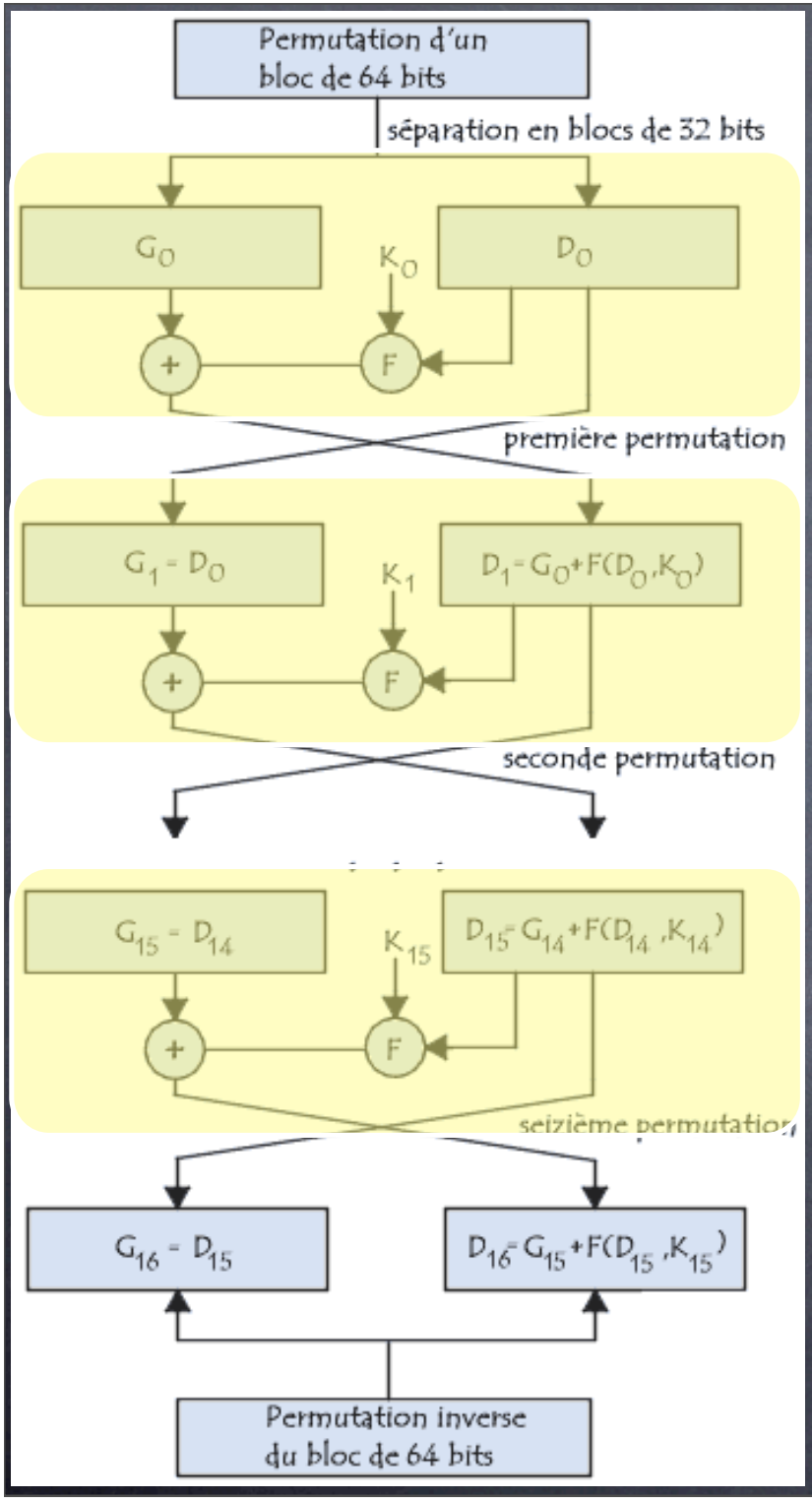
Data Encryption Standard (DES)

- DES est un standard NIST (des années 1970) pour le chiffrement à clé secrète de blocs de données de taille fixe. Sa version la plus simple (et la moins sûre) :
  - Chiffre des blocs de données de 64 bits.
  - La clé secrète est longue de 56 bits (en fait 64 bits dont 8 bits servent à tester la parité de chaque bloc de 8 bits de clé).



