Sécurité Informatique

Chapitre 2 :

Cryptographie symétrique

(Moderne)

Guellil zouaoui

plant

- Introduction
- Cryptographie symétrique
  - Cryptographie Classique
  - Techniques de la Substitution
  - Techniques de la Transposition

RC4

- Mode de chiffrement par flot (Stream Cypher Mode)
- clé de longueur variable.
- La clé partager est utilisé pour générer des bits pseudo-aléatoires qui produit un flux imprévisible sans connaissance de la clé.
- la sortie du générateur est appelée flux de clés, est combinée pour le de/chiffrement de flux de texte clair en utilisant XOR opération.

RC4 (Principe)

- Génération de clé
- Chiffrement

RC4 (Génération de clé 1/2)

- la clef RC4 permet d'initialiser un tableau de 256 octets (full version)
- la clef autant de fois que nécessaire pour remplir le tableau.
- des opérations très simples sont effectuées pour mélanger autant que possible le tableau.

Pseudo code **Génération de la permutation:**

```
pour i de 0 à 255   S[i] := i
j := 0
pour i de 0 à 255
    j := (j + S[i] + clé[i mod longueur_clé]) mod 256
    échanger(S[i], S[j])
finpour
```

RC4 (Génération de clé 2/2)

- **Le tableau S peut ensuite être utilisé pour générer un flux grace à des opérations de déplacement de valeurs et d'opération XOR**

```
i := 0
j := 0
tant_que générer une sortie:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    échanger(S[i], S[j])
    octet_chiffrement = S[(S[i] + S[j]) mod 256]
    result_chiffré = octet_chiffrement XOR octet_message
fintant_que
```

RC4 (Example)

- Version simplifier utilise un tableau de 8 valeur de 0 a7.
- Pt [3,7,1,0], Key =[5,1,2]

********_____ **Génération de la permutation (Key shedule)** _____********

[0,1,2,3,4,5,6,7]
[5,1,2,3,4,0,6,7]
[5,7,2,3,4,0,6,1]
[5,7,3,2,4,0,6,1]
[5,7,2,3,4,0,6,1]
[5,7,2,3,1,0,6,4]
[5,0,2,3,1,7,6,4]
[5,0,2,3,6,7,1,4]
[5,4,2,3,6,7,1,0]

********_____ **générer le flux et chiffrement** _____********

i	j	S[i]	S[j]	S[(S[i]+S[j])%8]	S	Cipher text
1	4	6	4	2	[5,6,2,3,4,7,1,4]	$(3)_{10}= (011)_2$; $(2)_{10}= (010)_2$ 3 XOR 2; [1,]
2	6	1	2	3	[5,6,1,3,4,7,2,6]	$(7)_{10}= (111)_2$; $(3)_{10}= (011)_2$ 7 XOR 3 ; [1,4,]
3	1	6	3	3	[5,3,1,6,4,7,2,1]	$(1)_{10}= (001)_2$; $(3)_{10}= (011)_2$ 1 XOR 3 ; [1,4,2,]
4	5	7	4	6	[5,3,1,6,7,4,2,5]	$(0)_{10}= (000)_2$; $(6)_{10}= (110)_2$ 1 XOR 3 ; [1,4,2,6]

RC4 (Example 2)

- Pt [7, 1, 3, 4, 6, 2, 1, 0, 5, 3], Key =[5,1,2]

******** **Génération de la permutation (Key shedule)** ********

[0,1,2,3,4,5,6,7]
[5,1,2,3,4,0,6,7]
[5,7,2,3,4,0,6,1]
[5,7,3,2,4,0,6,1]
[5,7,2,3,4,0,6,1]
[5,7,2,3,1,0,6,4]
[5,0,2,3,1,7,6,4]
[5,0,2,3,6,7,1,4]
[5,4,2,3,6,7,1,0]

******** **générer le flux et chiffrement** ********

it	i	j	S[i]	S[j]	S[(S[i]+S[j])%8]	S	Cipher text
0	1	4	6	4	2	[5,6,2,3,4,7,1,0]	[5,]
1	2	6	1	2	3	[5,6,1,3,4,7,2,0]	[5,2]
2	3	1	6	3	3	[5,3,1,6,4,7,2,0]	[5,2,0]
3	4	5	7	4	6	[5,3,1,6,7,4,2,0]	[5,2,0,2]
4	5	1	3	4	0	[5,4,1,6,7,3,2,0]	[5,2,0,2,6]
5	6	3	6	2	5	[5,4,1,2,7,3,6,0]	[5,2,0,2,6,7]
6	7	3	2	0	1	[5,4,1,0,7,3,6,2]	[5,2,0,2,6,7,0]
7	0	0	5	5	1	[5,4,1,0,7,3,6,2]	[5,2,0,2,6,7,0,1]
8	1	4	7	4	0	[5,7,1,0,4,3,6,2]	[5,2,0,2,6,7,0,1,5]
9	2	5	3	1	4	[5,7,3,0,4,1,6,2]	[5,2,0,2,6,7,0,1,5,7]

- Strengths of RC4
  - The difficulty of knowing where any value is in the table.
  - The difficulty of knowing which location in the table is used to select each value in the sequence.
  - Encryption is about 10 times faster than DES.
- Limitations of RC4
  - RC4 is no longer considered secure.
  - One in every 256 keys can be a weak key. These keys are identified by cryptanalysis that is able to find circumstances under which one of more generated bytes are strongly correlated with a few bytes of the key.
  - A particular RC4 Algorithm key can be used only once.
- Performance
  - Each of the UDI implementations is a hardware block specifically designed for the implementation. RAM space is required by the key byte generator to locally maintain the state table for key generation. This state would need to be preserved and restored in case of a context switch if other processes would need the same functionality. This overhead is not considered in the above performance projections. Encryption and decryption state data may be stored in separate state memories to allow for independent processes.
  - The following table summarizes the number of MIPS required for the algorithm encryption/decryption for 1 million bits per second for each of the three implementations.

Chiffre symétrique moderne

- Le **principe de Kirchhoff :**la sécurité d'un crypto système ne doit reposer que sur le secret de la clef.
- Claude Shannon : « l'adversaire connaît le système »
- Diffusion : cacher les propriétés statistiques
- Confusion : cacher la relation entre la clé et les messages cryptés

- La substitution ajoute de la confusion au procédé de chiffrement
- la transposition ajoute de la diffusion en éparpillant l'influence moyenne de chaque bit du clair, sur les bits du chiffré.
- La combinaison de ces deux transformations permet un chiffrement plus robuste.
- On appelle opérateur de Shannon ou « **round** » l'application de ces deux opération (substitution et transposition).
- Ce mode de chiffrement est appelé **par produit.**

Chiffrement par bloc

- Diviser le texte en blocs relativement gros, typiquement de **64 ou 128 bits**.
- Coder chaque bloc séparément.
- La même clé de chiffrement est utilisée pour chaque bloc.
- La clé de chiffrement détermine l'ordre dans lequel la substitution, le transport et d'autres fonctions mathématiques sont effectuées sur chaque bloc.

Chiffrement idéal

(Le masque jetable one pad time)

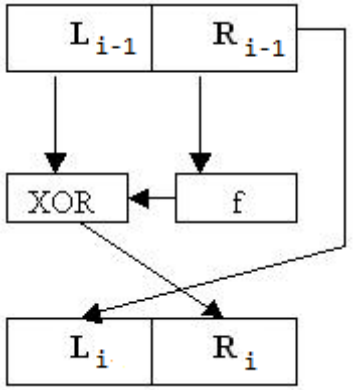
- Taille clé = taille msg
- Clé aléatoire et unique (utilisé une seul foi).
- Cette méthode de chiffrement est inviolable.
- Très difficile à mettre en place en pratique et donc peu utilisée.
  - Génération des clé aléatoire.
  - Distribution des clés.

