



Cryptographie symétrique

Partie 2: Cryptographie moderne

Prof: Mme F. Omary

Année: 2020-2021

Introduction

🔦 Historique

Il existe deux types de chiffrement symétriques:

- **Chiffrement par blocs**: les messages sont découpés en blocs
 - DES: blocs de 64 bits, clés de 64 bits
 - AES: blocs de 128 bits, clés de 128 ou 192 ou 256
- **Chiffrement en continu** ou par flot ou par flux
 - Traitement bit par bit
 - RC4: chiffrement octet par octet

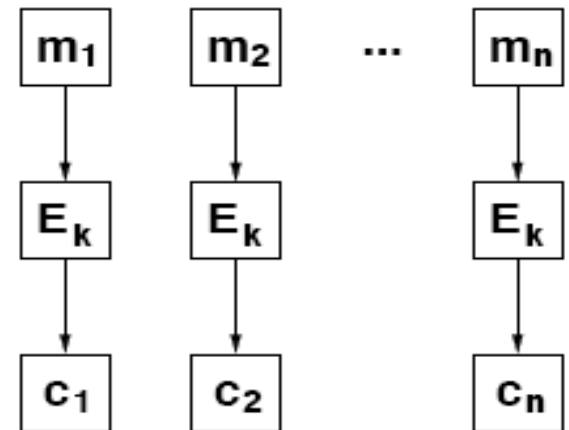
Modes de chiffrement par blocs

• **ECB** (Electronic CodeBook)

$$C_i = E_k(M_i)$$

$$M_i = D_k(C_i)$$

- Il permet le chiffrement en parallèle des différents blocs
- Un bloc est toujours chiffré identiquement
- Aucune sécurité, pas d'utilisation



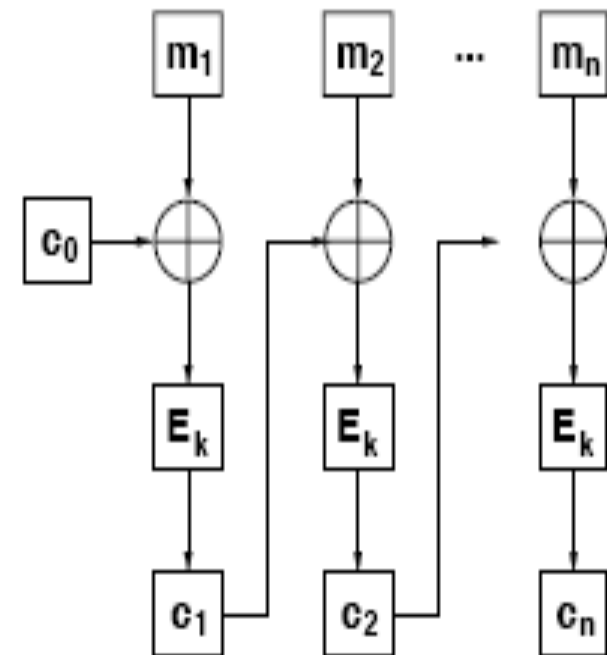
Modes de chiffrement par blocs (2)

🔦 **CBC** (Cipher Block Chaining)

$$C_i = E_k(M_i \oplus C_{i-1})$$

$$M_i = C_{i-1} \oplus D_k(C_i)$$

- Mode le plus utilisé
- Chaque C_i dépend non seulement du M_i correspondant, mais aussi des blocs précédents.
- Le premier bloc du texte en clair est combiné avec un bloc appelé vecteur d'initialisation C_0



Modes de chiffrement par blocs(3)

☀ **CFB**: chiffrement à rétroaction

Agit comme un chiffrement par flux

☀ **OFB**: chiffrement à rétroaction de sortie

Similaire à un chiffrement par flux

☀ **CTR**: chiffrement basé sur un compteur

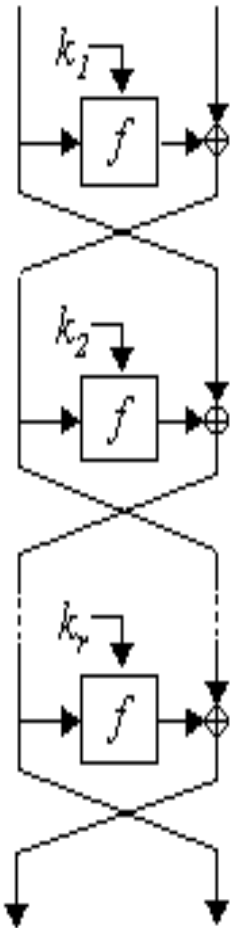
☀ (à étudier en séance d'application)

Modes de chiffrement par blocs (3)

✿ Chiffrement par *blocs avec itération* :

- Chiffrement des blocs par un processus comportant plusieurs rondes.
- Dans chaque ronde , la même transformation est appliqué au bloc en utilisant une sous clé dérivée de la clé secrète.
- Un nombre de ronde plus élevé garantit une meilleure sécurité

