UNIVERSITÉ LAVAL

Cryptographie symétrique

MOHAMED MEJRI

Groupe LSFM

Département d'Informatique et de Génie Logiciel

Université LAVAL

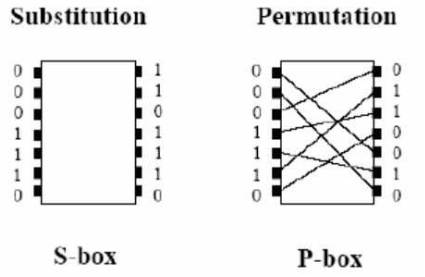
Québec, Canada

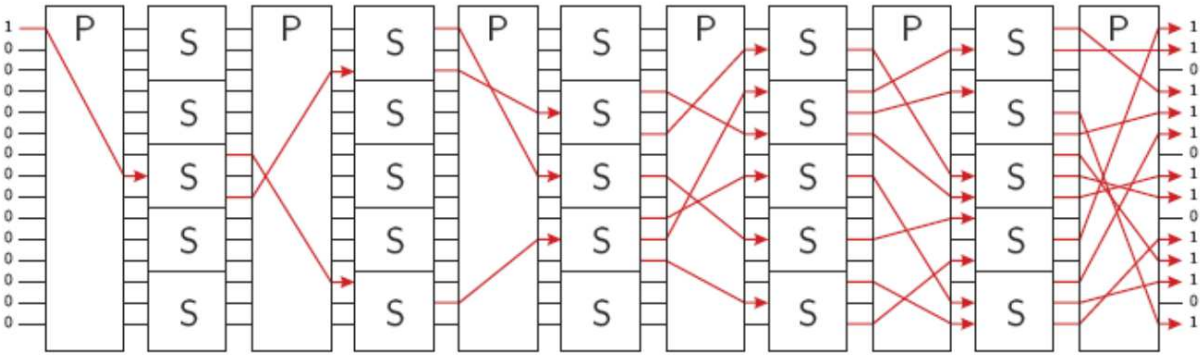
Plan

- Chiffrement par bloc
  - Chiffrement de Feistel
  - 3DES
  - AES
  - Modes de fonctionnements

Générer "l'aléatoire" : idée clé

- Sbox (Confusion) : Fortement non linaire + un petit effet d'avalanche : modifier un bit à l'entrée entraine souvent la modification d'au moins deux bits à la sortie
- Pbox (Diffusion) : Elle amplifie l'effet d'avalanche avec les itérations



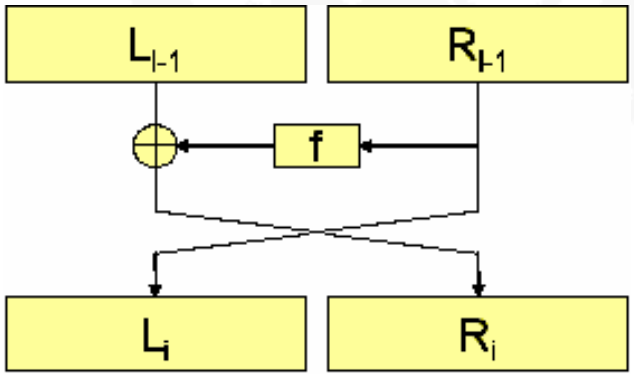


Chiffrement de Feistel

➪ Remarque : Pour construire un système cryptographique, il nous faut des fonctions réversibles et "aléatoires" (comportement imprévisible).

➪ Contribution de Feistel : Il a proposé une technique qui permet de construire une fonction réversible F à partir de n'importe quelle fonction f de la manière suivante :

$$F(L, R) \quad = \quad (R, L \oplus f(R))$$



Chiffrement de Feistel

➪ Contribution de Feistel (suite) :

– La fonction inverse de F, notée F^{-1}, est :

$$F^{-1}(L, R) = (R \oplus f(L), L)$$

– On voit bien que :

$$\begin{aligned} F^{-1}(F(L, R)) &= F^{-1}(R, L \oplus f(R)) \\ &= (L \oplus f(R) \oplus f(R), R) \\ &= (L, R) \end{aligned}$$

– Même si f n'est pas inversible, F le sera .

– Plusieurs itérations de Feistel permettent de construire des cryptosystèmes plus efficaces.

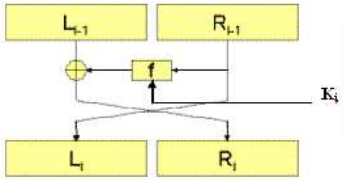
Chiffrement de Feistel

➪ Contribution de Feistel (suite) :

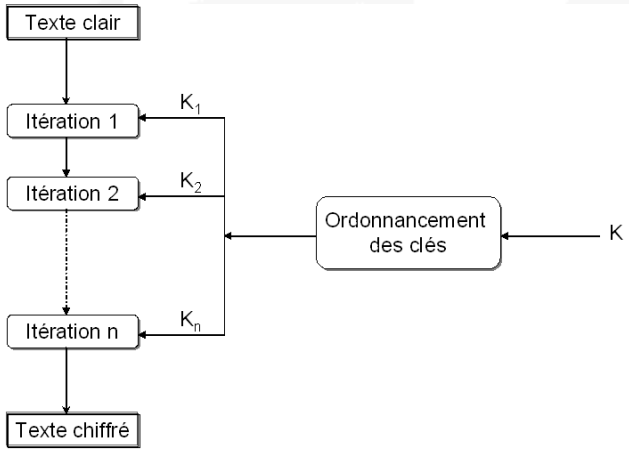
- **Théorème (Luby-Rackoff'85)** : si $f : K \times \{0,1\}^n \longrightarrow \{0,1\}^n$ est PRF sécuritaire, alors 3-itérations Feistel $F : K^3 \times \{0,1\}^{2n} \longrightarrow \{0,1\}^{2n}$ est un PRP sécuritaire à condition que les clés des itérations sont indépendantes l'une de l'autre
- $F : K \times X \longrightarrow Y$ est dite Pseudo Random Function (PRF) s'il existe un algorithme efficace permettant de la calculer.
- $E : K \times X \longrightarrow Y$ est dite Pseudo Random Permutation (PRP) si :
  - il existe un algorithme efficace et **déterministe** permettant de calculer $E(k, x)$
  - $E(k, .)$ est bijective
  - il existe un algorithme efficace pour calculer son inverse $D(k, y)$
- Une PRF (resp. PRP) est dite sécuritaire, si on ne peut pas distinguer son résultat de celui d'une autre PRF (resp. PRP) choisie aléatoirement de l'ensemble de PRF (resp. PRP) possibles

Chiffrement de Feistel

- Remarques :
  - Chaque itération peut être paramétrée par une clé.



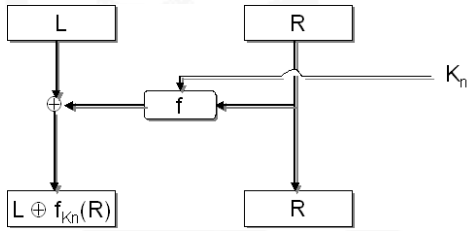
  - Question : Combien de clés nous prendra t-il ?
    - Réponse : Juste une seule à partir de laquelle on en dérive d'autres.



    - Remarque : DES dérive d'une clé de "56 bits" 16 clés de 48 bits.

Chiffrement de Feistel

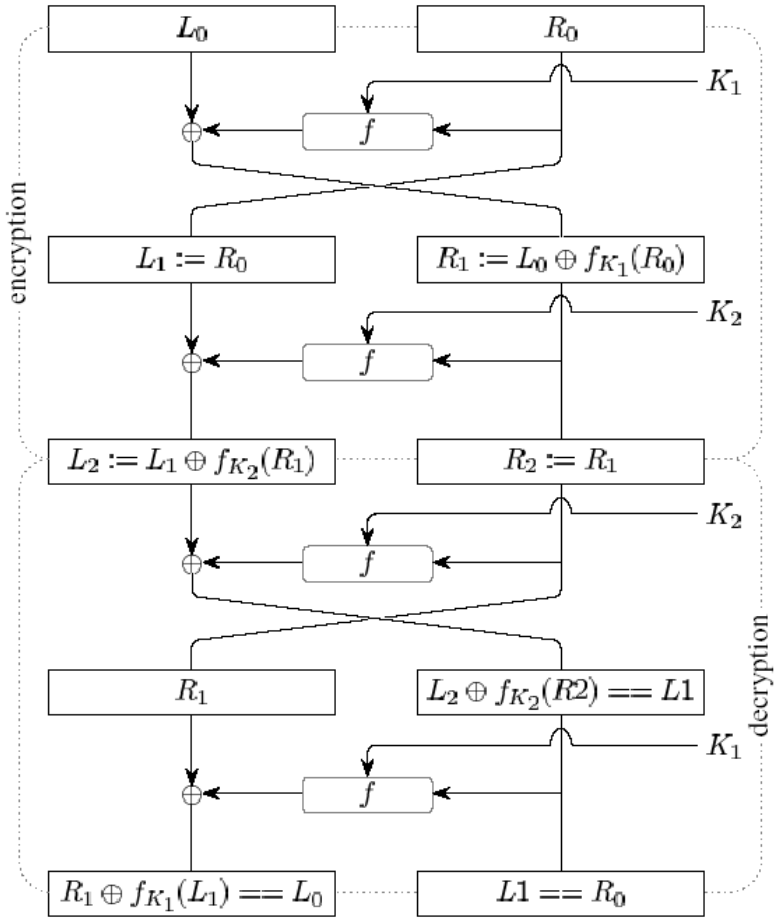
- Remarques (suite) :
  - Système cryptographique : Le schema précédent donne, modulo une petite modification, un système cryptographique élégant :
    - Modification : la fonction utilisée à la dernière itération est légèrement modifiée :



    - Conséquence : L'algorithme de décryptage sera identique à celui de cryptage sauf que les clés doivent être utilisées dans l'ordre inverse lors de décryptage.

Chiffrement de Feistel

- Exemple : $n = 2$.



Chiffrement de Feistel

- Remarques (suite) :
  - Question : Selon quels critères devrons-nous baser le choix de la fonction f ?
    - Suivre les conseils de Shannon : confusion, diffusion et non-linéarité (S'assurer que la fonction f est non linéaire : Cryptanalyser une fonction linéaire est un "jeu".)
  - Question : Combien d'itérations nous prend-il ?
    - Très difficile de trouver une réponse formelle (preuve mathématique) pour cette question.
    - Cependant on peut fixer ce paramètre en ayant comme repère les points suivants :
      - Un bit du text crypté devrait dépendre de tous les bits du texte clair et tous les bits de la clé.
      - Les textes cryptés doivent paraître "très aléatoires".