Bloc de données à chiffrer :

Ronde 1

Ronde 2

Ronde 16

1	2	3	4	5	6	...	61	62	63	64

Permutation initiale

58	50	42	34	26	18	...	31	23	15	7

DES dérive 16 clés de 48bits, une pour chaque ronde, à partir d'une seule clé de 64(56) bits.

G_0 D_0 K_0

$G_1 = D_0$

$D_1 = F(D_0, K_0) + G_0$

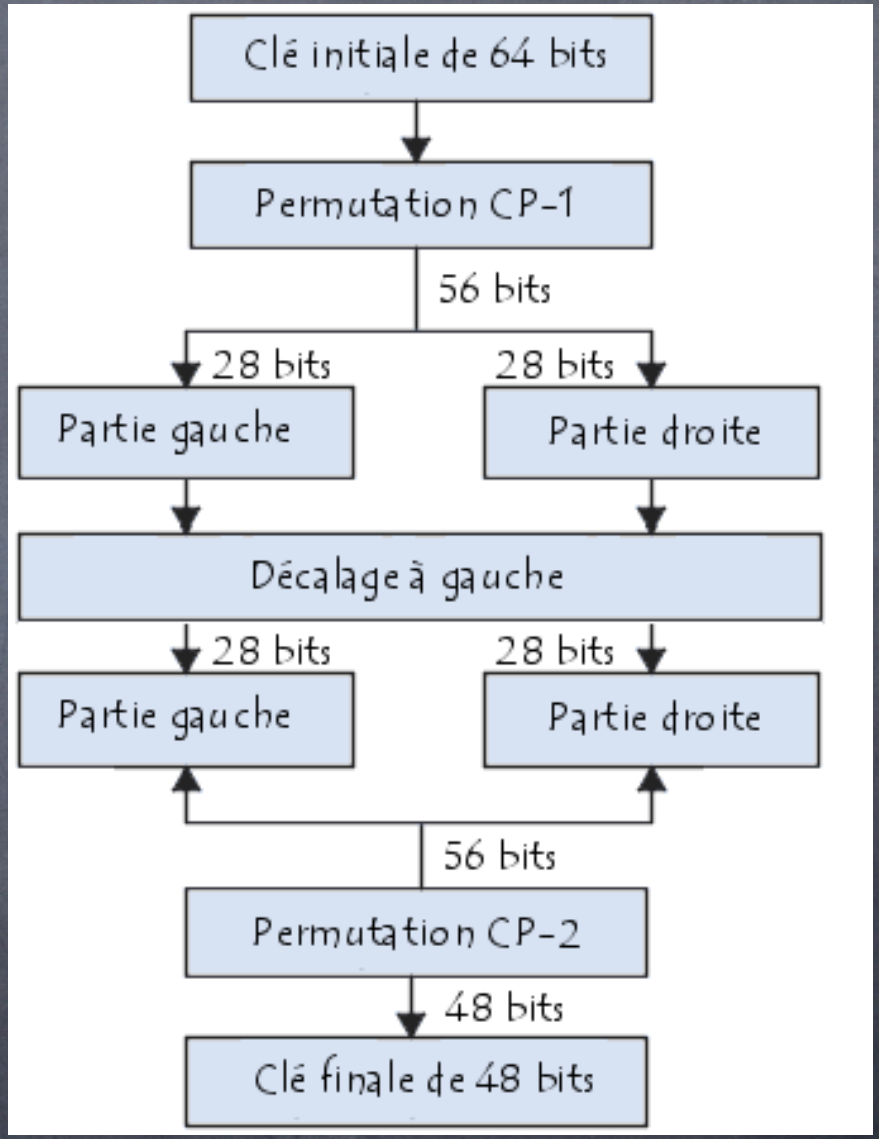
Ceci applique une substitution définie par la clé de ronde

Les clés DES sont codées sur 64 bits (8 octets) mais chaque octet contient un bit de parité (de sorte que la parité de chaque bloc est impaire). Le résultat est une clé de 64-8=56 bits!