Modes de chiffrement par blocs (4)



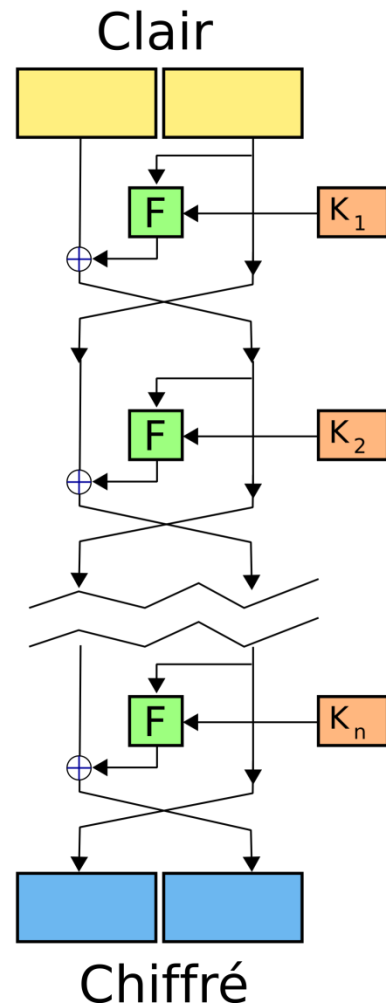
Chiffrement de Feistel

- Cas particulier de chiffrement par blocs avec itération
- Un bloc de texte en clair est découpé en deux
- La transformation de ronde est appliquée à une des deux moitiés, le résultat est combiné avec l'autre moitié par ou exclusif
- Les deux moitiés sont alors inversés pour la ronde suivante.

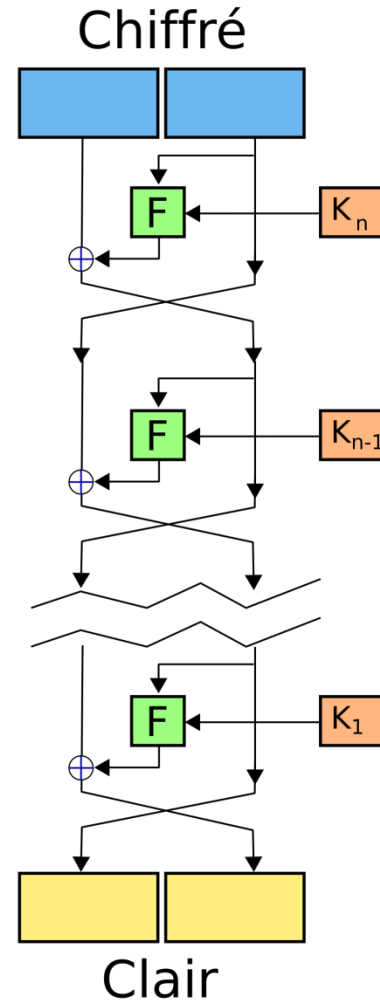
Exemples: DES (*Data Encryption Standard*), DES triple, RC5 (*Rivest's Code5*)

Schéma de Feistel

CHIFFREMENT



DÉCHIFFREMENT





Principe de Sécurité

Principe de Auguste Kerckhoffs (1883)

1. La sécurité repose sur le secret de la clef et non sur le secret de l'algorithme
2. Le déchiffrement sans la clef doit être impossible (En un temps raisonnable)
3. Trouver la clef à partir du clair et du chiffré est impossible (En un temps raisonnable)

DES (Data Encryption Standard) 1977

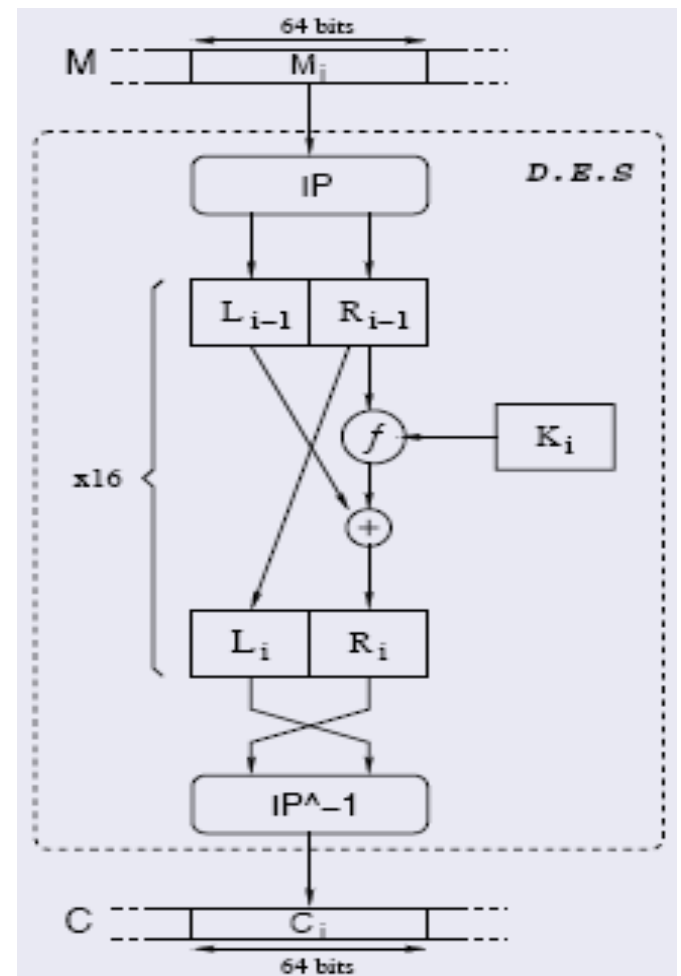
- Algorithme de chiffrement par blocs
- Agit sur des blocs de 64 bits
- C'est un chiffre de Feistel à 16 rondes
- La longueur de la clef est de 64 bits (dont 8 bit de parité)
- Les sous-clefs utilisées par chaque ronde ont une longueur de 48 bits.
- Structure générale:
 - Permutation initiale IP
 - 16 rondes « de Feistel »:

DES (1977)

$$L_i = R_{i-1}$$

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

- Permutation finale IP^{-1}



Un mot sur: permutations de DES

UN mot sur les permutations du DES

— Signification dans le calcul de

— $Y = IP(X)$:

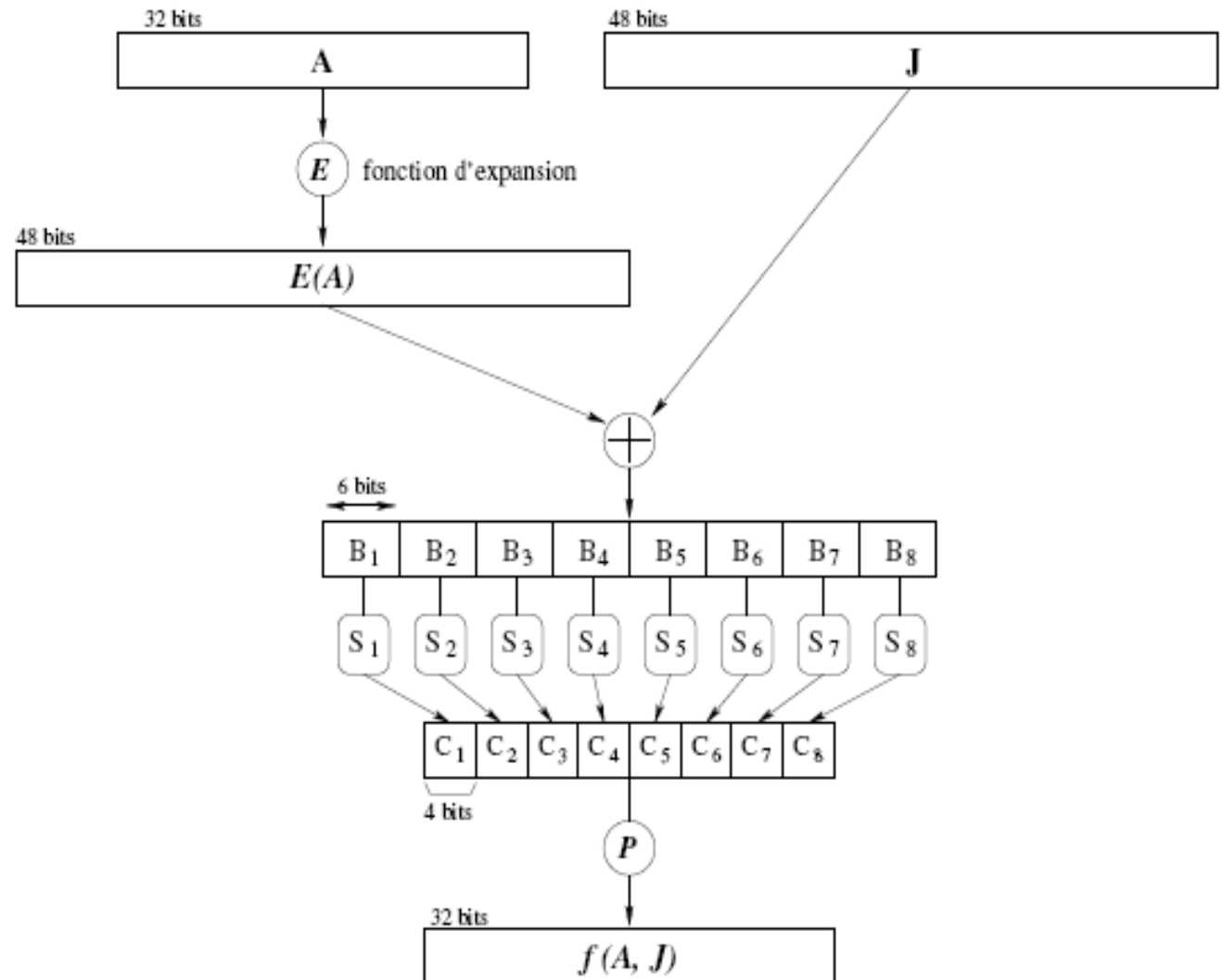
Le 58ième bit de X est:
le premier de Y

Le 50 ième bit de X est
le second de Y

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

La fonction $f(A, J)$ de DES



La fonction f de DES (suite)