Chiffrement de Feistel

- Utilisations : Plusieurs systèmes cryptographiques symétriques se basent sur le schéma de Feistel :

	Block Size	*Key Size*	*#Rounds*
DES	64	56	16
$Double-DES$	64	112	32
$Triple-DES$	64	168	48
$IDEA$	64	128	8
$Blowfish$	64	32..448	16
$RC5$	32, 64, 128	0..2, 040	*vbl*
$CAST-128$	64	40..128	16
$RC2$	64	8..1, 024	16

Plan

- Chiffrement par bloc
  - Chiffrement de Feistel
  - 3DES
  - AES
  - Modes de fonctionnements

DES : Algorithme

⇨ DES (Data Encryption Standard) :

- **1970** : Horst Feistel a mis au point Lucifer pour IBM

  taille de bloc =128 bits et taille de la clé=128 bits.

- **1973** : NBS (National Bureau of Standards) aujourd'hui connue sous le nom NIST (National Institute of Standards and Technology) a demandé une proposition pour un système à chiffrement par blocs.

  IBM a soumis une variante de Lucifer

- **1976** : NBS adopte DES comme un standard

  taille de bloc =64 bits et taille de la clé=56 bits.

- **1997** : DES n'est plus sécuritaire contre les attaques par recherche exhaustive
- **2000** : NIST a adopté Rijndael comme AES, un remplaçant de DES

DES : Algorithme

➪ Cryptage :

Crypter(m : message, K : clé)

Début

$m_0 = IP(m)$

L_0=32 bits du poids fort de m_0

R_0=32 bits du poids faible de m_0

Pour i allant de 1 à 16 faire

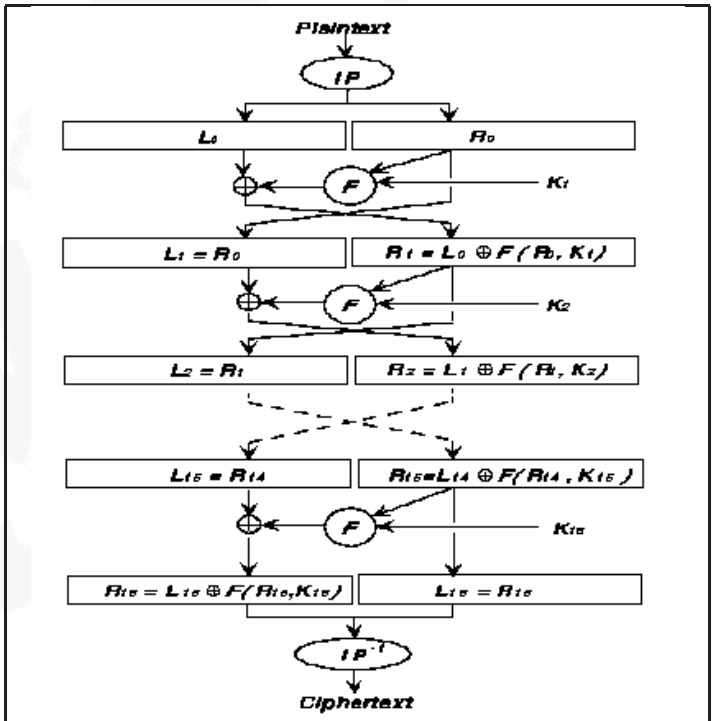
$L_i = R_{i-1}$

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

FinPour

retourner $IP^{-1}(R_{16}.L_{16})$

Fin



DES : Permutations

- Permutation initiale :

$$IP(b_1 b_2 b_3 \ldots b_{64}) = b_{58} b_{50} b_{42} \ldots b_7$$

- Permutation finale :

$$IP(b_1 b_2 b_3 b_{64}) = b_{40} b_8 b_{48} \ldots b_{25}$$

- Remarques : IP^{-1} est l'inverse de IP

$$\begin{aligned} & IP^{-1}(IP(b_1 b_2 b_3 \ldots b_{64})) \\ = \quad & IP^{-1}(b_{58} b_{50} b_{42} \ldots b_7) \\ = \quad & b_1 b_2 b_3 \ldots b_{64} \end{aligned}$$

$$\begin{aligned} & IP(IP^{-1}(b_1 b_2 b_3 \ldots b64)) \\ = \quad & IP(b_{40} b_8 b_{48} \ldots b_{25}) \\ = \quad & b_1 b_2 b_3 \ldots b_{64} \end{aligned}$$

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

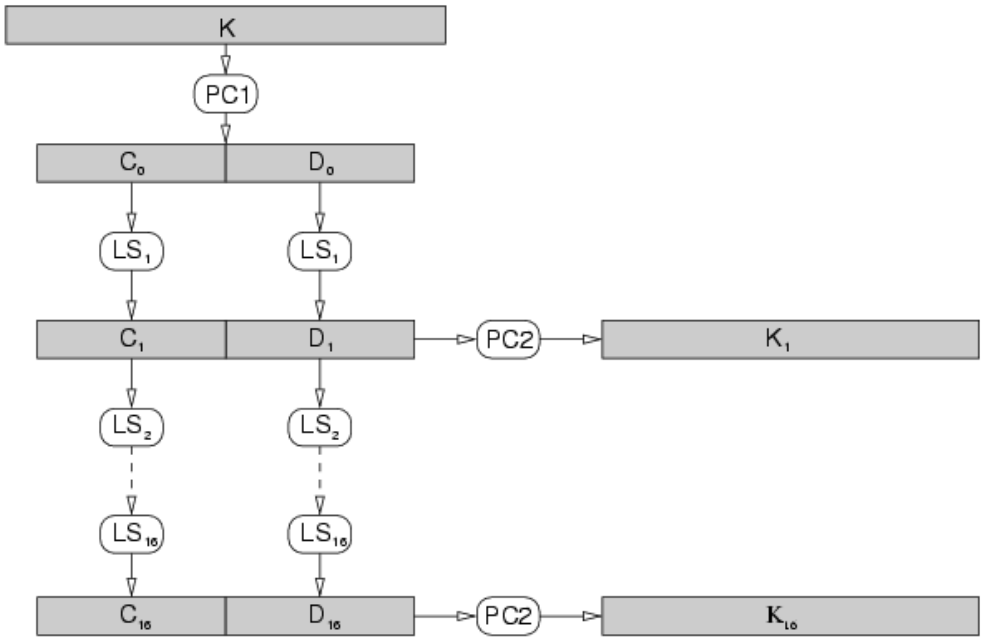
IP^{-1}

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

DES : Clés

- Initialement on a une clé K de 64 bits telle que les bits 8, 16, 24, 32, 40, 48, 56, 64 sont des bits de parité

- À partir du la clé K on va extraire 16 clés $K_1, \ldots, K_{16}$ de 48 bits de la manière suivante :



DES : Clés

- $C_0 D_0 = PC1(K = b_1 b_2 \ldots b_{64})$, avec :

$$\begin{cases} C_0 & = & b_{57} b_{49} \ldots b_{36} \text{ (moitié supérieure de } PC1) \\ D_0 & = & b_{63} b_{55} \ldots b_4 \text{ (moitié inférieure de } PC1) \end{cases}$$

- $i > 0 \Rightarrow \begin{cases} C_i & = & LS_i(C_{i-1}) \\ D_i & = & LS_i(D_{i-1}) \\ K_i & = & PC2(C_i, D_i) \end{cases}$

LS_i : Décalage à gauche de 1 bit si $i \in \{1, 2, 9, 16\}$,

de 2 bits sinon

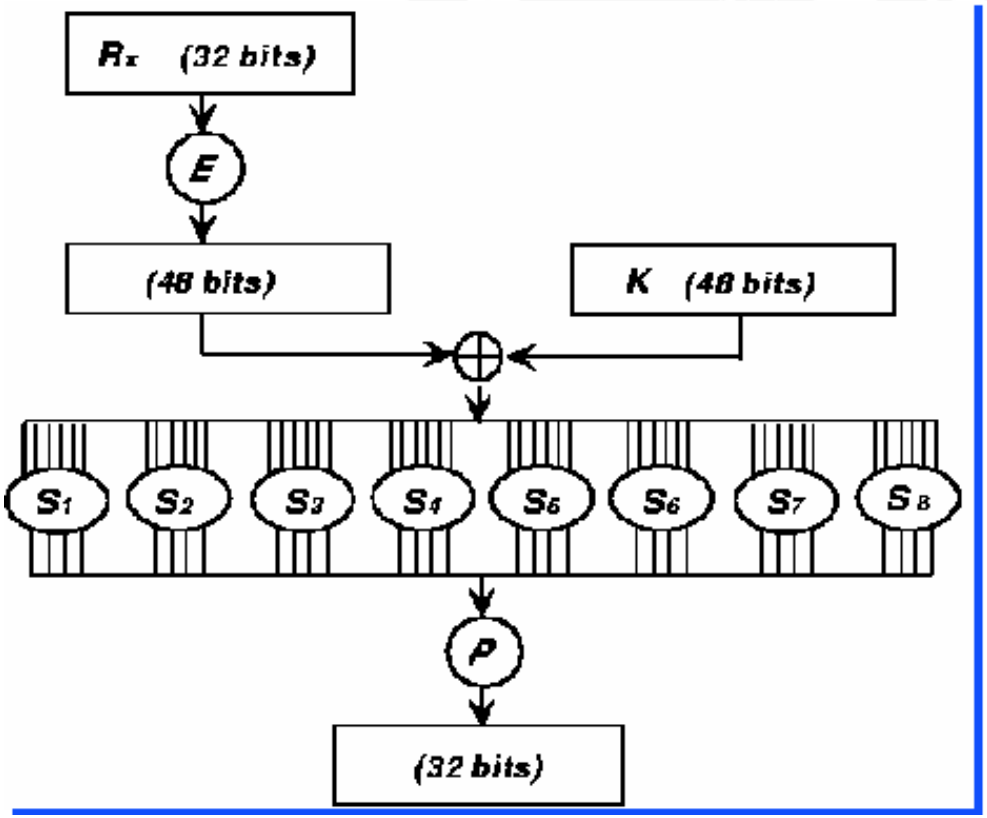
PC1

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

PC2

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 : La fonction F



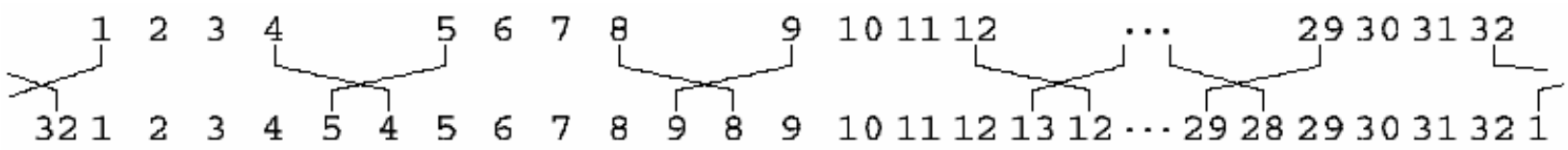
P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

DES : La fonction F

➵ La fonction E : Elle prend une suite de bits de longueur 32 bits et génère une autre de longueur 48 bits de la manière suivante :

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



➵ Remarque : Le fait qu'il y a des bits qui ont été dupliqués augmente la diffusion (modifier un bit d'entrer engendre la modification de plusieurs bits de sortie).

