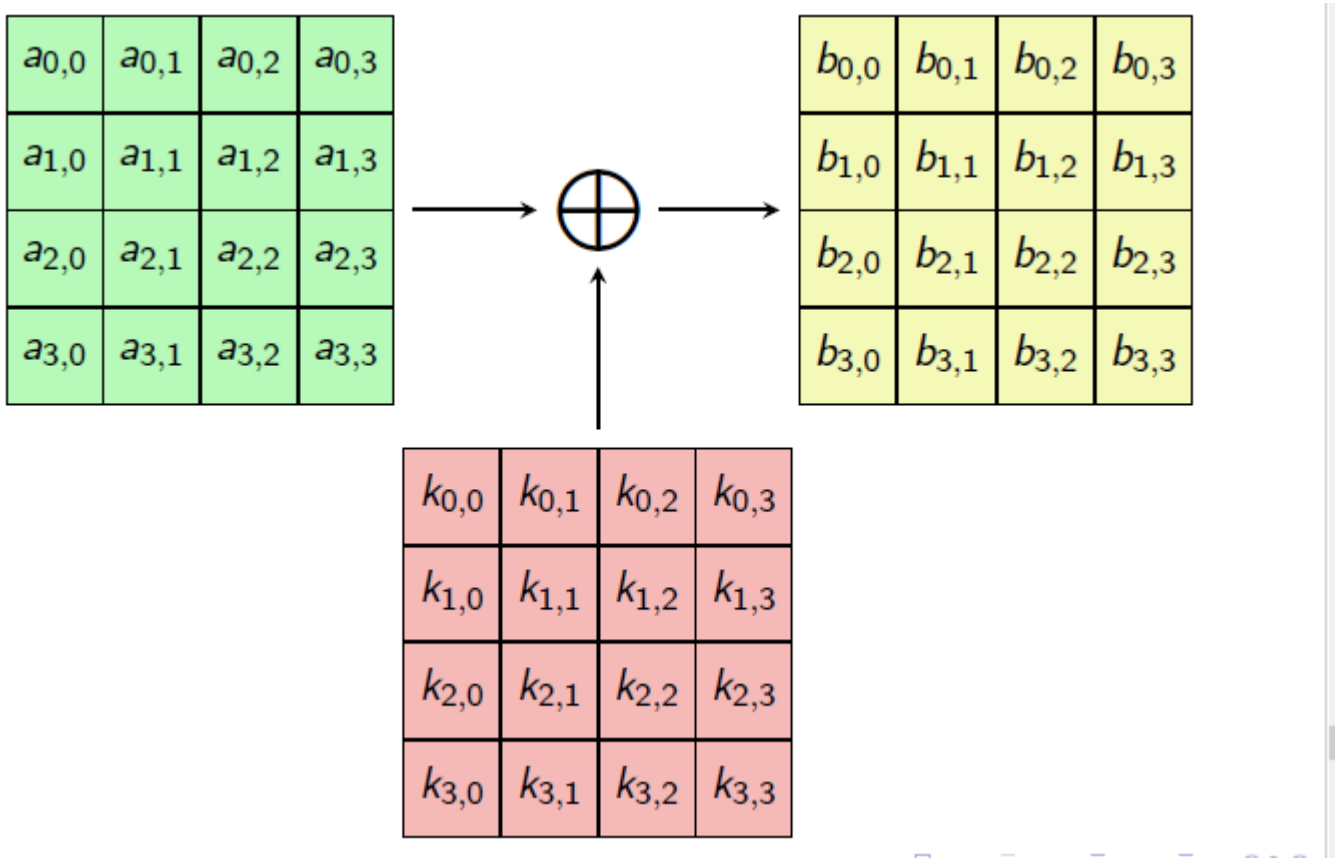
Advanced Encryption Standard : **AES**

A chaque tour, on effectue quatre opérations sur la matrice 4*4 :