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

La fonction f du DES (suite)

✚ Mathématiquement:

$$f : \{0,1\}^{32} \times \{0,1\}^{48} \longrightarrow \{0,1\}^{32}$$

- prend en entrée une chaîne de 32 bits (moitié de droite du texte courant) et une sous- clef dont la taille est 48 bits.
- donne comme sortie une chaîne de 32 bits

Détail du calcul autour des S-Box

8 « boîtes-S » S_1, S_2, \dots, S_8

tableaux $4 * 16$ entiers compris entre 0 et 15

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

Soit une sous chaîne de six bits $B_i = b_1.b_2.b_3.b_4.b_5.b_6$

- $b_1.b_6$ = indice de ligne i de S_i à considérer $0 \leq i \leq 3$
- $b_2.b_3.b_4.b_5$ = colonne c de S_i à considérer $0 \leq c \leq 15$
- A l'intersection de i et c : $C_i = S_i(B_i)$!

On obtient ainsi $C = C_1.C_2 \dots C_8$, chaîne de 32 bits

On applique la permutation P : $P(C) = f(A, J)$

Caractéristiques des boîtes de DES

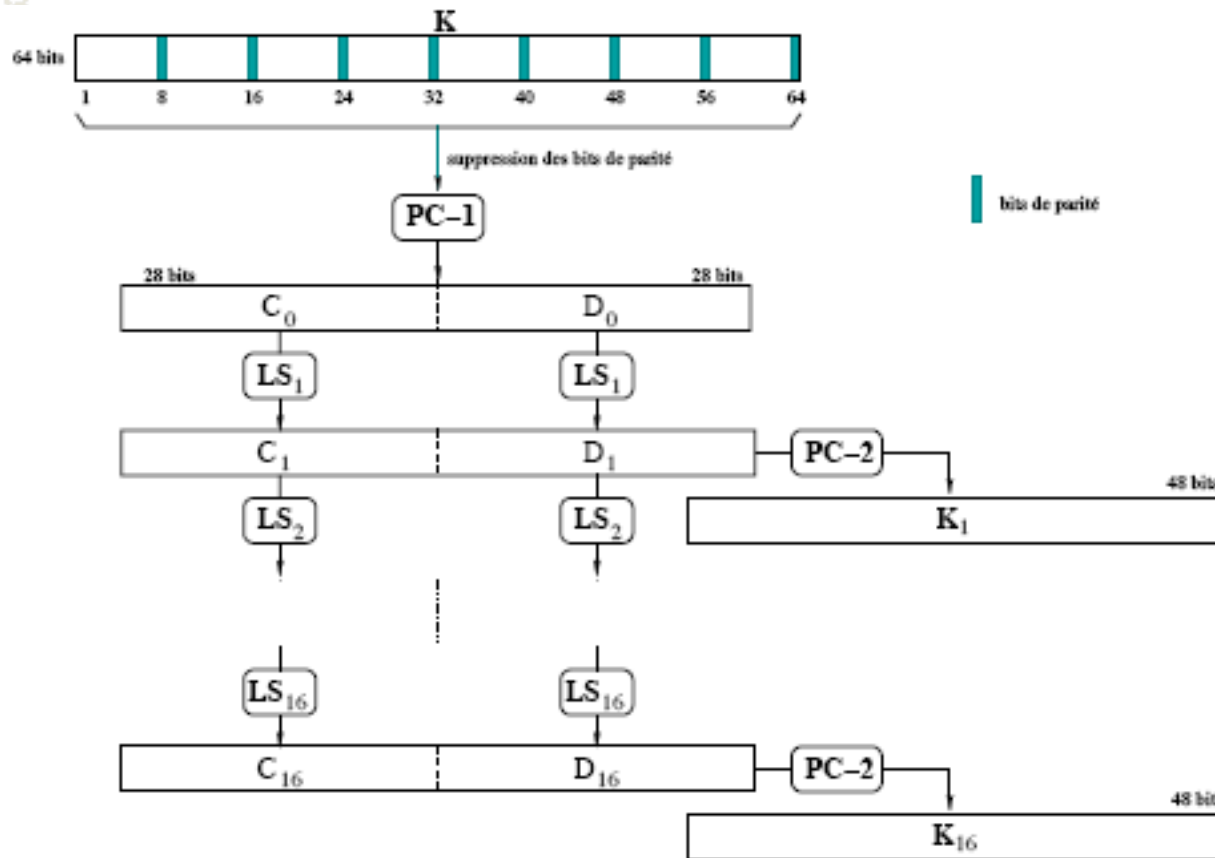
✶ Propriétés cryptanalytiques intéressantes