DES : Sbox

- Sbox Si : Elle prend 6 bits et retourne 4 bits de la manière suivante :
  - $S_i(b_1b_2b_3b_4b_5b_6)$= le contenu de la case $(b_1b_6,\ b_2b_3b_4b_5)$ du tableau S_i écrit en binaire
  - Compléter par des 0 les bits de poids forts pour avoir une représentation sur 4 bit.
  - Exemple :

$$
\begin{aligned}
S_1(001011) &= \text{contenu de la case } (01,\ 0101) \\
&= \text{contenu de la case } (1,\ 5) = 2 \\
&= 10 \text{ (binaire)}=0010
\end{aligned}
$$

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

DES : Sbox

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

DES : Sbox

- Remarques :

  - les blocs de substitution (Sbox) ne sont pas réversibles puisqu'ils prennent 6 bits et donnent en sortie 4 bits.

  - Cela ne pose pas de problème, puis que la technique de Feistel permet de générer des fonctions réversibles à partir des fonctions non-réversibles

DES : Remarque

- Effet d'avalanche : Bonne diffusion/confusion $\Rightarrow$ une petite modification au niveau des entrées donne beaucoup d'effet sur la sortie.

- Exemple :

<table>
<tr><td>Bloc-1</td><td>0^8 0^8 0^8 0^8 0^8 0^8 0^8 0^8</td></tr>
<tr><td>Bloc-2</td><td>bloc-1 avec le premier bit modifié
10^7 0^8 0^8 0^8 0^8 0^8 0^8 0^8</td></tr>
<tr><td>Clé</td><td>0^61 10^210^12 010^210^2 1^20^310
$0^21^30^2$ $0^21^20^3$ $0^21^30^2$ 01^20^210</td></tr>
<tr><td>Résultats</td><td>
<table>
<tr><th># itération</th><th># bits différents</th></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>4</td><td>39</td></tr>
<tr><td>8</td><td>29</td></tr>
<tr><td>12</td><td>30</td></tr>
<tr><td>16</td><td>34</td></tr>
</table>
</td></tr>
</table>

DES : Sécurité

- Sécurité en général :
  - Son design exceptionnel lui a permis de durer beaucoup plus longtemps que prévu au départ
  - À ce jour, aucune faiblesse structurelle majeure n'a été découverte
- Clé : La clé est de 56 bits était sécuritaire il y a 20 ans, mais susceptible aux attaques par recherche exhaustive de clés avec la technologie actuelle.
- Défi lancer par RSA (prix 10 000US$) : message = "The unknown messages is : XXXX...."
  m_1="The unkn" et $DES(k, m_1) = c_1$
  m_2="own mess" et $DES(k, m_2) = c_2$
  m_3="ages is :" et $DES(k, m_3) = c_3$
  $c_4, \ldots, c_n$ sont aussi donnés mais leurs parties claires ne sont pas connues
- Objectif trouver $k \in \{0,1\}^{56}$ telle que $DES(k, m_i) = c_i$ avec $i \in \{1, 2, 3\}$
- 1997 : recherche en utilisant le réseau Internet : **3 mois**
- 1998 : machine EFF (deep crack) : **3 jours** (250K $)
- 2003 : **4 heures**, (120 K Euro)
- 2006 : 7 jours projet COPACOBANA : 120 FPGAs (10K$)

Ne plus utiliser DES simple

DES double et triple

- DES-double : n'est pas sécuritaire (attaque MITM)
- Triple-DES (3DES) : Une variante de DES standardisé par le NIST.
  - Obtenue par utilisation de DES trois fois $E_{K_3}(D_{K_2}(E_{K_1}(m)))$
  - L'idée d'utiliser D est de pouvoir utiliser un circuit 3DES comme un simple DES lorsque $K_1 = K_2$ ou $K_2 = K_3$.
  - Nettement plus sécuritaire que DES, mais la lenteur est un problème, surtout en comparaison avec les systèmes plus récents.
  - DES n'est pas un groupe (groupe $\Rightarrow \forall\ k,\ k' \in\ \mathcal{K},\ \exists\ k'' \in\ \mathcal{K} \mid E_{k"}(m) = E_{k'}(E_k(m))$) autrement le Triple-DES n'aurait servi à rien.
  - l'attaque MITM permet de réduire l'espace de recherche à 2^{118}, mais cela reste sécuritaire.

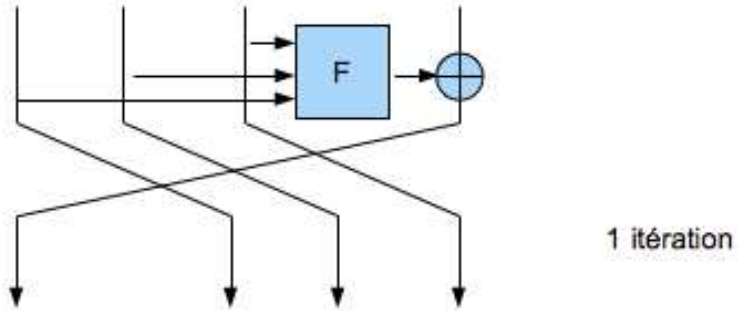
DESX

- Variante de DES : permet d'utiliser DES d'une manière plus sécuritaire contre la recherche exhaustive
- Standard ? n'est pas standardisé par le NIS
- Définition : $E : K \times M \longrightarrow M$

$$EX((k_1, k_2, k_3), m) = k_1 \oplus E(k_2, m \oplus k_3)$$

  - taille de la clé = $|k_1| + |k_2| + |k_3|$ =64+56+64= 184 bits
  - la meilleure attaque connue nécessite un temps de l'ordre de 2^{120}
  - remarque : $k_1 \oplus E(k_2, m)$ et $E(k_2, m \oplus k_3)$ n'améliorent pas DES

Variante de Feistel



Structure inversible pour toute fonction F

Application RC6 (Rivest et Robshaw) 1998, dérivé de RC5

Bloc 128 bits, taille de clé variable (supporte 128, 192 et 256, etc.), itérations variable (par défaut 20)

Autre que Feistel

- Camellia : introduit par Mitsubishi et NTT (Japan), approuvé par ISO/IEC, European Union's NESSIE et Japanese CRYPTREC.
  - comme AES, taille de bloc 128, clé=128, 192 ou 196
  - conçu pour être rapide sur matériels et logiciels

Autre que Feistel

➪ Camellia (suite) : FL et FL^{-1} : combinaison de XOR, AND et OR pour créer des irrégularités entre les étapes (+sécurité !)

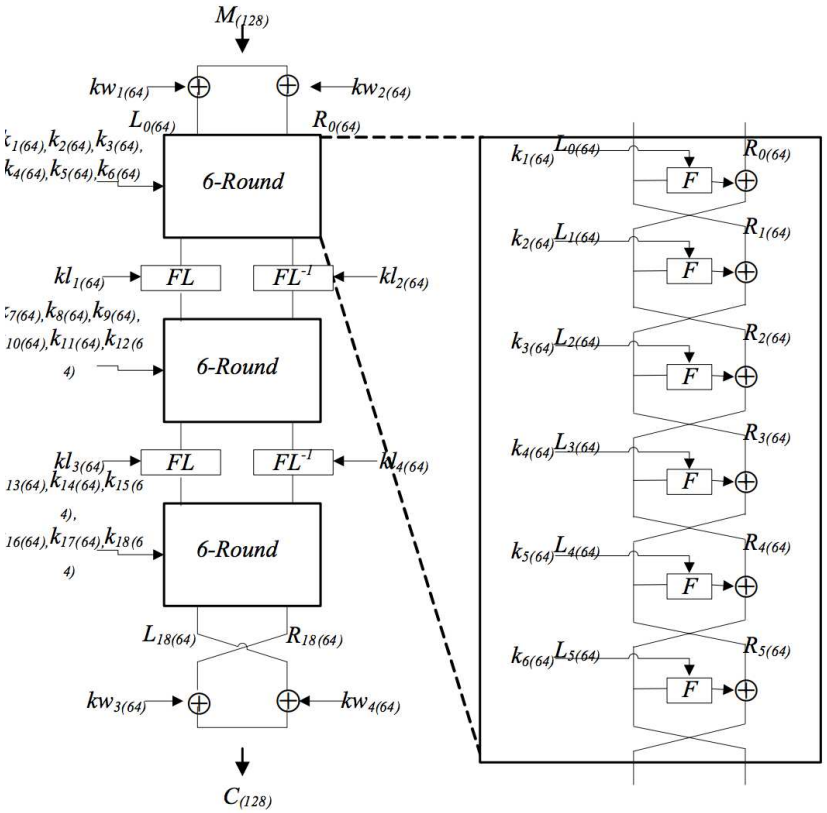
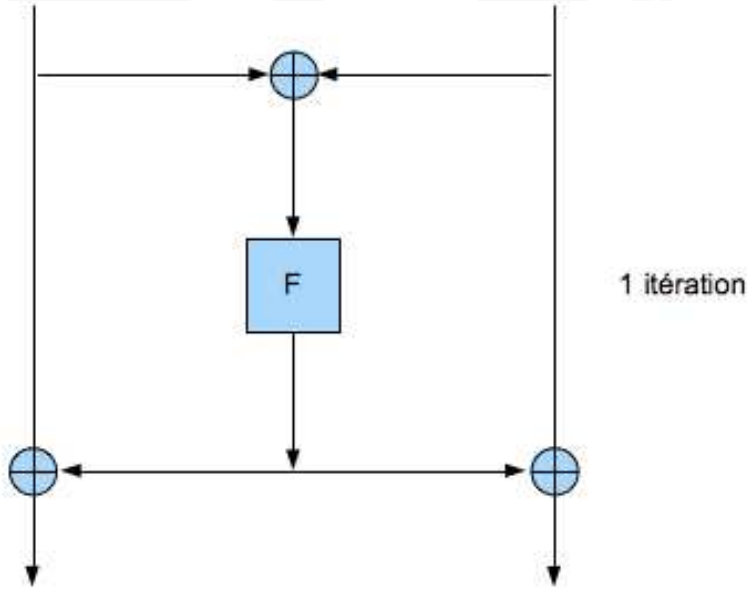