Schémas de Feistel

- Clé secrète K, relativement petite (de 80 à 128 bits,
- Chiffrement **itératif** : Pour chaque bloc, on itère *r* fois une fonction interne *F(Sub, permu, XOR)* .
- à chacun des *r* tours, la fonction *F* est paramétrée par une clef K_i, dérivé de la clé K
- Pour déchiffrer, il suffit d'utiliser le même processus à *r* tours en inversant l'ordre des clefs K_i

schema de Feistel

- C’est une fonction de chiffrement (et déchiffrement) qui opère sur des blocs de longueur 2t en n tours.
- Chiffrement: un bloc de texte en clair est découpé en deux; la transformation de ronde est appliquée à une des deux moitiés, et le résultat est combiné avec l'autre moitié par ou exclusif. Les deux moitiés sont alors inversées pour l'application de la ronde suivante.
- Déchiffrement: est structurellement identique au chiffrement.



schema de Feistel

- Chiffrement
  - On divise le texte clair p en deux moities L_0R_0.
  - On définit pour chaque tour i = 1 à n : $L_iR_i = R_{i-1}(L_{i-1}$ **XOR** $f_{ki}(R_{i-1}))$
  - E_k (pt)= L_nR_n

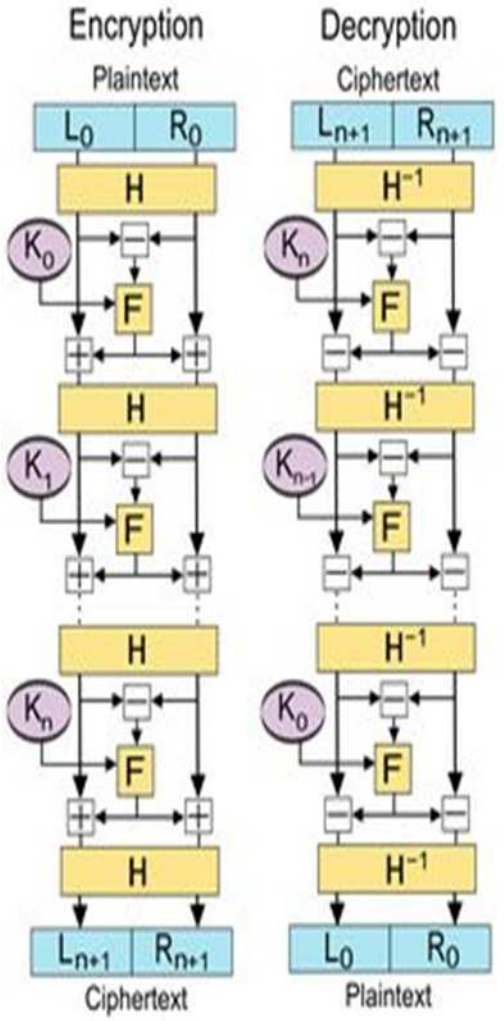
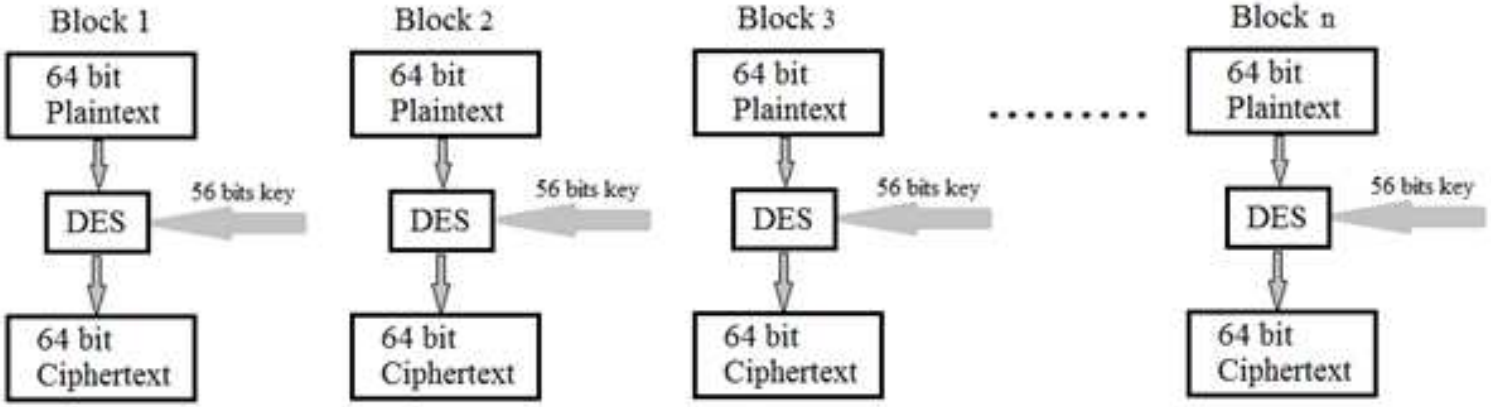


schéma de Feistel

- Déchirement
  - $R_{i-1}L_{i-1} = L_i(R_i$ **XOR** $f_{ki}(L_i))$
- Théorème de Sécurité des Réseaux Feistel: Si une fonction aléatoire sûreest utilisée pour trois tours de Feistelavec trois clés indépendantes, on obtient alors une fonction pseudo aléatoire avec des permutations pseudo aléatoire (Luby-Rackoff, 1985).

Data encryption standard (DES)

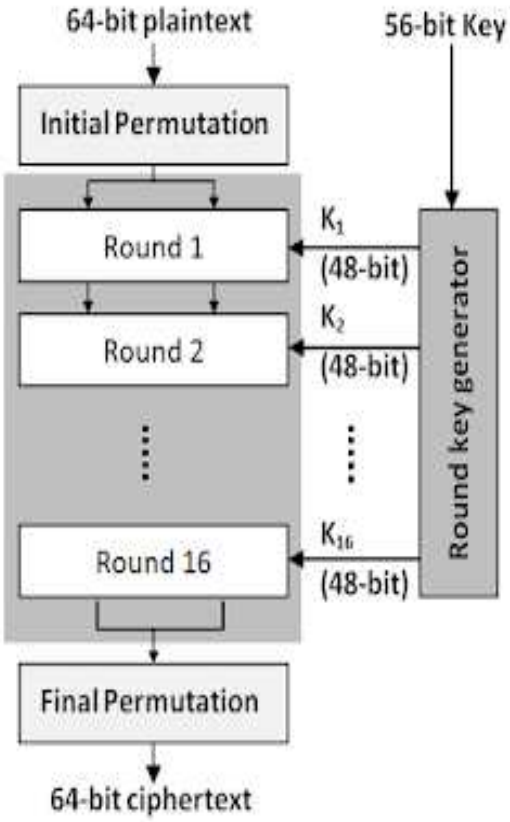
- DES est un algorithme de chiffrement a clé symétrique par blocs basé sur la structure de Feistel.
- Le même algorithme et la même clé sont utilisés pour le chiffrement et le déchiffrement, avec des différences mineures. La longueur de clé est de 56 bits.
- Il utilise des blocs de données de 64 bits, c-àdire 64 bits de texte brut sont transmis en entrée à DES, qui produit 64 bits de texte chiffré.