- Non linéaire (substitution différente de César ou Vernam) : pas d'attaque simple
- Spécialement conçues pour contrer la cryptanalyse différentielle [Coppersmith]
- Même de très petites modifications des S-box peuvent affaiblir considérablement le chiffre

✶ A la base de controverse autour de DES

- Cf secret entourant la génération de $\{S_i\}_{1 \leq i \leq 8}$

Diversification des clés dans DES



Un mot sur les clefs

- Le décalage à gauche LS_i , par permutation circulaire, de v_i positions est appliqué, comme suit:

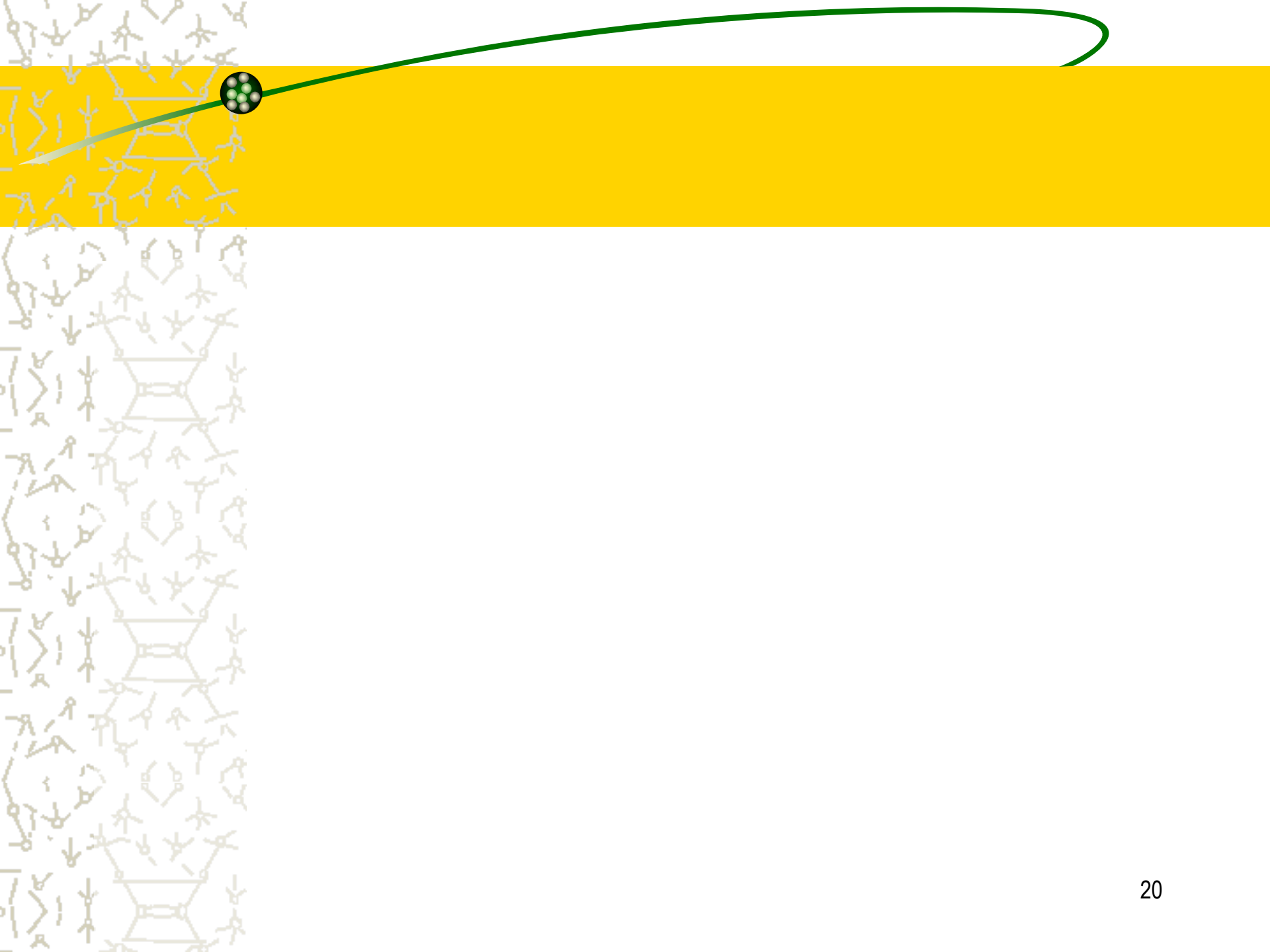
$$\begin{cases} v_i=1 & \text{pour } i \text{ dans } \{1, 2, 9, 16\} \\ v_i=2 & \text{sinon} \end{cases}$$