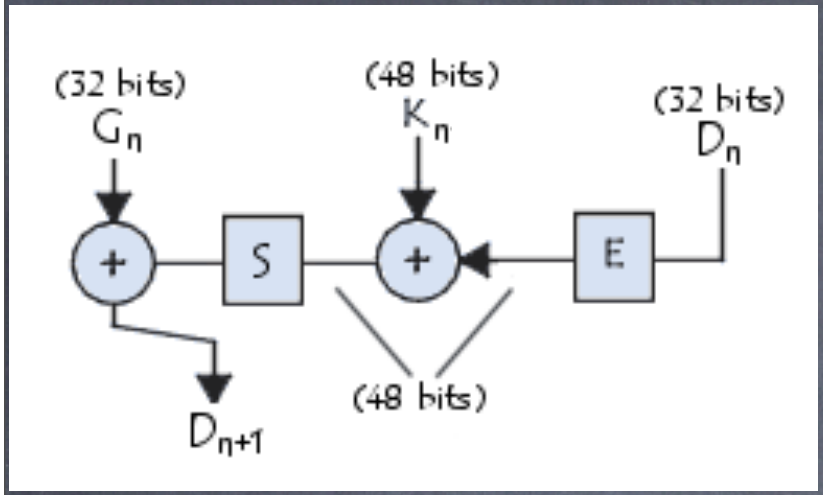
Notez que s'il y avait moins de 16 rondes, certaines attaques connues pourraient briser DES.

Dérivation des clés pour les rondes DES

- La clé de 56 bits est permutée.
- Elle est ensuite séparée en 2 clés de 28 bits.
- À chaque ronde, chaque partie est décalée d'une ou deux positions vers la gauche (dépendant de la ronde).
- Des sous-clés de 48 bits sont extraites en prenant 24 bits dans chaque partie de 28 bits.
- Les 24 bits considérés varient en fonction de la ronde.
- Chaque bit de clé original est utilisé dans environ 14 des 16 sous-clés générées (une par ronde).



Les rondes



$F(D_n,K_n)+G_n$

E: fonction d'expansion

S: fonction de substitution

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

La fonction E double certains des 32 bits pour en obtenir 48 dans un ordre un peu modifié.

Le bloc D_n , à la sortie de E, est scindé en 8 blocs de 6 bits $D_{n1},D_{n2},\ldots$

D_{n1}=001101
rang=01
col=0110

Le sous-bloc D_{ni} est utilisé pour l'évaluation de la fonction de substitution S_i. Le premier et dernier bit du sous-bloc indiquent la rangé de S_i à consulter tandis que les 4 bits du milieu indiquent la colonne de S_i à consulter.

Nous revenons à 32 bits!!!!

S_1

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

sortie : 1101=13

L'insécurité de DES simple

- Une clé de 56 bits n'est pas suffisante pour s'assurer qu'une fouille 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) :

  72 Billiards

  - De nombreux processeurs peuvent déchiffrer plus de 92 milliards de clés/sec. Un grand organisme peut accomplir une fouille 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:
  - $3DES_{KK'}(M) = DES_K(DES_{K'}^{-1}(DES_K(M)))$.

  2 chiffrements plus 1 déchiffrement DES avec deux clés sont nécessaires pour éviter certaines attaques.

Briser DES (56 bits)