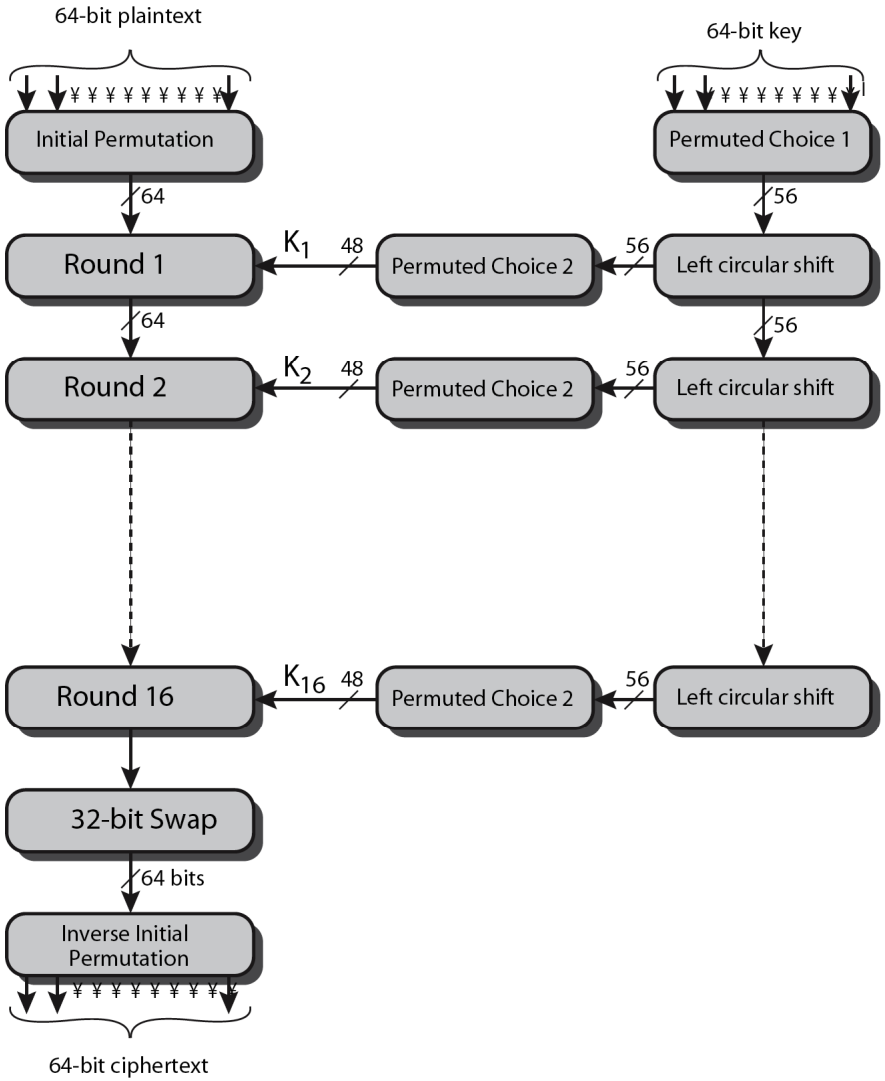
Conceptual working of DES

Structure Globale

- permutation initiale et fixe d'un bloc (sans aucune incidence sur le niveau de sécurité) ;
- le résultat est soumis à 16 itérations d'une transformation F, ces itérations dépendent à chaque tour d'une autre clé partielle de 48 bits.
- Cette clé de tour intermédiaire est calculée à partir de la clé initiale de l'utilisateur (grâce à un réseau de tables de substitution et d'opérateurs XOR).
- Lors de chaque tour, le bloc de 64 bits est découpé en deux blocs de 32 bits, et ces blocs sont échangés l'un avec l'autre selon un schéma de Feistel.
- Le bloc de 32 bits ayant le poids le plus fort (celui qui s'étend du bit 32 au bit 64) subira une transformation ;
- le résultat du dernier tour est transformé par la fonction inverse de la permutation initiale.



Structure Globale



permutation initiale (PI)

- Les 64 bits du bloc d'entrée subissent une permutation où le premier bit sera le bit 58, le second le bit 50 et le dernier bit en sortie sera la 7émé selon la matrice suivante :

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

Fonction itérative DES

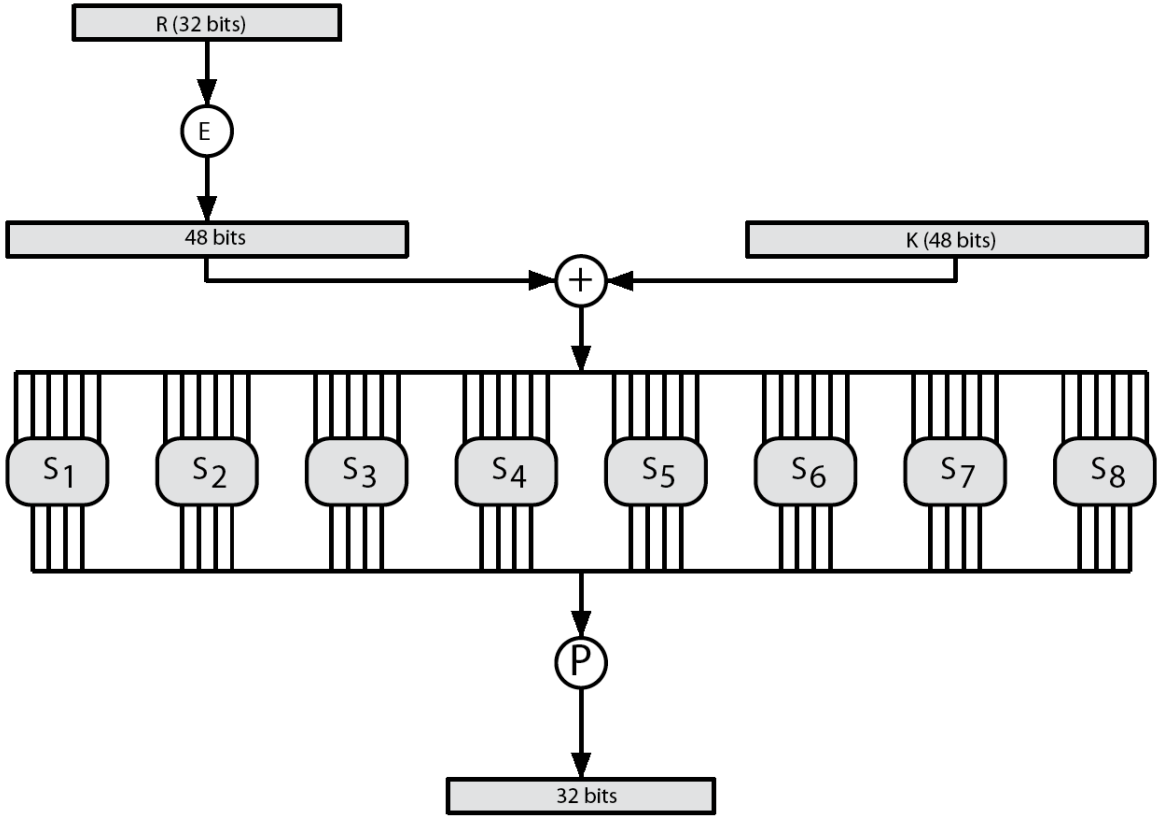
Chaque itération est constituée:

– d'une expansion de R_i de 32 bits vers 48 bits, (table d'*expansion)*

– d'un XOR avec 48 bits dérivés de la clef K_i,

– de l'application de 8 sous-fonctions de 6 bits vers 4 bits : les boîtes S

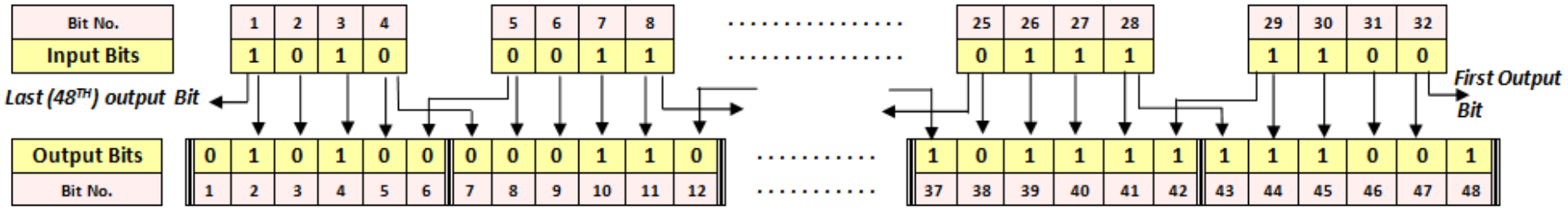
– d'une permutation des 32 bits sortants



Expansion

- Les bite du bloc R_i sont réorganisé de sort à obtenir 48 bits à partir de 32.
- Le premier bit en sortie est le 58émé en entré,
- dupliquer certain bit
- ex :4éme et 5éme

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



boîtes de substitutions (S-box)

- Les 8 Boîtes S du DES. Elles prennent 6 bits en entrée et en sortent 4.
- Composé de 4 ligne indexé de 0 a 3 (2 bit)
- et 16 colonne 0 a 15 (4 bit).
- Utilisation des boîtes-S:
- Pour une suite binnaire B=$b_1b_2b_3b_4b_5b_6$
- s(B) est le mot binaire situé à la ligne b_1b_6 et à la colonne $b_2b_3b_4b_5$.

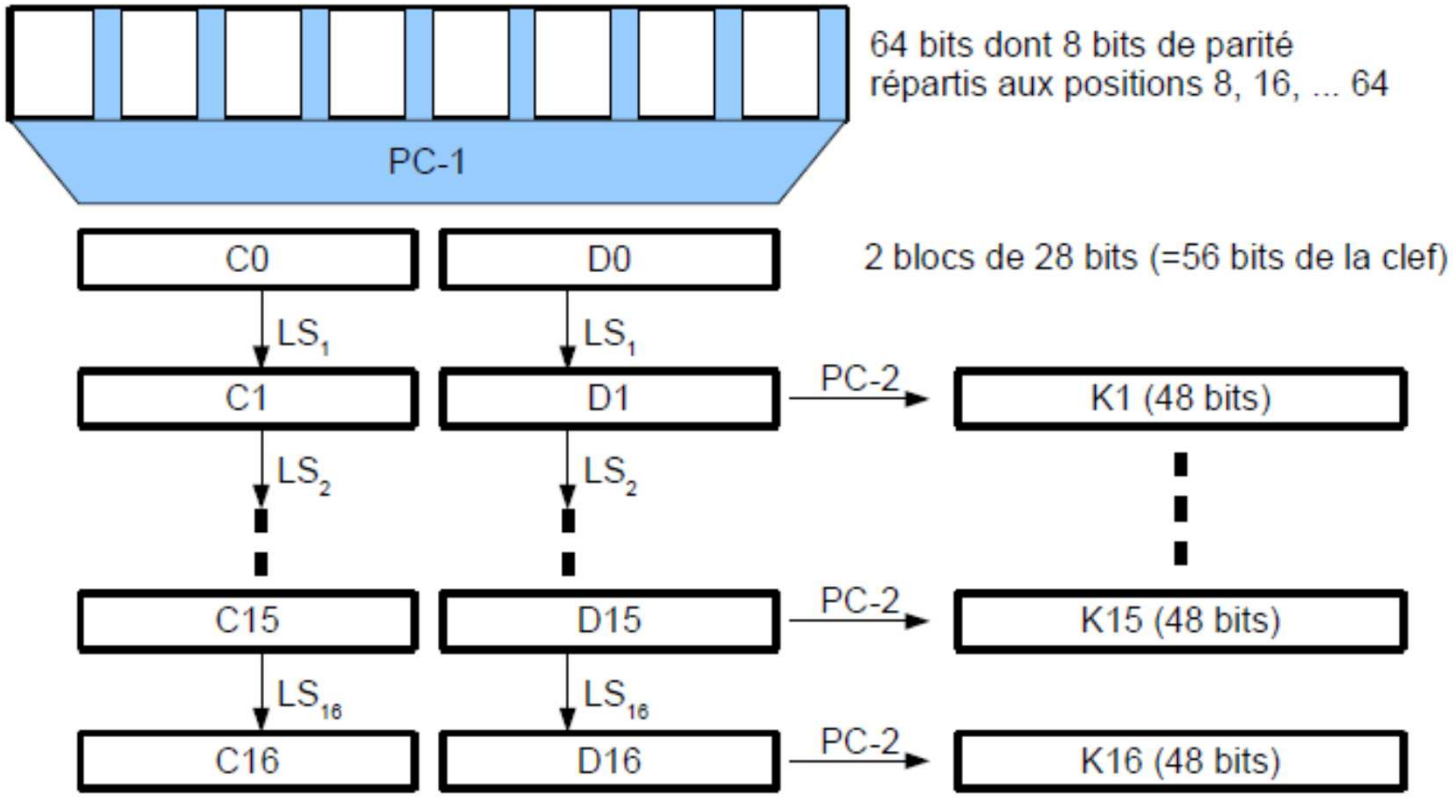
boîtes de substitutions (S-box)

- Exemple: $S_1(111001)$:
  - $(11)_2$=3éme Ligne
- $(1100)_2$ = colonne 12
- $S1(111001)=(10)_{10}=(1010)_2$

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

Génération des clefs

- Cette clé de 56 bits est dérivée d'une clé de 64 bits avec 8 bits de parité,
- chacun des 16 tours de l'algorithme utilise une sous-clé unique de 48 bits qui est dérivée de la clé de 56 bit.



Génération des clefs

- Réduire la clé de 64 bit à 56 bits on appliquant la transformation (PC-1) en supprimant les bits de parité (8, 16, ..., 64)
- Ex: K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001
- K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

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

Génération des clefs

- la clé transformer (56 bits) est divisé en deux moitiés gauche et droite C0, D0 chacune de 28 bits.
- $\boldsymbol{C_0}$ = 1111000 0110011 0010101 0101111
  $\boldsymbol{D_0}$ = 0101010 1011001 1001111 0001111
- Générer 16 blocs de C_n, D_n ($1 \leq n \leq 16$) en appliquant le **décalage** circulaire à **gauche**
- Nombre de décalage est 1 pour n= 1,2,9 et 16
- Nombre de décalage est 2 pour les autre tour.

Génération des clefs (Exemple)

- C_0 = 11110000110011001010101011111
  D_0 = 01010101011001100111100011111
- C_1 = 11100001100110010101010111111
  D_1 = 10101010110011001111000111110
- C_2 = 11000011001100101010101111111
  D_2 = 01010101100110011110001111101
- C_3 = 00001100110010101010111111111
  D_3 = 01010110011001111000111101001
- C_4 = 00110011001010101011111111100
  D_4 = 01011001100111100011110101001
- …etc.
- C_{16} = 11110000110011001010101011111
  D_{16} = 01010101011001100111100011111

Génération des clefs

- La clé K_i est formé par combinaison des partie C_iD_i

et d'appliquer la transformation PC-2 pour réduire la clé de 56bit à 48bit.