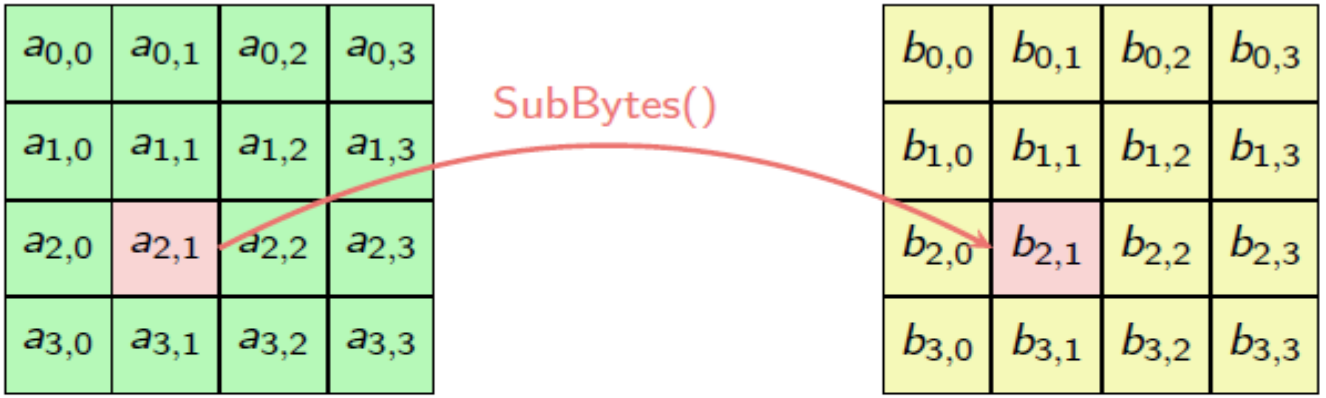
Advanced Encryption Standard : **AES**

AddRoundKey : Calcul des XOR des octets du bloc à chiffrer avec la matrice de la clé courante.



Advanced Encryption Standard : **AES**

SubByte : Substitue chaque octet du bloc à chiffrer.

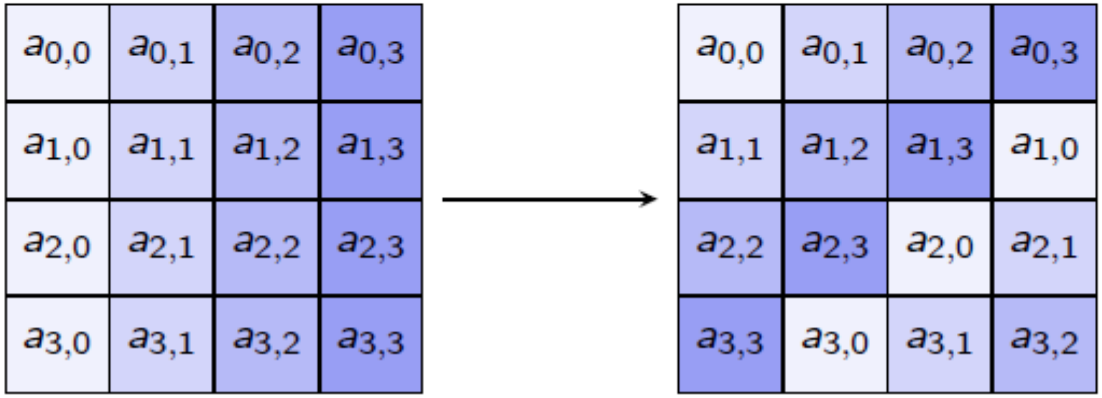


- *SubBytes* opère **indépendamment** sur chaque octet. C'est donc une application de $\{0, \ldots, 255\}$ dans lui-même.
- C'est un opérateur non-linéaire $\Longrightarrow$ *confusion*.

Advanced Encryption Standard : **AES**

Shiftrows : Applique des rotations aux rangées 2, 3, 4 de la matrice.