Coût : 250 000 $

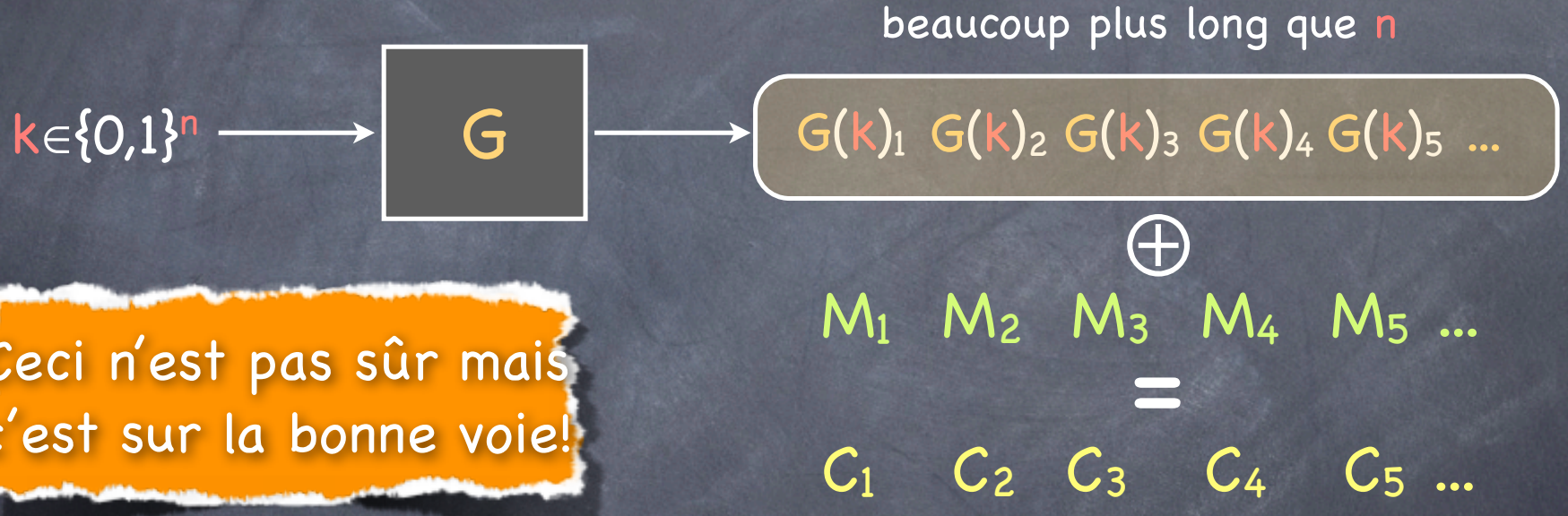
Cherche 92 milliards de clés/sec.

A déterminé une clé en 56 heures.

Kocher et Jaffe ont gagné 10 000 $ en 1998 : Défi DES organisé par RSA.

Chiffrement de flux («streamcipher»)

Un streamcipher est un chiffre qui permet le chiffrement de messages de longueurs arbitraires. Il est construit à partir d'un générateur pseudo-aléatoire G(k) :