- $\boldsymbol{C_1D_1}$ = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110
- ***K1*** = 000110 110000 001011 101111 111111 000111 000001 110010

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

DES: sécurité

- Aucune propriété du message source ne sera détectable
- Effet d'avalanche : Une légère modification du texte chiffré ou de la clef => de grosses modifications du texte chiffré
- la bonne résistance à la cryptanalyse différentielle a démontré que les tables a prouvait que les sbox étaient bien conçues.

DES: Attaques

- L'espace des clefs dans DES est relativement petit (2^{56} clefs possibles).
- Du fait de la faible taille de sa clé DES peut maintenant être cassé par la recherche exhaustive de toutes les clefs.
- En 1998,le défi «DES Challenge» a été lancé pour casser DES: la machine “DeepCrack” (spécialement conçu pour attaquer DES) a réussi en quelques jours à retrouver la clé par une attaque exhaustive.

DES: Les clefs faibles

- Clefs « faibles » : telles que $E_k(E_k(x)) = x$. Il en existe 4.
- Clefs « semi-faibles » : ce sont les paires de clefs (K1,K2) dont la deuxième peut décrypter un message encrypté par la première. Ce sont les clés telles que $E_{K1}(E_{K2}(x)) = x$. Il existe six paires de ce genre.
- Clés « pouvant être faibles » : le problème est similaire aux clés semi-faibles. Il en existe 48.

DES: Amélioration

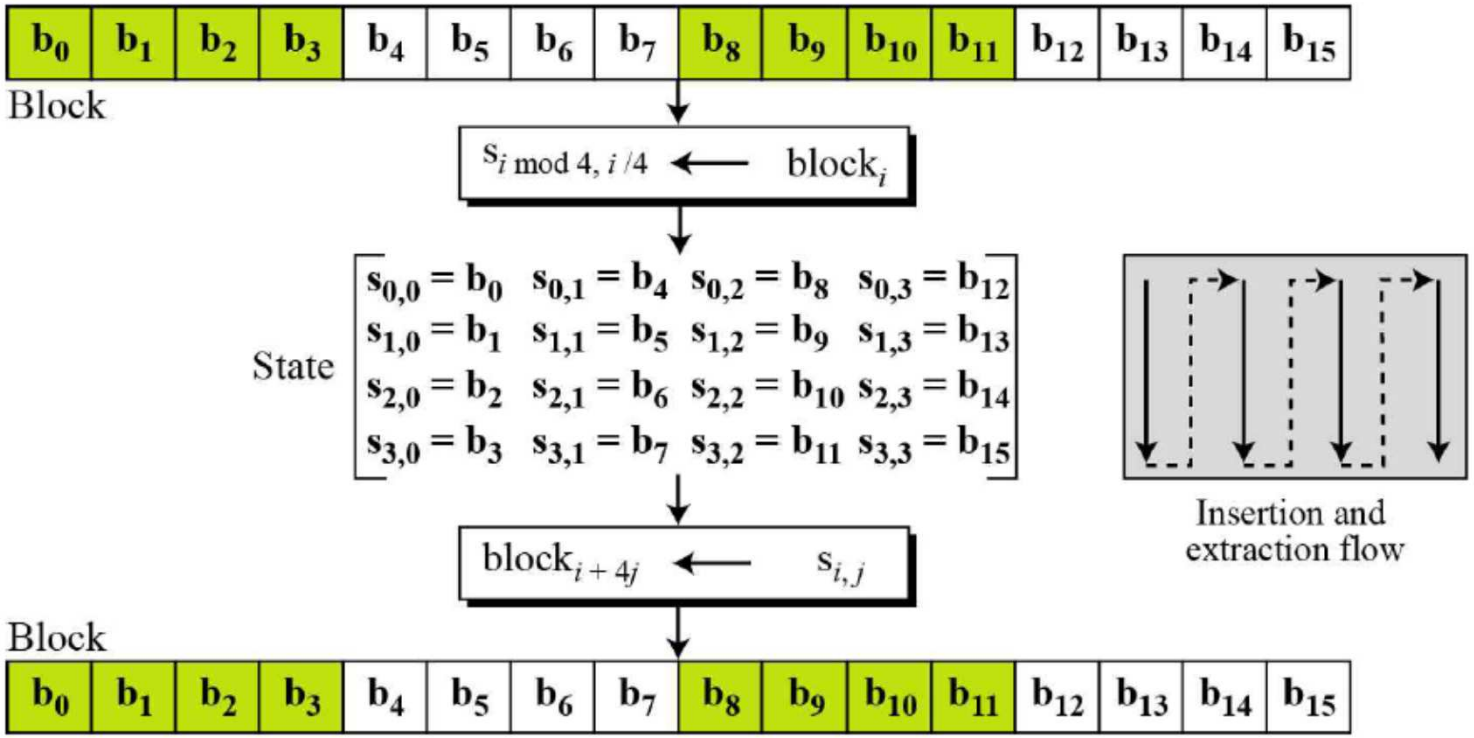
- Pour avoir une longueur de clef plus importante, le DES a été remplacé par le triple DES (3DES), qui est composé de 3 applications successives du DES, avec en entrée 64 bits de données et 2 ou 3 clefs DES différentes.
- assez simple à implémenter,
- sa force effective est de 112 bits et non pas 168 bits (56bit*3)
- plus lent à exécuter (3 fois le temps d'exécution d'un DES).
- Ceci a poussé le NIST à lancer un appel d'offre pour un algorithme à clef privée de longueur plus grande qui peut être considéré comme sûr pour tout échange d'information. => AES

Advanced Encryption Standard (AES)

- Adopté comme standard en Novembre 2011, conçu par *Johan Daemen* et *Vincent Rijmen*2 *(*d'où son nom original *Rijndael).*
- Techniques semblables à DES (substitutions, permutations, XOR…) complémentées par des opérations algébriques simples et très performantes.
- Contrairement à DES, AES est issu d'un processus de consultation et d'analyse ouvert à des experts mondiaux.
- Il s'agit également d'un *block cipher itératif* (comme DES) mais pas d'un *Feistel Cipher*
- Blocs *Plaintext/Ciphertext*: 128 bits.
- Clé de longueur variable: 128, 192, ou 256 bits.
- L'unité de données fondamentale dans l'algorithme AES est l'octet.
- LE jeu d'instructions des processeur X86 a était enrichie depuis 2008 pour prend en charge les opérations de chiffrement et de déchiffrement de l'Advanced Encryption Standard (AES).