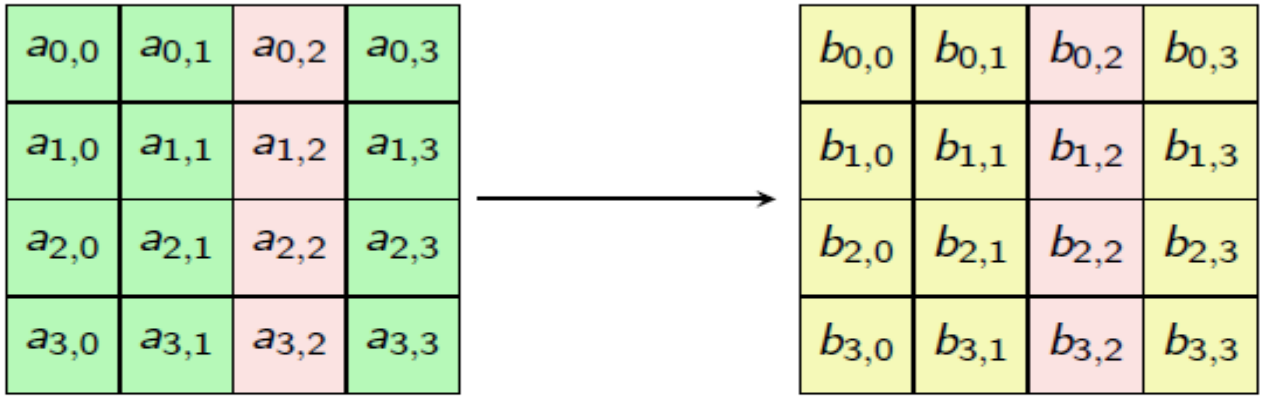
- Cet opérateur décale cycliquement les lignes d'un bloc. Chaque ligne est décalée d'une valeur différente suivant la taille du bloc sauf la ligne 0 qui reste toujours inchangée. C'est une opération de *diffusion*.
- Dans le cas du chiffrement AES-128, la ligne i est décalée de i octets vers la gauche pour $i = 1, 2, 3$:



Advanced Encryption Standard : **AES**

Mixcol : Multiplie chaque colonne du bloc courant par une matrice. Les multiplications sont dans un corps fini et les additions, des XOR.

Cet opérateur est une transformation linéaire (*diffusion*) agissant sur les colonnes de la matrice 4×4 :



Cette transformation s'écrit (multiplication dans $\mathbb{F}_{2^8}$) :

$$\begin{pmatrix} b_{0,2} \\ b_{1,2} \\ b_{2,2} \\ b_{3,2} \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_{0,2} \\ a_{1,2} \\ a_{2,2} \\ a_{3,2} \end{pmatrix}$$

Performance & Sécurité

- AES-128 est presque aussi rapide que DES. Précisément, AES-128 est 2.7 fois plus rapide que 3DES, lui-même 3 fois plus lent que DES.

- AES est donc très rapide et en outre peu gourmand en mémoire. Cela lui permet d'être efficace sur une grande variété de matériels.

- AES a été conçu pour résister aux attaques classiques comme la cryptanalyse linéaire ou différentielle.

Les chiffrements à flot

Il existe deux grandes familles de chiffrements symétriques :

Les chiffrements à flot - on génère à partir de la clé une suite chiffrant pseudo-aléatoire de même longueur que les données. On combine cette suite, par exemple avec un XOR bit-à-bit, avec les données à chiffrer.

- RC4 (Rivest, 1987) : SSL/TLS, WEP, WPA, WPA2,...
- E0 : Bluetooth
- A5 : GSM

Les chiffrements par bloc - les données à chiffrer sont découpées en blocs de taille fixe (typiquement 64 ou 128 bits). Les blocs sont chiffrés séparément et ensuite combinés selon un mode opératoire (ECB, CBC, CTR...).

- DES (IBM et NSA, 1975) - blocs 64 bits, clés 56 bits
- IDEA (Lai-Massey, 1992) - blocs 64 bits, clés 128 bits
- AES/Rijndael (Daemen-Rijmen, 1998) - blocs/clés 128, 192, 256 bits
- et aussi : RC5, RC6, Camellia,...