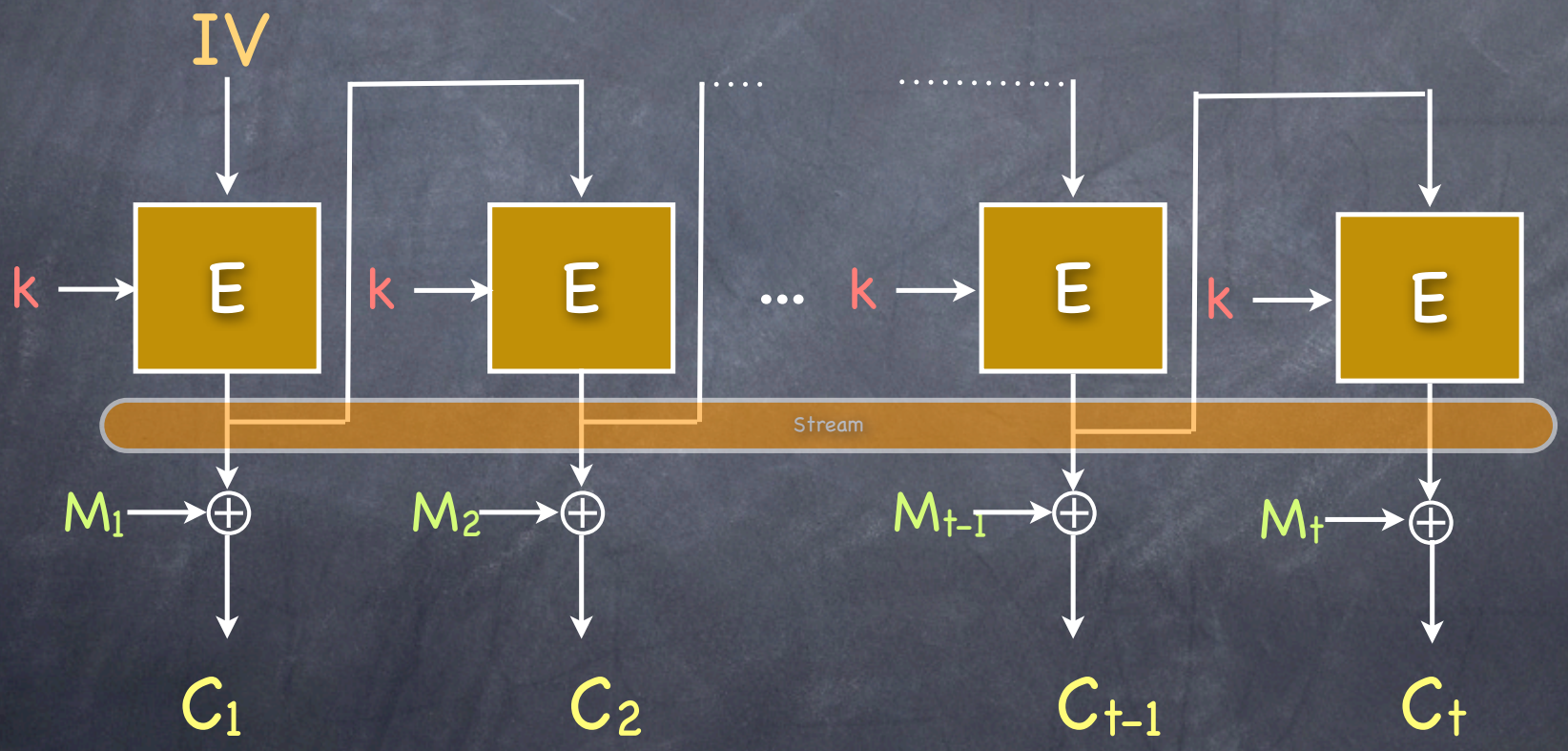


Ceci n'est pas sûr mais c'est sur la bonne voie!

G doit être tel qu'il est impossible de faire la différence efficacement entre G(k), lorsque k est choisi aléatoirement, et une vraie séquence aléatoire de longueur beaucoup plus grande que n.