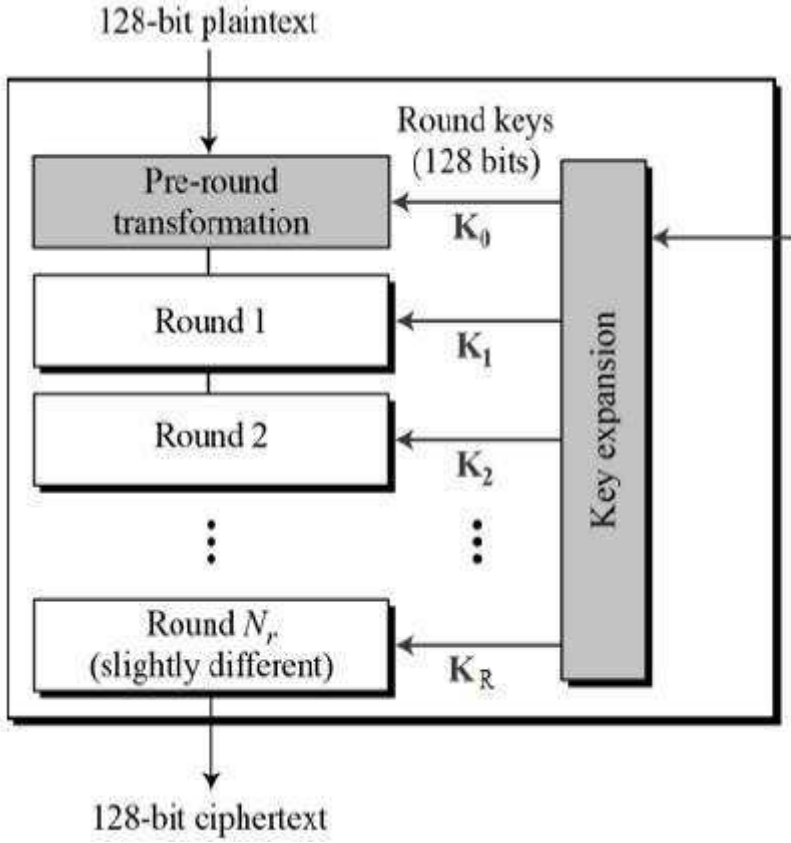
AES : définition

- L'unité de données fondamentale dans l'algorithme AES est l'octet.
- L'entrée, la sortie et la clef secrète sont traitées en interne sur une table à deux dimensions d'octets appelée ***État/state***.
- L'état se compose d'une matrice de quatre rangées et quatre colonnes d'octets.
- La donnée d'entrée est tout d'abord copiée dans la table d'état en utilisant les conventions de la Figure suivante:



AES : Principe

- Dérivez l'ensemble des clés rondes de la clé de chiffrement. (***Key Schedule***)
- Initialisez le tableau d'état avec les données de bloc (texte en clair).
- Pre-round : consiste à appliquer XOR entre la table d'État et la clef de chiffrement.
- Effectuez cycles de manipulation de l'État round $1..N_{r-1}$.
- Effectuez le dernier cycle de manipulation de l'État.
- Copiez le tableau d'état final en tant que données chiffrées (texte chiffré).

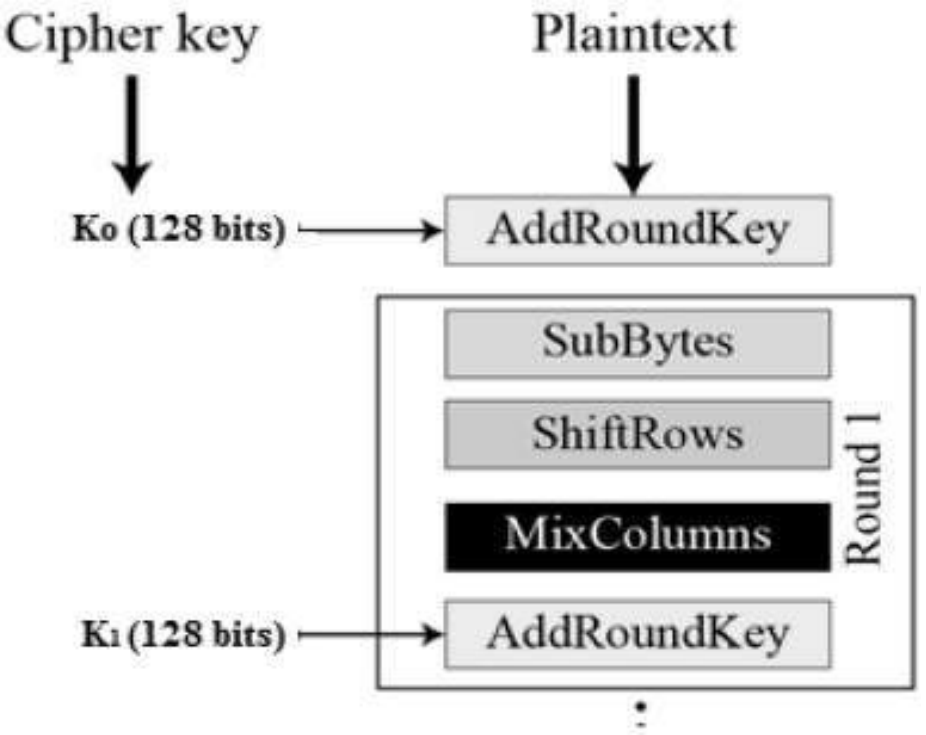


R	Key size
10	128
12	192
14	256

Relationship between number of rounds (R) and cipher key size

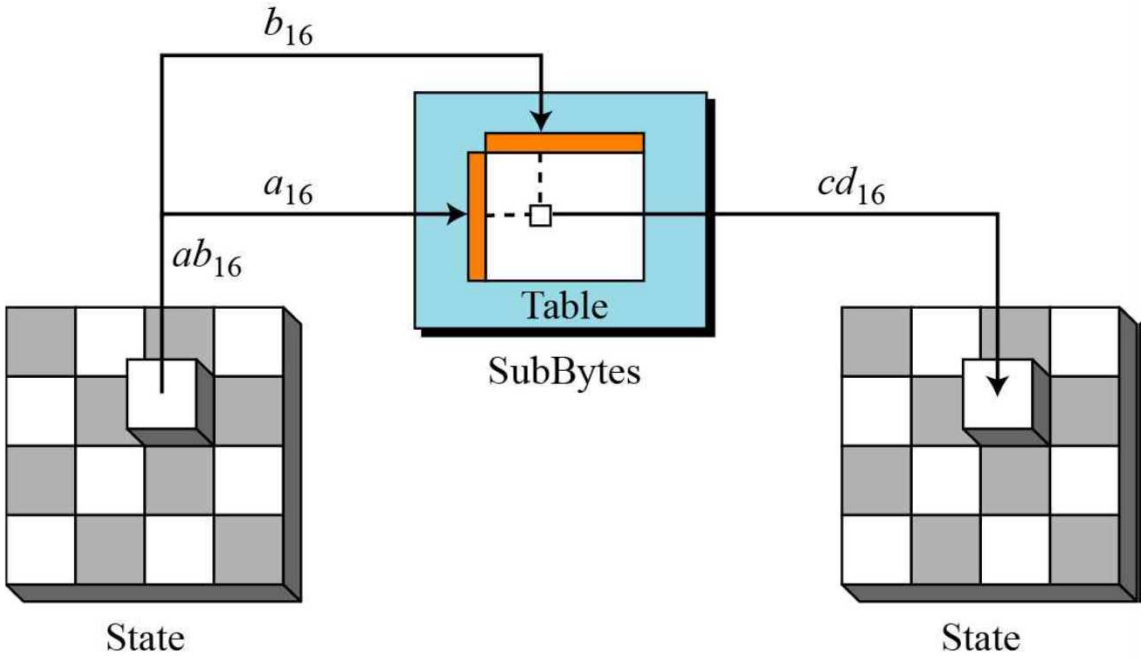
Détail d'une Etape (*round*) AES

- ***ByteSub***: Opération non linéaire (S-box) conçu pour résister à la cryptanalyse linéaire et différentielle.
- ***ShiftRow***: Permutation des bytes introduisant des décalages variables sur les lignes.
- ***MixColumn***: Chaque colonne est remplacée par des combinaisons linéaires des autres colonnes (multiplication des matrices!)
- ***AddRoundKey:*** XOR de la matrice courante avec la sous-clé correspondante à l'étape courante.



AES: SubBytes

- Les lignes (resp les colonnes) de la S-Box sont étiquetées de 0 à F en hexadécimal .
- Soit s(L,C) un élément de la S-Box de ligne L et de colonne C .
- Un byte codé 0xLC en hexadécimal sera substitué par la valeur de l'élément situé a la ligne **L** et conne **C**
- Par exemple si L = A et C = 6 alors u = 0xA6 .