- La fonction PC-1 (définie par 1 matrice cf annexe):

À un mot binaire K , de 64 bits $\Longrightarrow C_0$ et D_0 de 28 respectivement

- La fonction PC-2(cf annexe):

À un couple (C,D) de 28(resp) \Longrightarrow Un mot de 48 bits





Exemples d'utilisation de DES

- ☛ Cartes de crédit :

 - UEPS (Universal Electronic payment system)

- ☛ Protocole d'authentification sur réseaux

- ☛ Messagerie électronique :

 - PEM (Privacy-Enhanced Mail)

Déchiffrement dans DES

✪ Pour déchiffrer:

- On utilise la même structure algorithmique que le chiffrement, mais cette fois-ci en inversant l'ordre d'application des clés
- \longrightarrow on commence par la clé K16 et on termine avec la clé K1.
- Voir chiffrement de Feistel : compléments de cours.

Sécurité du DES

- ☛ L'ordinateur, baptisé « DES cracker » en 1998 par l'Electronic Frontier Foundation
 - Contient 1 536 puces et peut chercher 58 milliards de clés par seconde.
 - Coût: 250 000 euros
 - Il a gagné le concours des laboratoires RSA, en trouvant une clé de DES en 56 heures, en juillet 1998.



Autres variantes de DES

- L'attaque du DES par recherche exhaustive:

- → Remplacé par: Triple-DES ou DES-EDE

- Et aussi par le fameux AES

Exemple

✚ Considérons un DES ayant pour clé:

K= 133457799BBCDFF1

✚ Le message en clair est::

M= 0123456789ABCDEF

Résultat :voir séance d'application

Chiffrement en continu ou par flux

Chiffrement en continu (stream- cipher): On chiffre bit-à-bit (ou bloc de bits de très petite taille) sans attendre la réception complète des données à chiffrer.

Principe:

- Pas besoin de lire le message ni d'avoir sa longueur pour commencer à chiffrer
- Génération de pseudo-aléa, appelé flux de clé (keystream) que l'on combine (souvent par XOR) avec le flux de données.