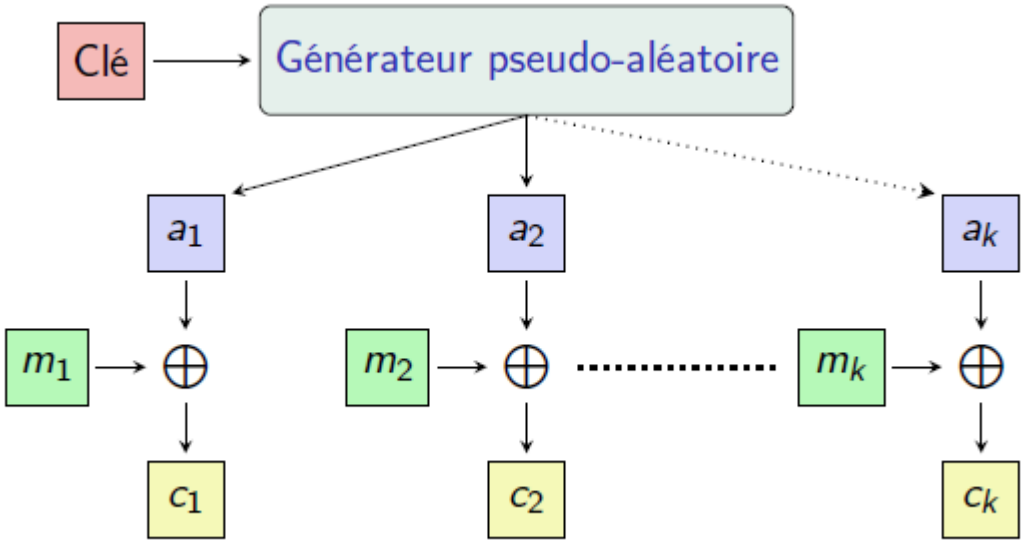
Les chiffrements à flot

Les chiffrements à flot

Le **chiffrement de flux**, **chiffrement par flot** ou **chiffrement en continu** (en anglais *stream cipher*) est une des deux grandes catégories de chiffrements modernes en cryptographie symétrique, Un chiffrement par flot arrive à traiter les données de longueur quelconque et n'a pas besoin de les découper.

Les chiffrements à flot

Des données pseudo-aléatoires, appelées flux de clé (keystream), sont générées et combinées (le plus souvent avec un XOR) aux données en clair pour produire les données chiffrées.