OFB et streamcipher

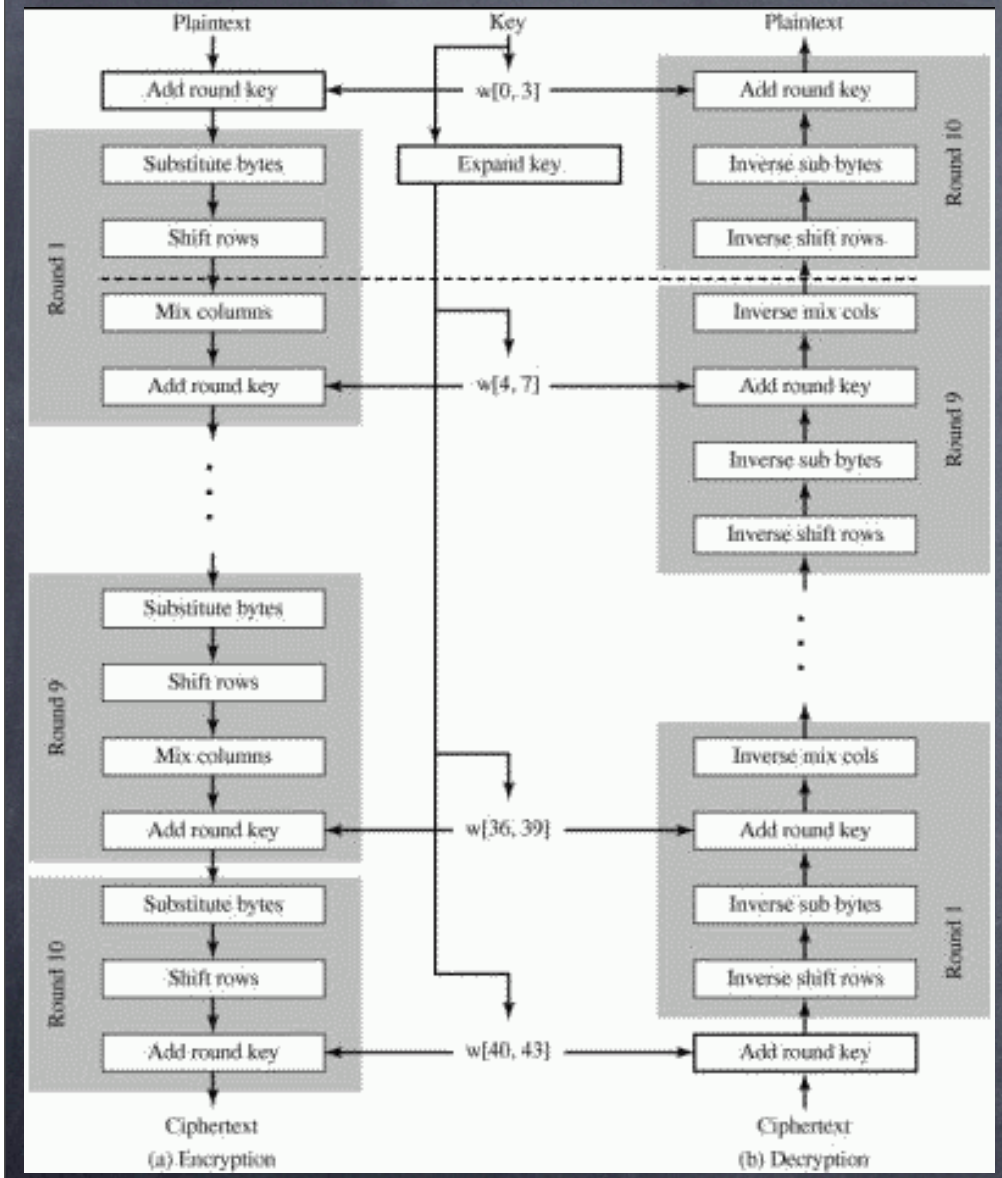
Le mode OFB consiste à transformer un chiffre par bloc en un chiffre de flux («streamcipher») :



AES: Advanced Encryption Standard

- Le système DES a été remplacé pour un nouveau système appelé AES : Advanced Encryption Standard.
- Il est devenu le nouveau standard NIST en 2001.
- Sa conception est similaire à celle de DES.
  - Il chiffre des blocs de 128 bits
  - avec des clés secrètes de 128, 192 ou 256 bits.
- Consomme peu de mémoire et est très efficace.
- Évidemment, il n'est pas plus sûr que DES, si un bon mode de fonctionnement n'est pas utilisé...

AES

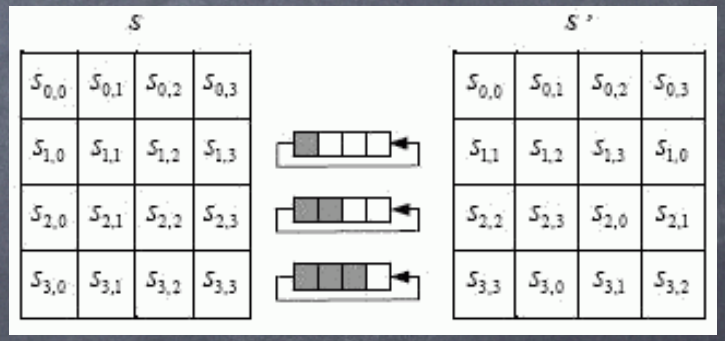


AddRoundKey : Calcul des XOR des octets du bloc à chiffrer avec la matrice de la clé courante.

SubByte : Substitue chaque octet du bloc à chiffrer.

Voyons le bloc courant comme une matrice 4X4 d'octets :

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



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

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Chiffrement d'un flux de données («streamcipher»)

- Certaines méthodes de chiffrement sont spécifiquement conçues pour chiffrer un flux de données continu par opposition aux méthodes par blocs comme DES et AES.
- L'idée est la suivante :
  - Une séquence pseudo-aléatoire est générée à partir d'une valeur initiale IV et d'une clé K en utilisant un générateur G :
    - $G_K(IV) = r_1, r_2, r_3, \ldots$
  - Le i-ième bit du message m_i est chiffré par
    - $c_i = r_i \oplus m_i$
  - Le message chiffré $C=(IV, c_1, c_2, c_3, \ldots)$ est alors transmis.