AES: SubBytes

- Sub(00)=63
- Sub(12)=C9

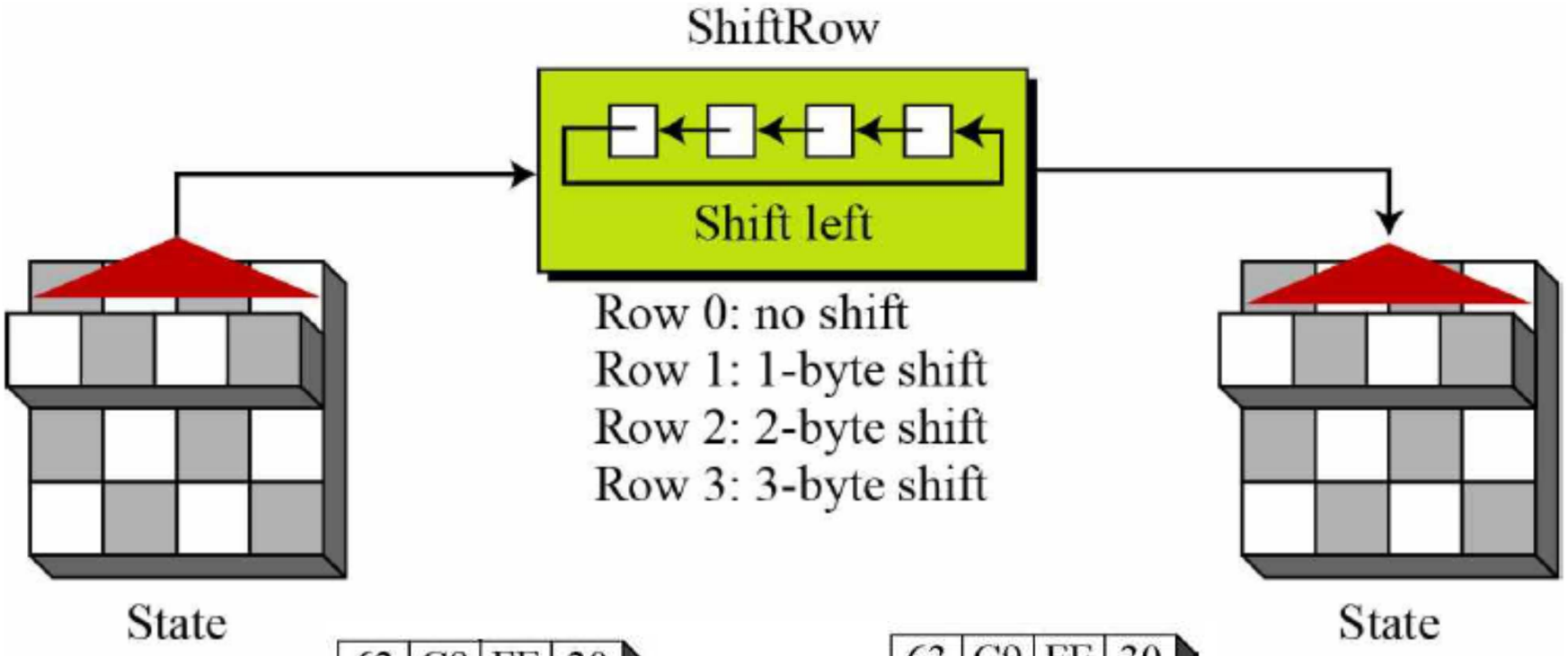
$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix}$$

$$\begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & F2 & 63 & 26 \\ C9 & C9 & 7D & D4 \\ FA & 63 & 82 & D4 \end{bmatrix}$$

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
A	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
B	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
C	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
D	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
E	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
F	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

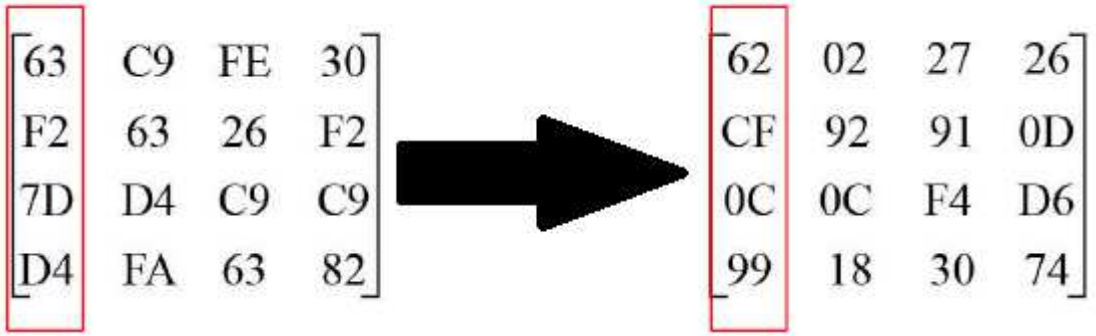
AES: ***ShiftRow***

- Consiste à décaler les lignes en rotation (diffusion).



AES: ***MixColumn***

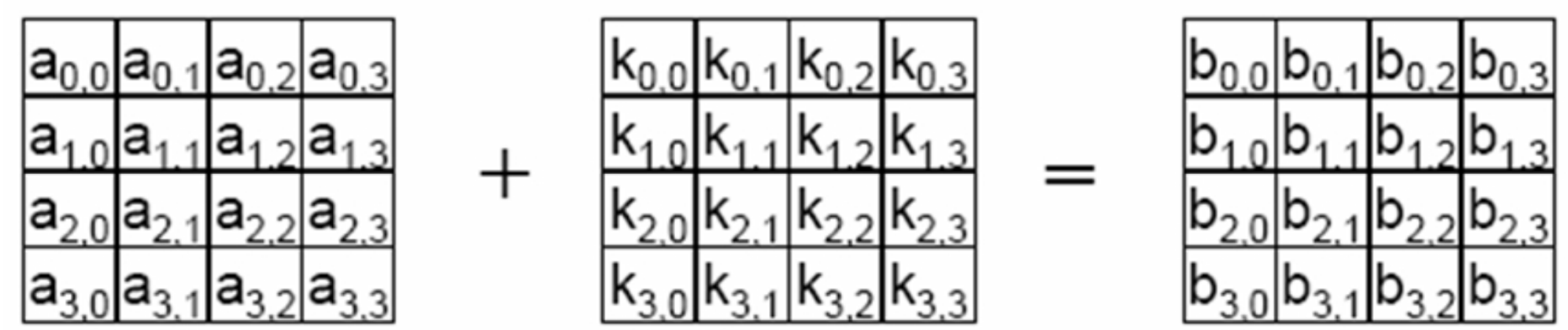
- Pour chaque colonne on applique une multiplication par une matrice, sauf le dernier tour d'AES (diffusion)



$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} 63 \\ F2 \\ 7D \\ D4 \end{bmatrix} = \begin{bmatrix} 62 \\ CF \\ 0C \\ 99 \end{bmatrix}$$

AES:AddRoundKey

- XOR de la matrice courante avec la sous-clé correspondante à l'étape courante.
- $Y_{ij}=S_{ij}$ XOR K_{ij}



AES : L'expansion de clef (KeySchedule)