Fig.1 Encryption process of Camellia

Source : article X. ZHAO , T. WANG , Y. ZHENG Cache Timing Attacks on Camellia Block Cipher

Autre que Feistel

➪ Lai-Massey



Structure inversible pour toute fonction F

Autre que Feistel

Application

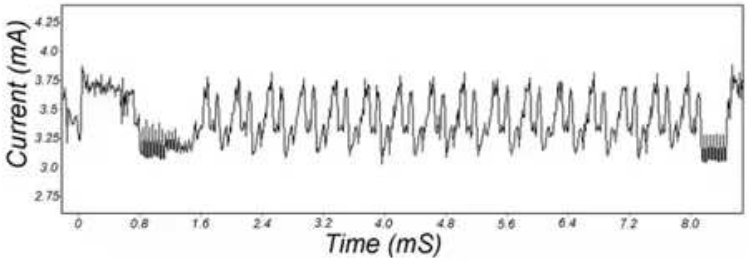
⇨ IDEA I.D.E.A. (International Data Encryption Algorithm)

- Lai et Massey en 1992
- Blocs de 64 bits, clé 128 bits, 8 itérations
- Performance comparable à DES
- Problème de brevet (le produit était disponible pour utilisation non commerciale)
- Dernier brevet expiré en 2012

Autres attaques sur les chiffrements par blocs

➪ Side Channel attacks : mesurer le temps de l'encryption/decryption ou l'énergie consommée peut révéler la clé.

smartcard



[Kocher, Jaffe, Jun, 1998]

l'image précédente montre le temps des 16 itérations DES avec les deux permutations pour une clé donnée. Chercher des corrélations entre les clés et l'énergie consommée

➪ Fault attacks : forcer votre "smart card" à faire des erreurs (surchauffer le processeur, etc.). Le processeur va mal fonctionner et retourner des donnés erronées qui peuvent révéler des informations sur la clé

On vous recommande de ne pas utiliser votre propre système cryptographique si vous n'avez pas assez de compétence. N'essayez même pas d'implanter par vous même des systèmes existants

Plan

- Chiffrement par bloc
  - Chiffrement de Feistel
  - 3DES
  - AES
  - Modes de fonctionnements

Advanced Encryption Standard (AES)

La faiblesse de DES (sécurité faible pour une clé de 56 bits, vitesse inadéquate pour 3DES) devient de plus en plus inquiétante et il faut lui trouver un successeur qui sera nommé AES.

- 1997 Le NIST fait un appel d'offre pour AES (algorithme symétrique). Les exigences les plus importantes du cahier des charges sont les suivantes :
  - Grande sécurité : sécurité "prouvable" (forte résistance aux différentes attaques connues).
  - Large portabilité : peut être facilement implanté sur des machines 32 bits, des cartes "processeur 8 bits", etc.)
  - Excellente Rapidité : aussi rapide que DES simple (soft + hard)
  - Forte flexibilité : des blocs de longueurs variables (128 bits, etc) ; des clés de tailles variables 128 bits, 192 bits 256 bits.
  - Bonne Simplicité : Facile à comprendre et à analyser
- 1998 15 candidats se présentent (5 US, 2 Canada, 1 Austria, Belgium, Costa Rica, France, Germany, Japan, Korea, UK/Israel/Norway). 1ère conférence AES commence : les propositions sont analysées par toute la communauté.
- 1999 Mai : 2ème AES Conférence. Août : 5 finalistes (MARS (IBM), RC6 (RSA Laboratories), Twofish (Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Niels Ferguson, Chris Hall), Serpent (Ross Anderson, Eli Biham, Lars Knudsen), Rijndael (Joan Daemen, Vincent Rijmen)). Les analyses se poursuivent.
- 2000 Avril : 3ème AES conférence. Octobre : NIST déclare Rijndael comme vainqueur .

Advanced Encryption Standard (AES)

- Adopté comme standard en Novembre 2001
- Conçu par Johan Daemen et Vincent Rijmen (d'ou son nom original Rijndael)

Vincent Rijmen et Joan Daemen

- Il s'agit également d'un block cipher itératif (comme DES) mais pas d'un Feistel Cipher.
- Blocs Plaintext/Ciphertext de taille variable : 128, 192, 256 bits (AES garde seulement 128 comme taille du bloc)
- Clé de longueur variable : 128, 192, ou 256 bits.
- La taille de la clé est indépendante de la taille des blocs.

Advanced Encryption Standard (AES)

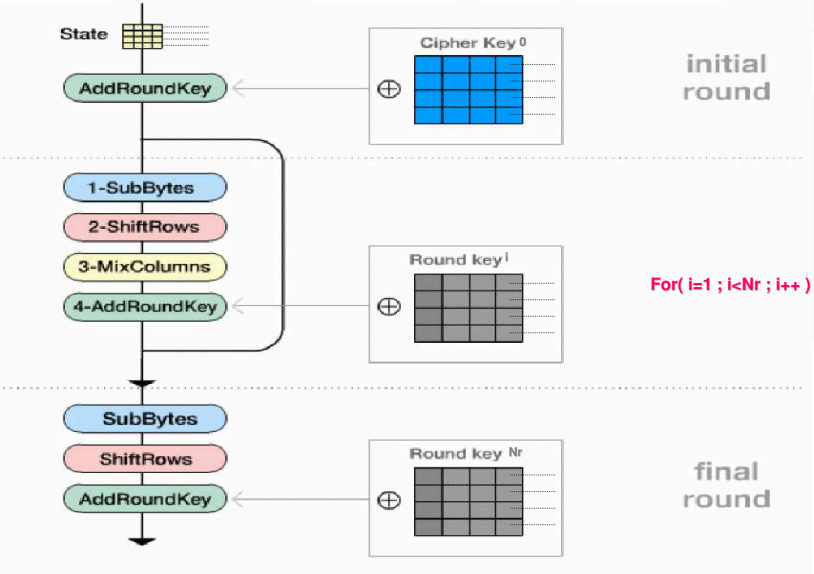
- Techniques semblables a DES (substitutions, permutations, XOR) basées sur des operations algébriques simples, élégantes et très performantes.
- Toutes les operations s'effectuent dans le corps GF(2^8) : le corps fini de polynômes de degré 7 ($b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x_2 + b_1x^1 + b_0$) avec des coefficients dans GF(2). En particulier, un octet pour AES est un élément dans GF(2^8) et les opérations sur les octets (additions, multiplications,...) sont définies comme sur GF(2^8).
- 2 fois plus performant (en software) et 1022 fois (en théorie...) plus sur que DES. (software 52MByte/sec sur 800MHz Pentium III ; hardware 650MByte/sec sur ASIC)
- Évolutif : La taille de la clé peut être augmentée si nécessaire.

Advanced Encryption Standard (AES)

- Initial Round (1 opération). AddRoundKey.
- N Rounds (4 opérations par itération) : SubBytes, ShiftRows, MixColumns, AddRoundKey. N dépend de la taille de la clé et la taille des blocs. Appliquer le principe de Shannon : (Permutation + Substitution)^N : diffusion + confusion.

Nr	Nb=4	Nb=6	Nb=8
Nk=4	10	12	14
Nk=6	12	12	14
Nk=8	14	14	14

- Final Round (3 opérations) : SubBytes, MixColumns, AddRoundKey.



- **Remarque :** plus la taille de la clé est grande, plus le système est sécuritaire, mais il devinet plus lent

Advanced Encryption Standard (AES)

Entrées : un état e et une clé k

Sortie : e chiffré

$K \leftarrow KeySchedule(k)$;

$e \leftarrow e \oplus k_0$

Pour $i = 1$ à $N_r - 1$ faire

$\quad e \leftarrow Tour(e, k_i)$

$e \leftarrow TourFinal(e, k_{N_r})$

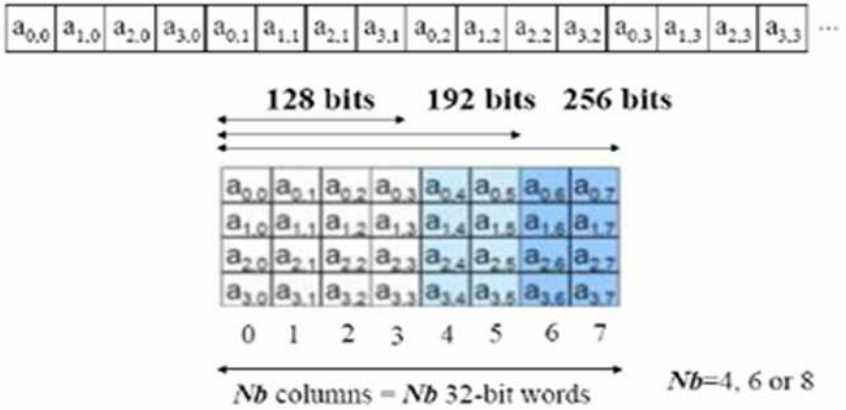
$Tour(e, k_i)$	$TourFinal(e, k_i)$
$e \leftarrow SubByte(e)$	$e \leftarrow SubByte(e)$
$e \leftarrow ShiftRow(e)$	$e \leftarrow ShiftRow(e)$
$e \leftarrow MixColumns(e)$	$e \leftarrow e \oplus k_i$
$e \leftarrow e \oplus k_i$	