Cette séquence est appelée flux de clés («keystream»).

Un chiffre par blocs en mode OFB est presque un chiffrement de flux.

Chiffrement de flux (II)

- Le chiffrement de flux de données défini précédemment est du type asynchrone puisque le flux de clés est indépendant du message clair.
  - Un problème est que la perte/l'ajout d'un bit au cryptogramme rend le décodage très problématique...
  - Par contre, un bit d'erreur n'affecte que le bit du message clair correspondant. Ceci ouvre la porte aux attaques qui modifient quelques bits du message transmis.
- Il y a aussi des méthodes de chiffrement de flux s
permettent une resynchronisation après un certain
- La différence entre un chiffre par flux et un chiffre par blocs est que ce dernier demande au message clair d'être au moins de la taille du bloc. Les messages plus courts doivent être soumis au remplissage («padding»). Ceci est inutile pour le chiffrement de flux.

Imaginez que les données sont reçus et chiffrées par DES en paquets de 32bits. Ceci produira un cryptogramme 2 fois plus long que celui par flot.

RC4

- RC4 est le chiffre de flux le plus connu. Il s'agit du Rivest Cipher 4 développé pour RSA en 1987.
- Le code de RC4 était maintenu secret jusqu'à ce qu'un cyberpunk le rende public en 1993.
- Utilisé par exemple pour WEP, WPA (réseaux sans fil) et SSL.
- Très très rapide.
- Des faiblesses sont connues. Les premiers octets de la séquence pseudo-aléatoire (flux de clés) doivent être éliminés (1024!) parce que biaisés.

RC4 (II)

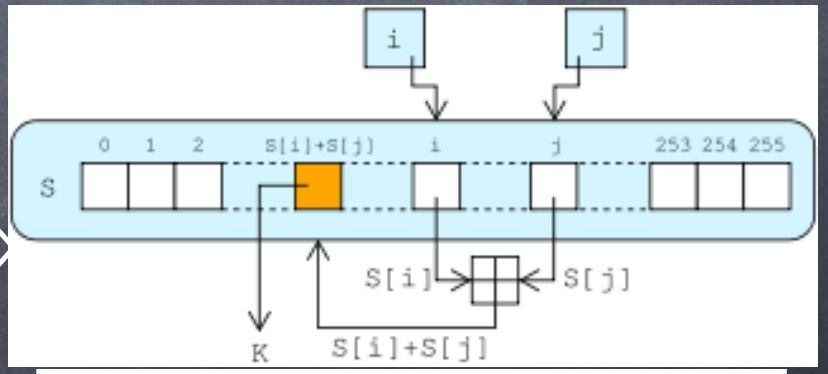
Génération de clés (key scheduling):

```
for i from 0 to 255
    S[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap(S[i],S[j])
endfor
```

Initialise un tableau S (une permutation) à l'aide d'une clé de 1 à 256 octets. La plupart du temps entre 5 et 16.

Flux de clés (keystream):

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap(S[i],S[j])
    output S[(S[i] + S[j]) mod 256]
endwhile
```



La sortie (un nouveau bit de flux de clés) est obtenue en regardant S(S(i)+S(j) mod 256). S(i) et S(j) sont permutés...

Utilisation de RC4

- Attention : si deux flux de données sont chiffrés avec la même clé, alors il faut faire quelque chose pour éviter que la même séquence aléatoire soit générée...
- Il faut donc faire intervenir une valeur IV (RC4 n'en a pas directement) utilisée une seule fois seulement avec la même clé K. Il faut aussi indiquer comment générer une nouvelle clé K' en combinant la clé initiale (long terme) K avec IV :
  - Une approche est l'utilisation d'une fonction de hachage H et poser K' = H(K,IV). Il faut utiliser une bonne fonction de hachage.
  - Une autre approche est l'utilisation de K'=(K,IV) avec IV public... ceci cause des problèmes étant donné la faiblesse relative de RC4.