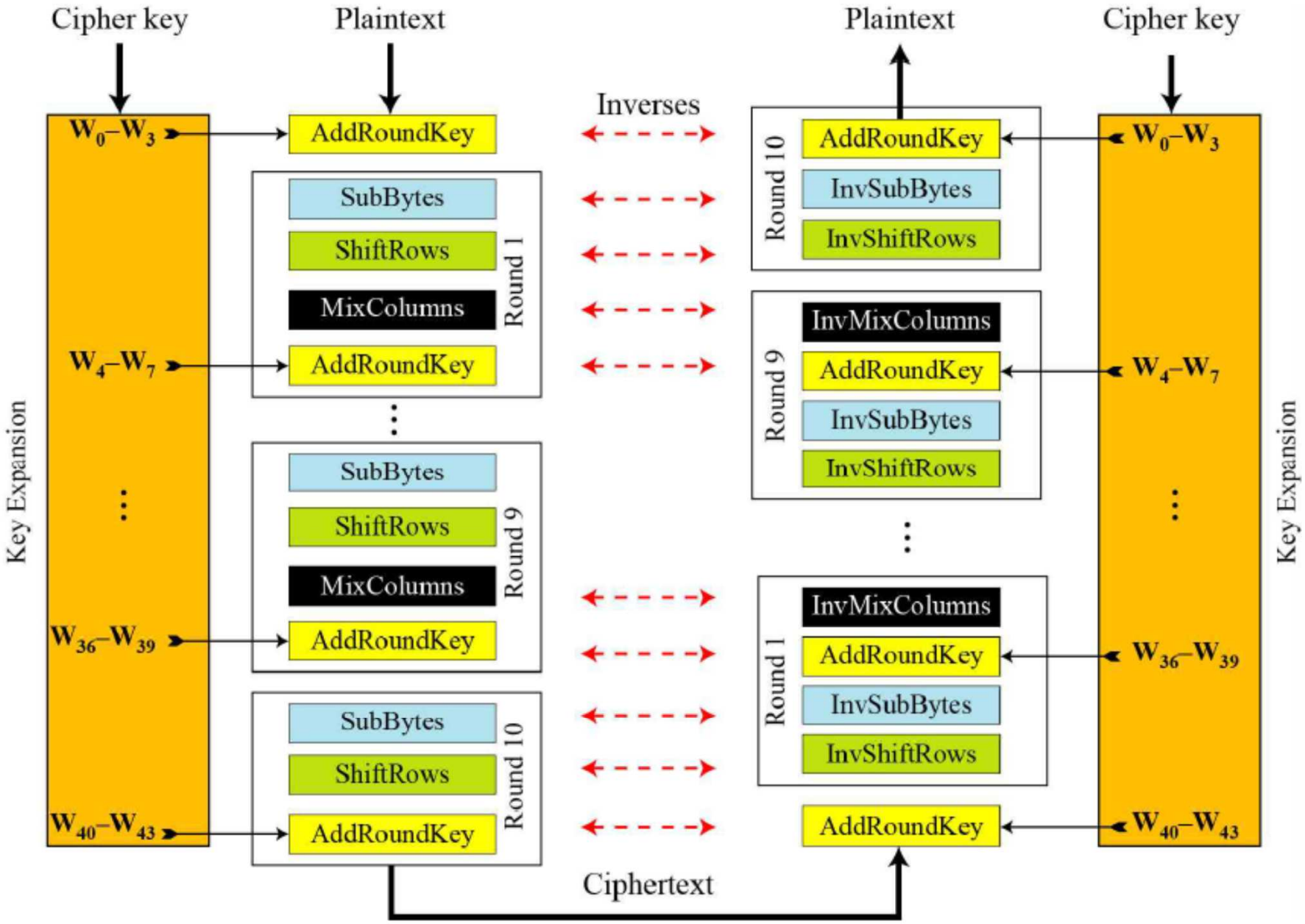
- Après avoir subi une extension (Key Expansion), la clé sera découpée en sous-clés (appelées clés de rondes),
- Definition:
- N :longeur de la clé en words (32bit): 4 pour AES-128, 6 pour AES-192 et 8 pour AES-256
- K_0, K_1, ... K_{N-1} as the 32-bit words of the original key
- R :nombre de clés nécessaires : 11 pour AES-128, 13 pour AES-192 et 15 pour AES-256
- W_0, W_1, ... W_{4R-1} : les mots de la clé développer, cas AES-128 :$K_0(W_0, W_1, W_2, W_3)$, $K_1(W_4, W_5, W_6, W_7)$ …etc.
- RotWord :fonction de décalage circulaire à gauche d'un octet: RotWord($[b_0b_1b_2b_3]$) =$[b_3b_0b_1b_2]$
- SubWord application de subBytes à chaque octets du mot.
- SubWord($[b_0b_1b_2b_3]$) =$[S(b_0)\ S(b_1)\ S(b_2)\ S(b_3)]$

AES : L'expansion de clef (KeySchedule)

- Pour i =1..4R-1

$$W_i = \begin{cases} K_i & \text{if } i < N \\ W_{i-N} \oplus \text{SubWord}(\text{RotWord}(W_{i-1})) \oplus rcon_{i/N} & \text{if } i \geq N \text{ and } i \equiv 0 \pmod{N} \\ W_{i-N} \oplus \text{SubWord}(W_{i-1}) & \text{if } i \geq N, N > 6, \text{ and } i \equiv 4 \pmod{N} \\ W_{i-N} \oplus W_{i-1} & \text{otherwise.} \end{cases}$$

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	$(\underline{\mathbf{01}}\ 00\ 00\ 00)_{16}$	6	$(\underline{\mathbf{20}}\ 00\ 00\ 00)_{16}$
2	$(\underline{\mathbf{02}}\ 00\ 00\ 00)_{16}$	7	$(\underline{\mathbf{40}}\ 00\ 00\ 00)_{16}$
3	$(\underline{\mathbf{04}}\ 00\ 00\ 00)_{16}$	8	$(\underline{\mathbf{80}}\ 00\ 00\ 00)_{16}$
4	$(\underline{\mathbf{08}}\ 00\ 00\ 00)_{16}$	9	$(\underline{\mathbf{1B}}\ 00\ 00\ 00)_{16}$
5	$(\underline{\mathbf{10}}\ 00\ 00\ 00)_{16}$	10	$(\underline{\mathbf{36}}\ 00\ 00\ 00)_{16}$


Cipher key
Plaintext
Plaintext
Cipher key
Inverses
Key Expansion
W0–W3
W4–W7
W36–W39
W40–W43
AddRoundKey
SubBytes
ShiftRows
MixColumns
Round 1
Round 9
Round 10
InvSubBytes
InvShiftRows
InvMixColumns
Ciphertext

AES: Synthèse

- La plus grande force de AES réside dans sa simplicité et dans ses performances, y compris sur des plate-formes à capacité de calcul réduite (p.ex. des cartes à puces avec des processeurs à 8 bits).
- Depuis sa publication officielle, des nombreux travaux de cryptanalyse ont été publiés avec des résultats très intéressants. En particulier, N. Courtois et P.Pieprzyk1 ont présenté une technique appelée XSL permettant de représenter AES comme un système de 8000 équations quadratiques avec 1600 inconnues binaires. L'effort nécessaire pour casser ce système est estimé (il s'agit encore d'une conjecture...) à 2^{100}.
- Ces attaques se basent sur le caractère fortement algébrique (et largement contesté...) de AES. De plus, il suffit de quelques *known plaintexts* pour les mettre en place, ce qui les distingue des attaques linéaires et différentielles.
- Les prochaines années risquent d'apporter des développements très intéressants dans la cryptanalyse de AES et d'un bon nombre de *block ciphers* itératifs.
- Même si AES reste un algorithme robuste à ce jour, il convient de surveiller de près les travaux de recherche algébrique2 qui pourraient révolutionner toutes les théories de conception actuelles relatives aux *block ciphers*.