- Intel Westmere (aussi ADM Bulldozer) a introduit des instructions machines AES
- **aesenc** (*Tour*) et **aesenclast** (*TourFinal*)
- Deux registres de 128 bits **xmm1** (état), **xmm2** (clé)
- **aesenc xmm1**, **xmm2** (fait un tour et retourne le résultat dans **xmm1**)
- **asekeygenassist** permet le $KeySchedule$
- avec ces instructions on peut avoir du code 14 fois plus rapide

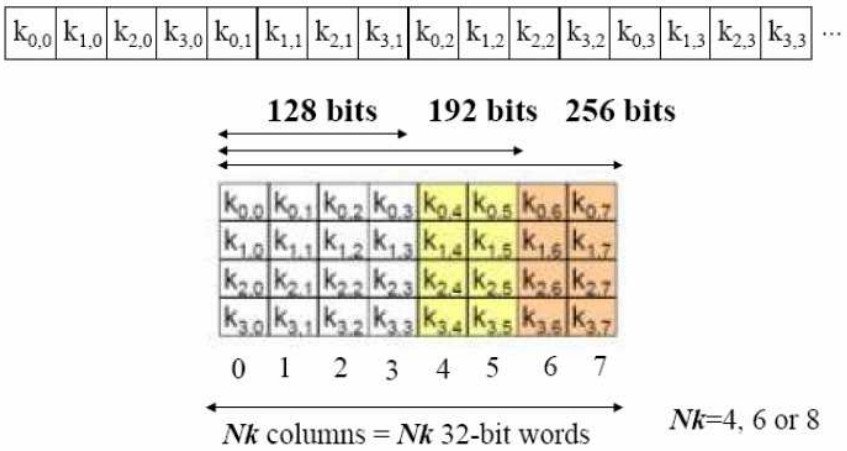
Advanced Encryption Standard (AES)

- La clé et le message sont présentés sous forme de matrices (chaque case est un octet).

Message



Clé



Advanced Encryption Standard (AES)

- Initial Round (1 opération). AddRoundKey. Toute manipulation qui précède l'intervention de la clé est vaine de point de vu sécurité.

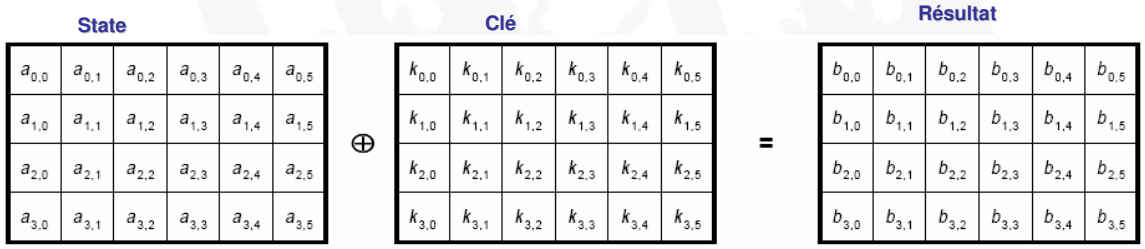


Figure 5: In the key addition the Round Key is bitwise EXORed to the State.

$b_{ij}=a_{ij} \oplus k_{ij}$

Exemple

32	88	31	E0
43	5A	31	37
F6	30	98	07
A8	8D	A2	34

⊕

2B	28	AB	09
7E	AE	F7	CF
15	D2	15	4F
16	A6	88	3C

=

19	A0	9A	E9
3D	F4	C6	F8
E3	E2	8D	48
BE	2B	2A	08

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) :
  - SubBytes : C'est une substitution (S-Box) non linéaire (pour resister aux attaques différentielles, linéaires, etc.) effectuée sur la matrice State issue de l'opération précédente.
    - Définition dans $GF(2^8)$: $SubBytes(S) = S'$ tel que $S'[i,j] = M * (S[i,j]^{-1}) + V$
      - Supposons que $S[i,j] = \alpha_7\alpha_6\alpha_5\alpha_4\alpha_3\alpha_2\alpha_1\alpha_0$
      - Alors $S[i,j]^{-1} = \beta_7\beta_6\beta_5\beta_4\beta_3\beta_2\beta_1\beta_0$ est l'inverse de $S[i,j]$ dans $GF(2^8)$ avec $P(x) = x^8 + x^4 + x^3 + x + 1$. À noter que l'inverse n'est pas une opération linéaire
      - Autrement dit : $S[i,j](x) * S[i,j]^{-1}(x) \equiv 1 \ mod \ (x^8 + x^4 + x^3 + x + 1)$
      - Par convention $(00^{-1} = 00)$
      - Avec $S[i,j](x) = \alpha_7.x^7 + \alpha_6.x^6 + \alpha_5.x^5 + \alpha_4.x^4 + \alpha_3.x^3 + \alpha_2.x^2 + \alpha_1.x + \alpha_0$
      - Et $S[i,j]^{-1}(x) = \beta_7.x^7 + \beta_6.x^6 + \beta_5.x^5 + \beta_4.x^4 + \beta_3.x^3 + \beta_2.x^2 + \beta_1.x + \beta_0$
      - Finalement $S'[i,j] = s'_7, s'_6, s'_5, s'_4, s'_3, s'_2, s'_1, s'_0$

$$\underbrace{\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \\ s'_4 \\ s'_5 \\ s'_6 \\ s'_7 \end{bmatrix}}_{S'[i,j]} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}}_{M} \bullet \underbrace{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \end{bmatrix}}_{S[i,j]^{-1}} + \underbrace{\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}}_{V}$$

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) :
  - SubBytes : (suite)
    - En pratique : Le calcul de $SubBytes(S)$ peut être simplifié en utilisant des (SBox) (à la DES) :

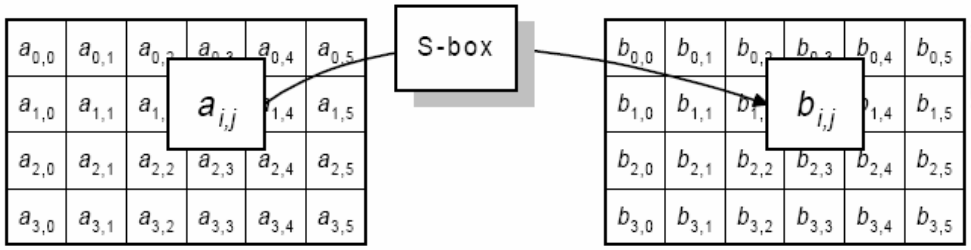


Figure 2: ByteSub acts on the individual bytes of the State.

Source=http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) :
  - SubBytes : (suite)
    - En pratique : Le calcul de $SubBytes(S)$ peut être simplifié en utilisant la table (SBox) suivante (à la DES) :

hex	y															
	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

x

S-Box

    - $SubBytes(S) = S'$ tel que $S'[i,j] = SBox[i_1, j_1]$ avec i_1 est le premier caractère hexadecimal de $S[i,j]$ et j_1 est le deuxième caractère hexadecimal de $S[i,j]$.

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) :
  - SubBytes : (suite)

hex	y: 0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x: 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

S-Box

19	A0	9A	E9
3D	F4	C6	F8
E3	E2	8D	48
BE	2B	2A	08

D4	E0	B8	1E
27	BF	B4	41
11	98	5D	52
AE	F1	E5	30

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) : SubBytes, ShiftRows, MixColumns, AddRoundKey. AES n'utilise pas le réseaux de Feistel. Dans une seule itération, on travaille le bloc au complet (au lieu de travailler juste sa moitié comme pour DES). C'est pour cela AES nécessite moins d'itérations que DES.
  - ShiftRows : Un simple décalage de lignes de la matrice State issue de l'opération précédente.

Nb=Taille d'un bloc d'entrée divisée par 32=Nombre de colonnes dans la matrice State

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

Table 2: Shift offsets for different block lengths.

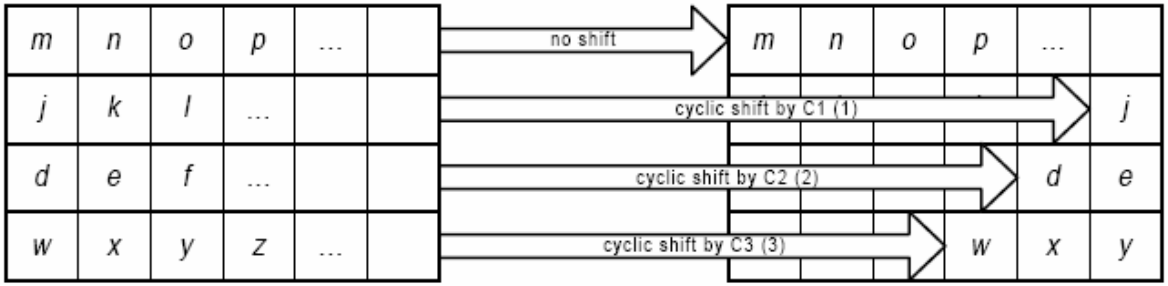


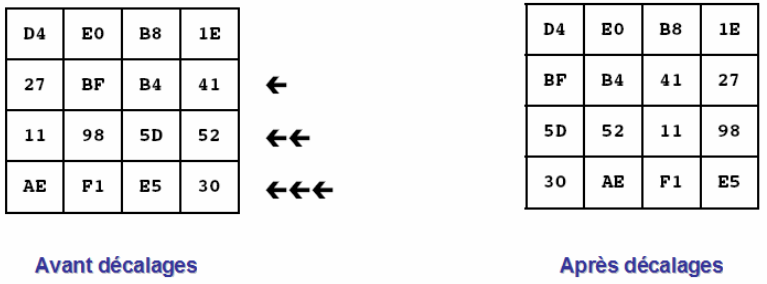
Figure 3: ShiftRow operates on the rows of the State.

Source=http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) :
  - ShiftRows :

    Exemple avec Nb=4
    - La premiere ligne n'est pas décalée.
    - La deuxième ligne est décalée de 1 byte vers la gauche.
    - La troisième ligne est décalée de 2 bytes vers la gauche.
    - La quatrième ligne est décalée de 3 bytes vers la gauche.



Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) : MixColumns : C'est une multiplication de deux matrices (une matrice constante et la matrice "State" issue de l'opération précédente) dans le corp de Galois GF(2^8). L'objectif est la diffusion : modifier un élément d'une colonne de la matrice, impliquera une modification sur tous les éléments de la colonne correspondante de la matrice résultat.