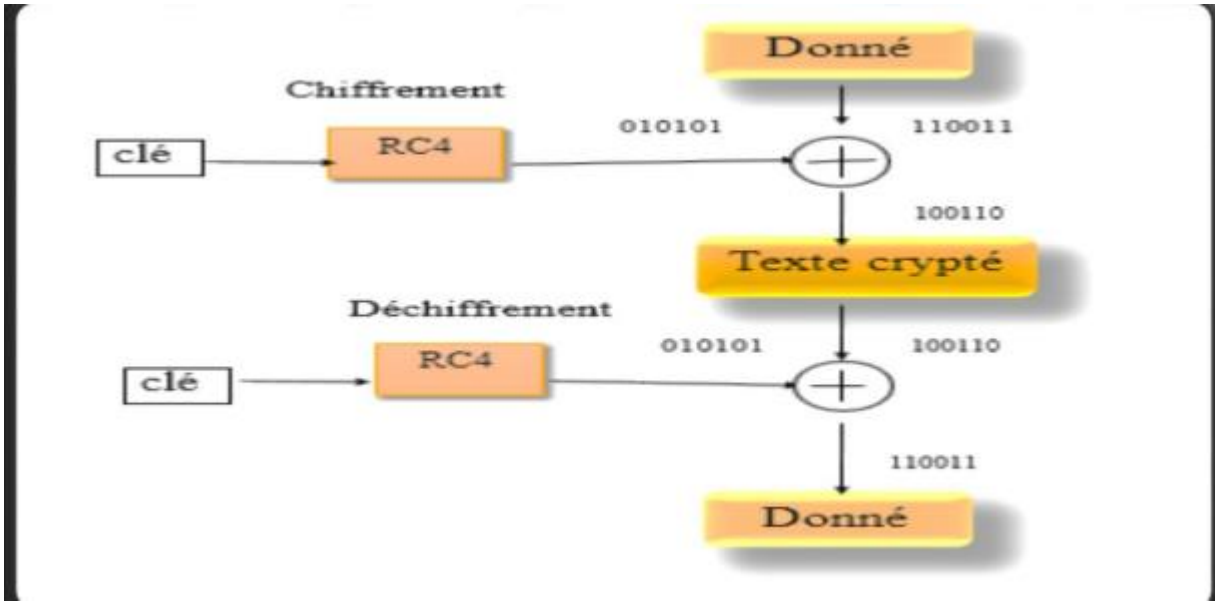


RC4

Les algorithmes de RC sont un ensemble d'algorithmes de chiffrement à clé symétrique inventé par Ronald Rivest .

- RC4 est un algorithme de chiffrement à flux le plus couramment utilisé dans le monde de la cryptographie.
- Il est connu aussi sous les noms tels que le ARC4 et ARCFOUR.
- RC4 est conçu en 1987 par Ronald Rivest pour RSA Security
- Cet algorithme est utilisé dans le SSL (également connu sous le nom de TLS), ou encore WEP

RC4 est un générateur de bits pseudo-aléatoires dont le résultat est combiné avec le texte en clair via une opération XOR, le déchiffrement se fait de la même manière



Description de l'Algorithme RC4

Phase : Initialisation

Tableau K contient N octets de la clé et S (table d'états = flux appliqué sur le texte clair) contient les nombres 0…255

Pour i=0 à 255 faire
S[i]=i
Fin
Pour j=0
Pour i=0 à 255 faire
j=(j+S[i]+K[i mod N]) mod 256
Echanger(S[i],S[j])
Fin Pour

Phase : Keystream i=j=k=0;

Pour k=0 à m faire
i=(i+1) mod 256
j=(j+S[i]) mod 256
Echanger(S[i],S[j])
t=(S[i]+S[j]) mod 256
KS[k]=S[t]
Fin Pour

Description de l'Algorithme RC4

Phase : Chiffrement & Déchiffrement Soit M un message de longueur m (octets)

Chiffrement :

Pour i=0 à m faire
Mc=M xor KS
Fin Pour

Déchiffrement :

Pour i=0 à m faire
M=Mc xor KS
Fin Pour

Caractéristiques

- Les chiffrements à flot sont très rapides, les implantations matérielles étant particulièrement efficaces.

- On chiffre à la volée sans attendre d'avoir lu, tout ou une partie, des données : bien adapté aux applications temps réel.

- Ils sont très utilisés pour la protection des données multimedia.

- Ce chiffrement est adapté du chiffrement de Vernam (théoriquement inviolable) sauf qu'ici on génère une suite pseudo-aléatoire à partir d'une clé aléatoire, généralement de petite taille comparée à la taille des données. Le chiffrement de Vernam, quant à lui, utilise une clé aléatoire à usage unique de même taille que les données à chiffrer.

- Le chiffrement de Vernam est certes sûr mais, en pratique, très difficile à mettre en oeuvre. Le chiffrement à flot peut être vu comme un "pseudo-Vernam" adapté à un usage concret.

Table de comparaison

	Chiffrement par bloc	Chiffrement par flot
Définition	Convertit le texte brut en prenant son bloc à la fois.	Convertit le texte en prenant un octet du texte brut à la fois.
Complexité	Simple	Complexe
Implémentation	Feistel Cipher	Vernam Cipher
Réversibilité	Inverser le texte crypté est difficile.	Il utilise XOR pour le cryptage qui peut facilement être inversé au texte brut.
Modes d'algorithme utilisés	ECB (Electronic Code Book) CBC (Cipher Block Chaining)	CFB (Cipher Feedback) OFB (Output Feedback)
Nombre de bits utilisés	64 bits ou plus	8 bits