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

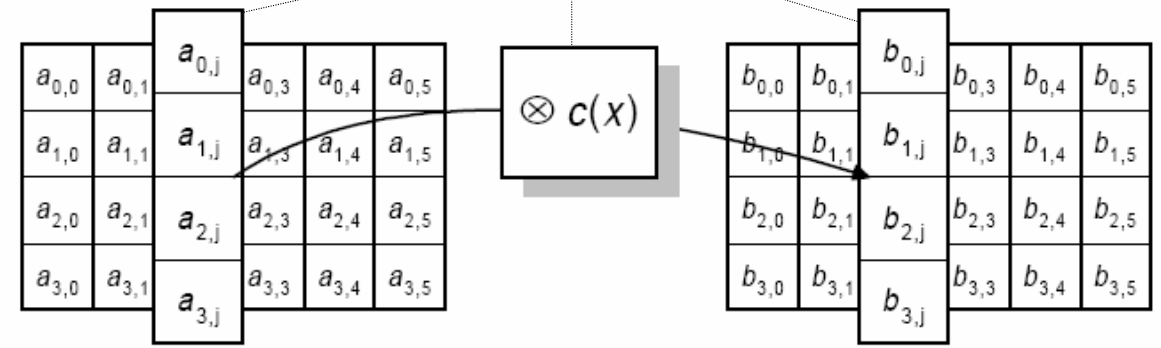


Figure 4: MixColumn operates on the columns of the State.

Source=http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) : MixColumns : Augmenter la diffusion !

Exemple :

$$\text{Matrice Canstante} = \begin{array}{|c|c|c|c|}\hline 02 & 03 & 01 & 01 \\ \hline 01 & 02 & 03 & 01 \\ \hline 01 & 01 & 02 & 03 \\ \hline 03 & 01 & 01 & 02 \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|c|}\hline 02 & 03 & 01 & 01 \\ \hline 01 & 02 & 03 & 01 \\ \hline 01 & 01 & 02 & 03 \\ \hline 03 & 01 & 01 & 02 \\ \hline \end{array} \times \begin{array}{|c|c|c|c|}\hline D4 & E0 & B8 & 1E \\ \hline BF & B4 & 41 & 27 \\ \hline 5D & 52 & 11 & 98 \\ \hline 30 & AE & F1 & E5 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline ? & ? & ? & ? \\ \hline ? & ? & ? & ? \\ \hline ? & ? & ? & ? \\ \hline ? & ? & ? & ? \\ \hline \end{array}$$

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) : MixColumns : Augmenter la diffusion !

Exemple (suite) :

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

Canstante

$\bullet$

$D4$	$E0$	$B8$	$1E$
BF	$B4$	41	27
$5D$	52	11	98
30	AE	$F1$	$E5$

State

$=$

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

Result

$Result[0,0] = 02 \bullet D4 \oplus 03 \bullet BF \oplus 01 \bullet 5D \oplus 01 \bullet 30$

Multiplication dans le corps de Galois :

$$02 \bullet D4 = a(x) \bullet b(x) \; mod \; p(x)$$

$$02 = 00000010 \Longrightarrow a(x) = 0x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 1x^1 + 0x^0 = x$$

$$D4 = 11010100 \Longrightarrow b(x) = 1x^7 + 1x^6 + 0x^5 + 1x^4 + 0x^3 + 1x^2 + 0x^1 + 0x^0 = x^7 + x^6 + x^4 + x^2$$

$p(x) = x^8 + x^4 + x^3 + x^1 + 1$ C'est un polynôme irréductible dans $GF(2^8)$

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) : MixColumns : Augmenter la diffusion !

Exemple (suite) :

$$\begin{aligned} a(x) \bullet b(x) \ mod \ p(x) &= x \bullet (x^7 + x^6 + x^4 + x^2) \ mod \ (x^8 + x^4 + x^3 + x^1 + 1) \\ &= x^8 + x^7 + x^5 + x^3 \ mod \ (x^8 + x^4 + x^3 + x^1 + 1) \\ &= (x^8 + x^4 + x^3 + x^1 + 1) \\ &\quad \oplus (x^8 + x^7 + x^5 + x^3) \ mod \ (x^8 + x^4 + x^3 + x^1 + 1) \\ &= (x^7 + x^5 + x^4 + x^1 + 1) \ mod \ (x^8 + x^4 + x^3 + x^1 + 1) \\ &= 1011 \ 0011 = B3 \end{aligned}$$

Multiplication en calcul binaire :

- Multiplication par 01 (0000 001 en binaire) : Le nombre reste inchangé !
- Multiplication par 02 (0000 0010 en binaire) : Enlever le bit à gauche et ajouter un zéro à droite. Si le bit enlevé est un 0 alors le résultat obtenu et le final sinon on lui fait un xor avec 0001 1011

$$\begin{aligned} D4 &= 1101 \ 0100 \\ 02 \bullet D4 &= 1010 \ 1000 \oplus 0001 \ 1011 = 1011 \ 0011 = B3 \end{aligned}$$

- Multiplication par 03 (0000 0011 en binaire) : $03 \bullet N = 02 \bullet N \oplus N$

$$\begin{aligned} BF &= 1011 \ 1111 \\ 02 \bullet BF &= 0111 \ 1110 \oplus 0001 \ 1011 = 0110 \ 0101 \\ 03 \bullet BF &= 02 \bullet BF \oplus BF = 0110 \ 0101 \oplus 1011 \ 1111 = 1101 \ 1010 \end{aligned}$$

Advanced Encryption Standard (AES)

- N Rounds (4 opérations par itération) : MixColumns : Augmenter la diffusion !

Exemple (suite) :

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

Canstante

$\bullet$

$D4$	$E0$	$B8$	$1E$
BF	$B4$	41	27
$5D$	52	11	98
30	AE	$F1$	$E5$

State

$=$

04	$E0$	48	28
66	CB	$F8$	06
81	19	$D3$	26
$E5$	$9A$	$7A$	$4C$

Result

$$
\begin{aligned}
Result[0,0] &= 02 \bullet D4 \oplus 03 \bullet BF \oplus 01 \bullet 5D \oplus 01 \bullet 30 \\
&= 10110011 \oplus 11011010 \oplus 01011101 \oplus 00110000 = 0000\ 0100 \\
&= 04_h \\
Result[0,1] &= 02 \bullet E0 \oplus 03 \bullet B4 \oplus 01 \bullet 52 \oplus 01 \bullet AE = E0 \\
&\vdots
\end{aligned}
$$

Advanced Encryption Standard (AES)

- Extension de la clé (Key Expansion) : Il s'agit d'étendre le tableau de la clé initiale K à un autre tableau plus grand W qui contiendra les clés de toutes les itérations.
  - Dans ce qui suit, si T est un tableau alors on note pat $T[i]$ la colonne i de T. Le remplissage des colonnes de W se fait de la manière suivante :

$$W[i] = \begin{cases} K[i] & si \quad 0 \leq i < Nk \\ W[i-Nk] \oplus SubByte(RotByte(W[i-1])) \oplus Rcon[i/N_k] & si \quad (i \geq Nk) \text{ et } (i\%Nk = 0) \\ W[i-Nk] \oplus SubByte(W[i-1]) & si \quad (i \geq Nk > 6) \text{ et } (i\%Nk = 4) \\ W[i-Nk] \oplus W[i-1] & sinon \end{cases}$$

  - $RotByte$: Rotation des octets (bytes) d'un vecteur vers le haut, i.e :

$$RotByte(\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}) = \begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix}$$

  - $Rcon$: C'est une matrice constance calculée à partir de la formule suivante :

$$Rcon[i] = \begin{bmatrix} (02)^{i-1} \\ 00 \\ 00 \\ 00 \end{bmatrix}$$

avec $(02)^{i-1}$ est la représentation du polynôme $\underbrace{02 \bullet \ldots \bullet 02}_{i-1}$ dans $GF(2^8)$ $((02)^0 : 01, (02)^1 : 02, (02)^2 : 04, (02)^3 : 08, (02)^4 : 10, (02)^5 : 20, (02)^6 : 40, (02)^7 : 80, (02)^8 : 1B, (02)^9 : 36, \ldots)$.

Advanced Encryption Standard (AES)

- Extension de la clé (Key Expansion) :
  - Exemple :

$W[0]$	$W[1]$	$W[2]$	$W[3]$	$W[4]$
$2B$	28	AB	09	?
$7E$	AE	$F7$	CF	?
15	$D2$	15	$4F$	?
16	$A6$	88	$3C$	?

Clé initiale

$$W[4] = W[0] \oplus SubByte(RotByte(W[3])) \oplus Rcon[1]$$

$$\begin{bmatrix} A0 \\ FA \\ FE \\ 17 \end{bmatrix} = \begin{bmatrix} 2B \\ 7E \\ 15 \\ 16 \end{bmatrix} \oplus \begin{bmatrix} 8A \\ 84 \\ EB \\ 01 \end{bmatrix} \oplus \begin{bmatrix} 01 \\ 00 \\ 00 \\ 00 \end{bmatrix}$$

Advanced Encryption Standard (AES)

- Extension de la clé (Key Expansion) :
  - Exemple :

$W[0]$	$W[1]$	$W[2]$	$W[3]$	$W[4]$	$W[5]$	...
$2B$	28	AB	09	$A0$	?	...
$7E$	AE	$F7$	CF	FA	?	...
15	$D2$	15	$4F$	FE	?	...
16	$A6$	88	$3C$	17	?	...

$$\begin{matrix} W[5] \\ \begin{bmatrix} 88 \\ 54 \\ 2C \\ B1 \end{bmatrix} \end{matrix} = \begin{matrix} W[1] \\ \begin{bmatrix} 28 \\ AE \\ D2 \\ A6 \end{bmatrix} \end{matrix} \oplus \begin{matrix} W[4] \\ \begin{bmatrix} A0 \\ FA \\ FE \\ 17 \end{bmatrix} \end{matrix}$$

Advanced Encryption Standard (AES)

- Clé de l'itération i :

Round key `i` is given by the Round Key buffer words `W[Nb*i]` to `W[Nb*(i+1)]`. This is illustrated in Figure 6.

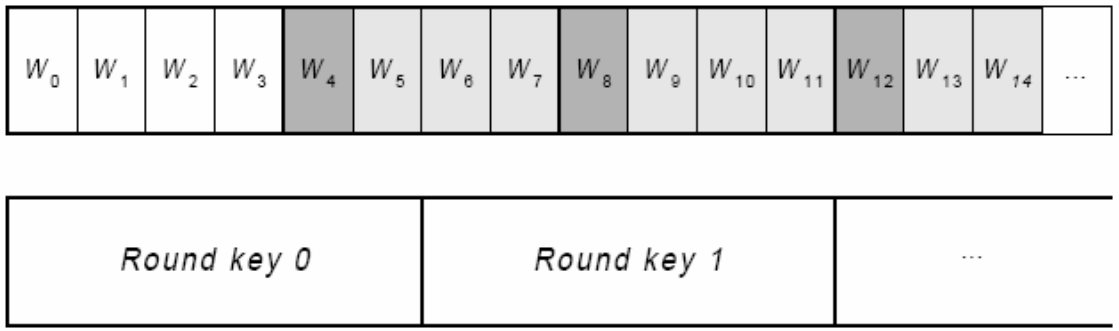


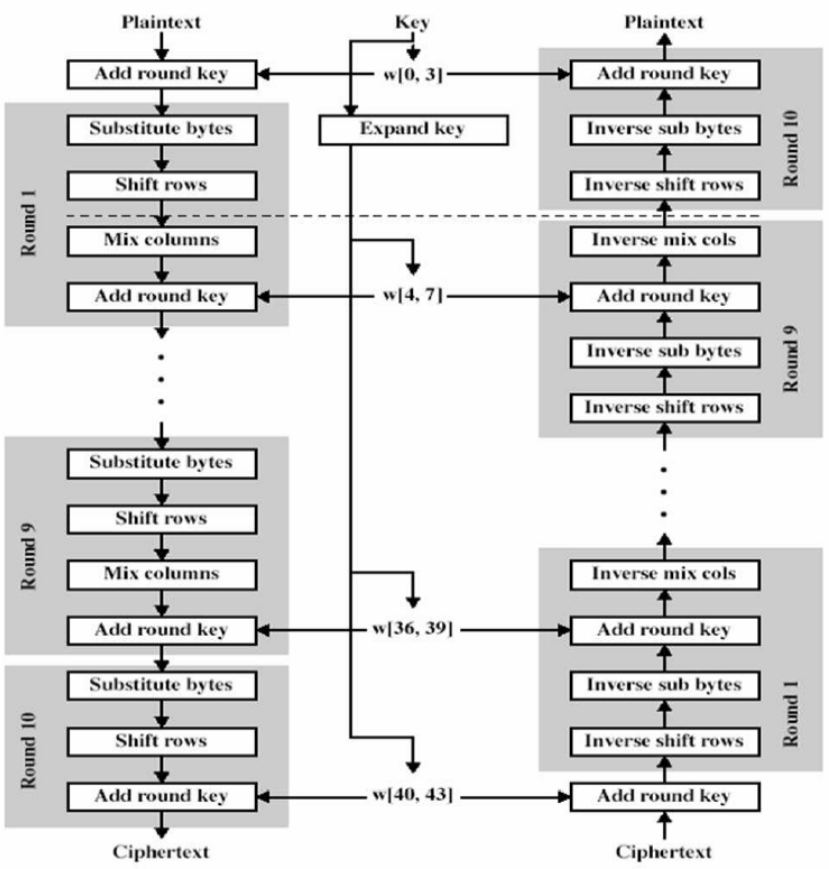
Figure 6: Key expansion and Round Key selection for `Nb` = 6 and `Nk` = 4.

Note: The key schedule can be implemented without explicit use of the array `W[Nb*(Nr+1)]`. For implementations where RAM is scarce, the Round Keys can be computed on-the-fly using a buffer of **`Nk`** words with almost no computational overhead.

Source=http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf

Advanced Encryption Standard (AES)

- Decryption : Chaque opération est inversible.



Advanced Encryption Standard (AES)

➻ Attaques :

- Meilleur attaque contre la recherche de la clé : 4 fois plus rapide que la recherche exhaustive (exemple 2^{126} au lieu de 2^{128})

  une clé de AES-128 bits n'offre que la sécurité d'une clé 126 bits d'un système "parfait"

- Attaque particulière sur AES-256 : Si on a 2^{99} paires (m_i, c_i) générées à partir de 4 clés ayant un certain lien entre elles (ayant une distance de Hamming très faible), on peut retrouver la clé dans un temps de l'ordre de 2^{99}

  pas très pratique car les clés sont supposées être générées aléatoirement

Plan

- Chiffrement par bloc
  - Chiffrement de Feistel
  - 3DES
  - AES
  - Modes de fonctionnements

Modes de fonctionnement

⇨ Problème :

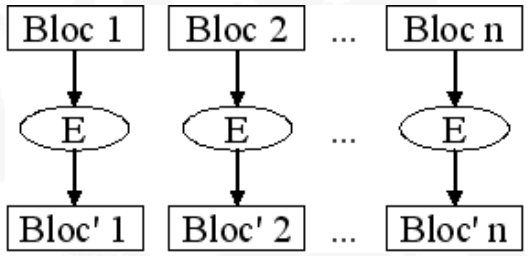
- La plupart de cryptosystèmes modernes permettent de crypter des blocs de taille fixe (n bits).
- DES (n=64 bits), AES (128bits ,etc.)
- *Quoi faire si la taille de message est supérieure à n ?*

⇨ Solution :

- On découpe le message en segments de taille n.
- On crypte les segments en utilisant le chaînage.

Modes de fonctionnement

- ECB (Electronic codebook) :
  - Principes :
    - On découpe $m = m_1 \dots m_n$
    - On calcule $c_i = E_K(m_i)$
    - On envoie $c = c_1 \dots c_n$



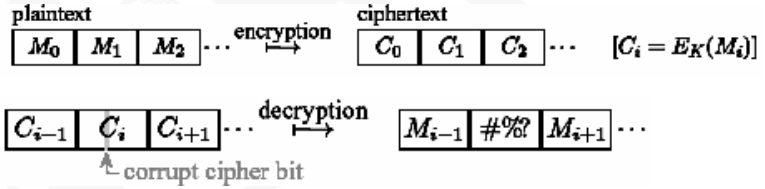
  - Caractéristiques :
    - Erreurs :

      - L'effet d'une erreur est local.

      Bloc c_i érroné $\Rightarrow$ seul bloc m_i affecté

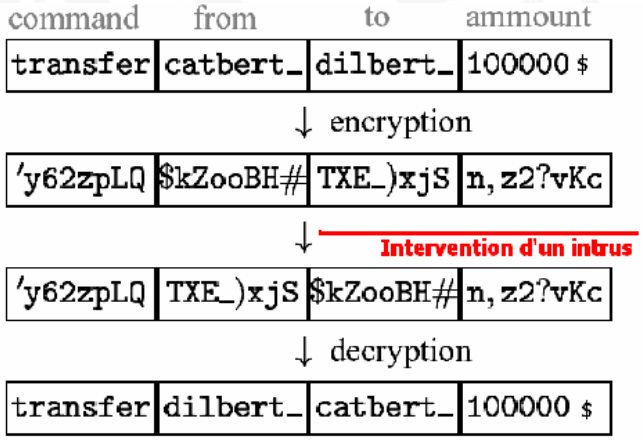
      - La perte du bloc c_i au complet n'affecte que le bloc m_i.

      - La perte ou l'ajout d'un bit dans le bloc c_i affecte le bloc $m_i,\ m_{i+1} \dots$



Modes de fonctionnement

- Caractéristiques (suite) :
  - Sécurité :
    - Pas de camouflage des "patterns" : (monoalphabétique).
    - Les permutations et les substitutions des blocs sont possibles.



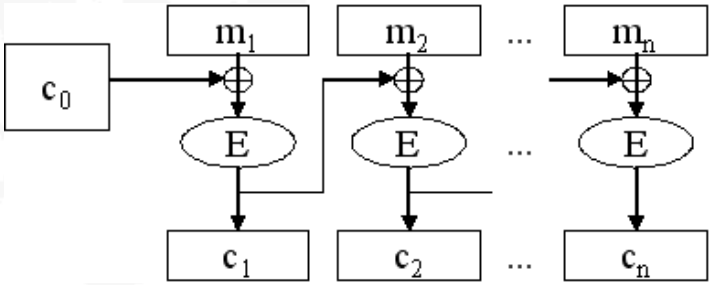
    - Possibilité de construction d'un dictionnaire ("code book attack").
  - Efficacité :
    - Processing en parallèle possible. Pas de préprocessing.
    - Besoin de padding
    - Accès direct à un enregistrement chiffré

Modes de fonctionnement

- CBC (Cipher Block Chaining) :
  - Principes :
    - On découpe $m = m_1 \dots m_n$
    - On choisit un premier bloc c_0 aléatoire
    - On calcule $c_i = E_K(m_i \oplus c_{i-1})$
    - On envoie $c_0, c_1 \dots c_n$
    - On décrypte par $m_i = c_{i-1} \oplus D_K(c_i)$

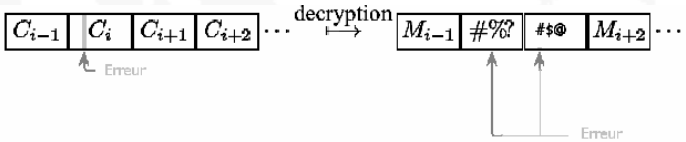
      En effet :

$$\begin{aligned} & c_{i-1} \oplus D_K(c_i) \\ = \; & c_{i-1} \oplus D_K(E_K(m_i \oplus c_{i-1})) \\ = \; & c_{i-1} \oplus m_i \oplus c_{i-1} \\ = \; & m_i \end{aligned}$$



Modes de fonctionnement

- CBC (Cipher Block Chaining) :
  - Caractéristiques :
    - Erreurs : Une erreur au niveau de bloc C_i aura un effet sur M_i et M_{i+1} seulement ($M_i = C_{i-1} \oplus D_K(C_i)$). La synchronisation, suite à une perte ou un ajout de bits, est difficile.

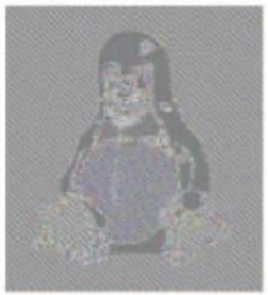


    - Sécurité :
      - Camouflage des "patterns" + construction d'un dictionnaire est difficile
      - Les permutations et les substitutions des blocs sont difficiles.
      - L'élimination du bloc initial ou final est possible.
      - Modification possible du premier bloc : en remplaçant m_1 par $m_1 + \Delta$ et c_0 par $c_0 + \Delta$
  - Efficacité :
    - Pas de parallélisation pour l'encryption (possible pour la décryption). Pas de préprocessing.
    - Besoin de padding
    - Accès direct à un enregistrement chiffré ? Besoin de C_i et C_{i-1} pour trouver M_i

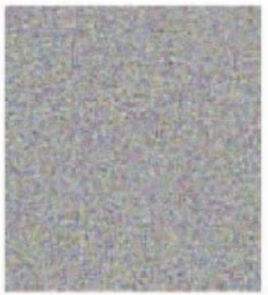
Modes de fonctionnement

➪ CBC vs ECB :

Original



ECB



CBC

Modes de fonctionnement

CFB (Cipher FeedBack) :

Principe :

$$I_1 = IV$$

$$I_{j+1} = 2^r * I_j + c_j \ mod \ 2^n$$

$$O_j = E_k(I_j)$$

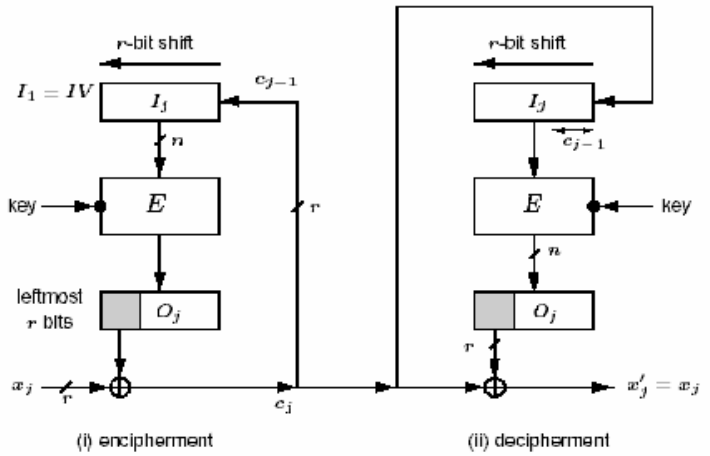
$$L(O_i, r) = r \text{ bits de gauche de } O_i$$

Encryption

$$c_i = L(O_i, r) \oplus m_i$$

Décryption

$$m_i = L(O_i, r) \oplus c_i$$



Modes de fonctionnement

- CFB (Cipher FeedBack) :
  - Caractéristiques :
  - Erreurs : Une erreur dans un bloc affecte plusieurs autres blocs. La synchronisation est automatique après un certain nombre de blocs.
  - Sécurité :
    - Camouflage des "patterns" : (polyalphabétique)
    - Les permutations et les substitutions des blocs sont difficiles.
    - L'élimination de blocs (début ou fin) est possible.
    - La construction d'un dictionnaire est difficile.
  - Efficacité :
    - Pas de parallélisation pour l'encryption.
    - Pas de préprocessing possible.
    - Pas besoin de padding
    - Accès direct à un enregistrement chiffré ? Parfois, besoin de $C_i, C_{i-1} \ldots C_{i-\lceil n/r \rceil}$ pour trouver M_i !

Modes de fonctionnement

- OFB (Output FeedBack) :
  - Principe :

$$O_0 = IV$$

$$O_i = E_k(O_{i-1}) \quad i > 0$$

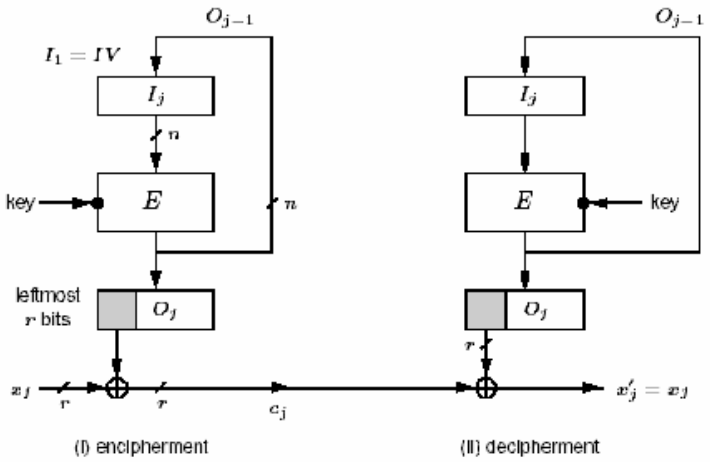
$$L(O_i, r) = r \text{ bits de gauche de } O_i$$

  - Encryption

$$c_i = L(O_i, r) \oplus m_i$$

  - Décryption

$$m_i = L(O_i, r) \oplus c_i$$



Modes de fonctionnement

- OFB (Output FeedBack) :
  - Caractéristiques :
  - Erreurs : Une erreur dans un bloc n'affecte pas les autres. La synchronisation est difficile.
  - Sécurité :
    - Camouflage des "patterns" : (polyalphabétique)
    - Les permutations et les substitutions des blocs sont difficiles.
    - L'élimination de blocs (fin) est possible.
    - La construction d'un dictionnaire est difficile.
  - Efficacité :
    - Processing en parallel est possible (après préparation des O_i) .
    - Préprocessing est possible (préparation des O_i).
    - Pas besoin de padding
    - Accès direct à un enregistrement chiffré ? Pour accéder à M_i, besoin de C_i et de chifrrer IV i fois !

Modes de fonctionnement

- CTR (Counter) : un nouveau mode très demandé (utilisé dans IPsec)
  - Principe : Similaire à OFB sauf qu'on chiffre le contenu d'un compteur au lieu de faire un "Feedback".

$$Ctr_0 = IV \text{ (compteur)}$$

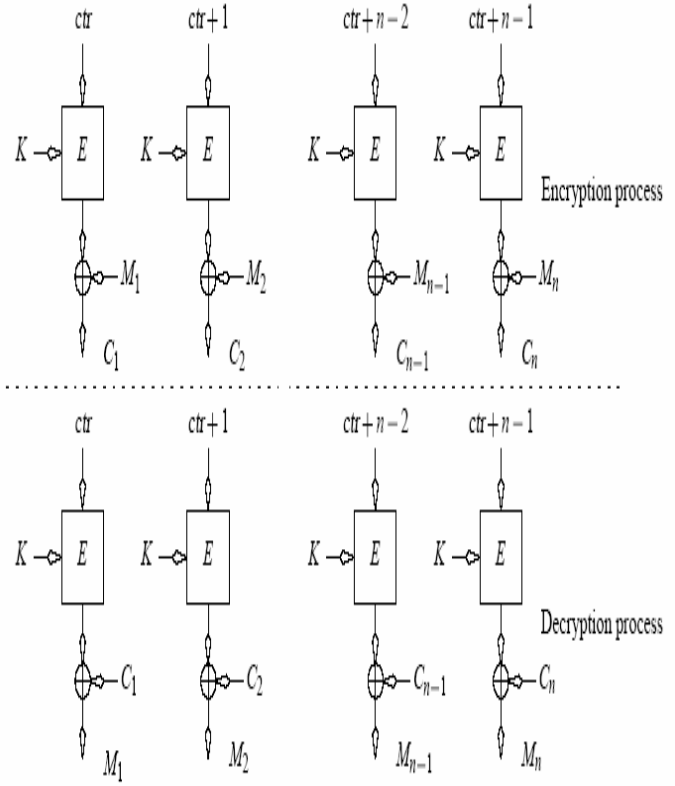
$$Ctr_i = Ctr_{i-1} + 1 \quad i > 1$$

- Encryption

$$c_i = E_k(Ctr_{i-1}) \oplus m_i$$

- Décryption

$$m_i = E_k(Ctr_{i-1}) \oplus c_i$$



Modes de fonctionnement

- CTR (Counter) :
  - Caractéristiques :
  - Erreurs : Une erreur dans un bloc n'affecte pas les autres. La synchronisation est difficile.
  - Sécurité : Très bonne sécurité comparativement aux autres modes.
  - Efficacité :
    - Processing en parallel est possible.
    - Préprocessing est possible (préparation des $E_k(Ctr_{i-1})$).
    - Taille du message n'augmente pas (pas besoin de faire un "padding").
    - Accès rapide aux blocs d'un fichier (ou une base de données) encrypté.
  - Remarque : On doit s'assurer que la valeur d'un compteur n'est jamais utilisée plus qu'une fois avec la même clé.

Modes de fonctionnement

- Bourrage (padding)) :
  - PKCS#7 (RFC 5652) :
    - Peut traiter des blocs de taille maximale 256 octets
    - Utilisé dans OpenSSL et autres
    - Les octets manquants du dernier bloc sont complétés par des octets qui indiquent le nombre d'octets manquants.
      - Si la taille d'un bloc est de 64 bits (8 octets) et le dernier bloc contient deux octets O_1 et $O2$, il sera compété comme suit :

O_1	O_2	06	06	06	06	06	06

      - Si la taille d'un bloc est de 64 bits (8 octets) et le dernier bloc est complet on est obligé d'ajouter le bloc

08	08	08	08	08	08	08	08

      - Si la taille d'un bloc est de 128 bits (16 octets) et le dernier bloc contient deux octets O_1 $O2$, il sera compété comme suit :

O_1	O_2	$0E$	$0E$	$0E$	$0E$	$0E$	$0E$	$0E$	$0E$	$0E$	$0E$	$0E$	$0E$	$0E$	$0E$

Modes de fonctionnement

- Bourrage (padding)) :
  - ANSI X.923 :
    - Seul le dernier octet contient le nombre d'octets ajoutés, les restes contiennent des zéros.
    - Si la taille d'un bloc est de 64 bits (8 octets) et le dernier bloc contient deux octets O_1 et $O2$, il sera compété comme suit :

O_1	O_2	00	00	00	00	00	06

  - ISO 10126 :
    - Le dernier octet contient le nombre d'octets ajoutés, les restes sont générés aléatoirement.
    - Si la taille d'un bloc est de 64 bits (8 octets) et le dernier bloc contient deux octets O_1 et $O2$, il sera compété comme suit :

O_1	O_2	$0A$	$E1$	11	29	70	06

  - ISO/IEC 7816-4 :
    - Le premier octet de bourrage ajouté doit ête 08, ceux qui suivent doivent être des 00
    - Si la taille d'un bloc est de 64 bits (8 octets) et le dernier bloc contient deux octets O_1 et $O2$, il sera compété comme suit :

O_1	O_2	08	00	00	00	00	00

  - Zero padding Ajouter des zéros partout dans les octets qui maquent

O_1	O_2	00	00	00	00	00	00