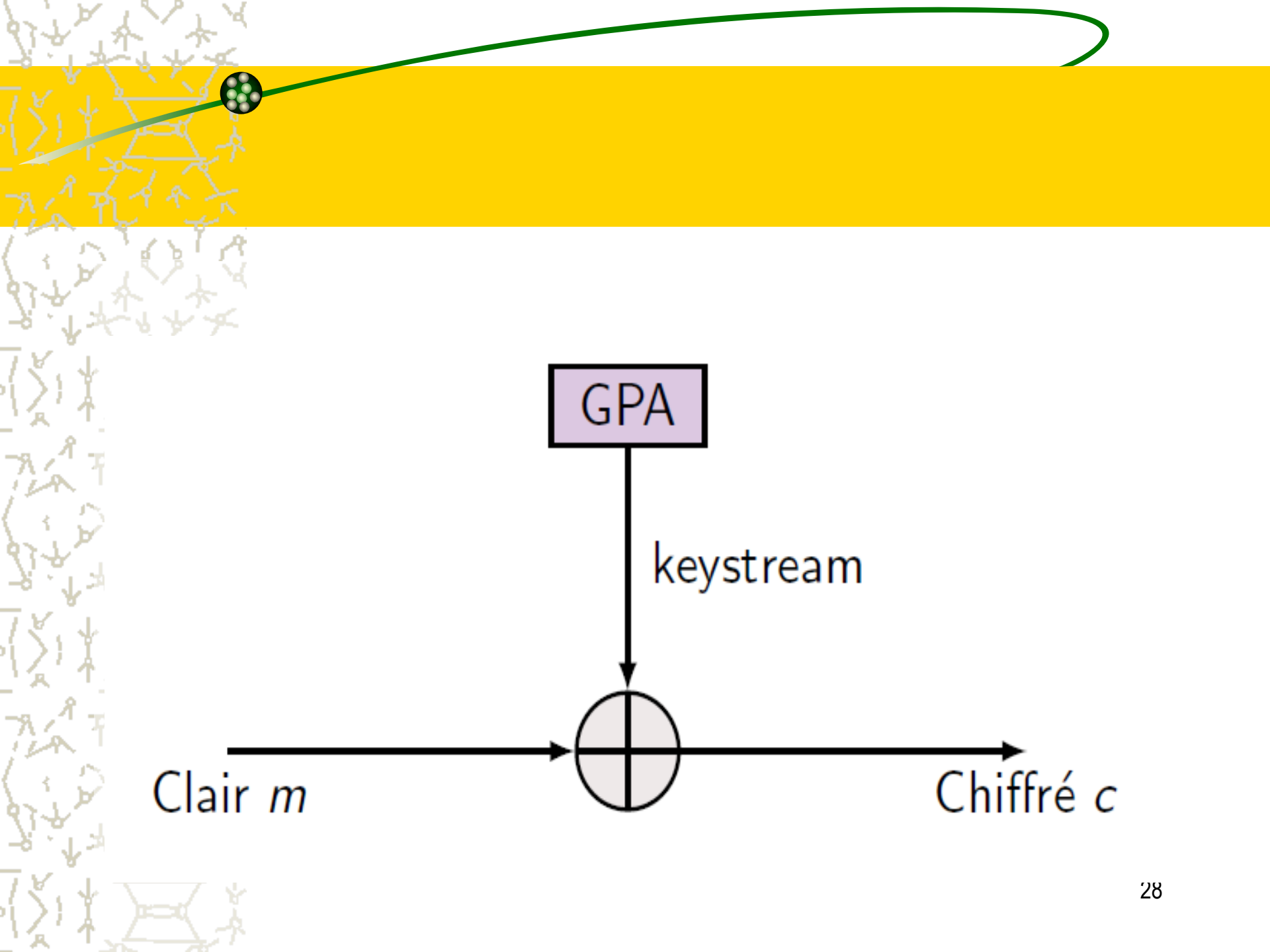
Chiffrement par flux:Inspiration

🔦 Le chiffrement de Vernam(masque jetable):

- Clé =suite aléatoire parfaite
- Clé aussi longue que le message
- Opérations très simples
- Inconditionnellement sûr

🔦 Chiffrement par pseudo-aléa

- Les chiffrements s'inspirent du chiffre de Vernam mais utilisent une suite pseudo-aléatoire, générée à partir de quelques bits de clé réellement aléatoires²⁷



Chiffrement par flux: caractéristiques

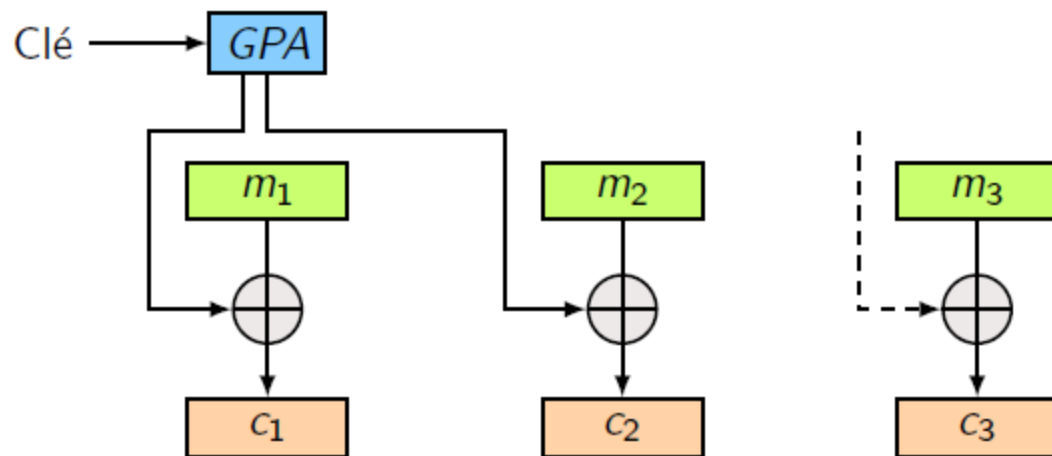
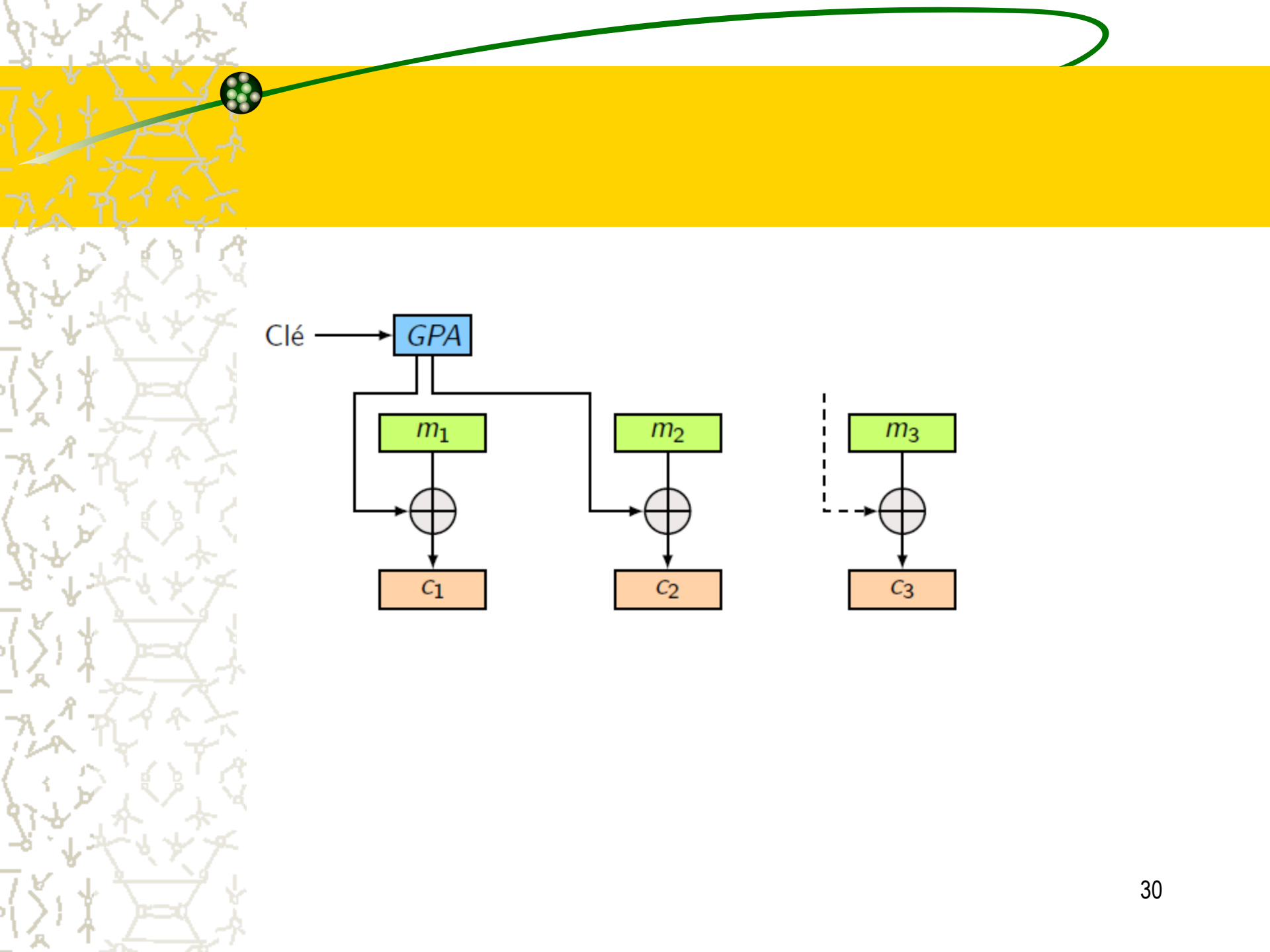
☀ Avantages:

Les stream ciphers sont de façon générale:

- Très rapides (en matériel et en logiciel)
- Implémentation matérielle simple
- Adaptés aux applications temps réel

☀ Inconvénients

- Principaux problèmes:
- Propagation d'erreur(Problème de synchronisation)
- Sécurité difficile à atteindre (pas de preuve)

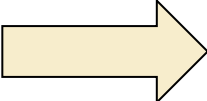


Chiffrement par flux: Usage

- ✱ Les chiffrements par flot sont très utilisés pour protéger les données multimédia:
 - RC4 (utilisé dans SSL et dans le WIFI 802.11)
 - E0/1 (norme Bluetooth)
 - A5/1, A5/2, A5/3 (utilisés dans le GSM)

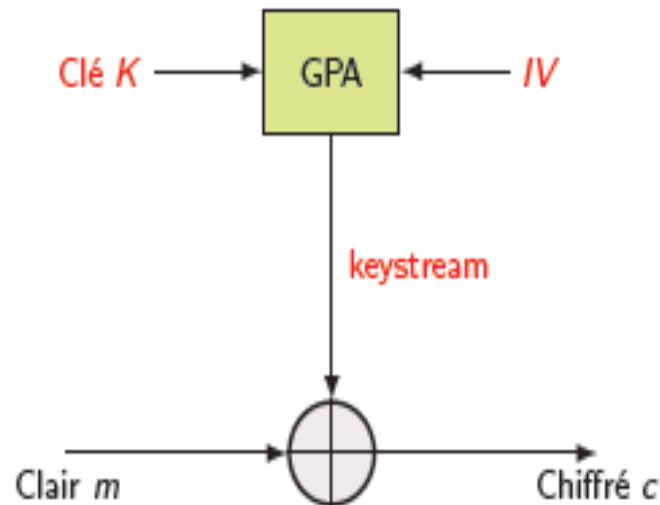
Mise en Place

❗ Problème: Si on chiffre plusieurs messages avec la même clé:

- Même clé  même aléa
- Or $\text{chiffré} = \text{clair} \oplus \text{clé(en flux)}$ d'où:
- $E_k(M_1) \oplus E_k(M_2) = M_1 \oplus M_2$
- On peut ainsi déchiffrer tout autre message chiffré par la même clé
- Solution: ne pas réinitialiser le GPA entre 2 chiffrements?
 - pas réaliste(obligation de mémoriser un état)

Mise ne place: Vecteur d'initialisation

- ✶ Solution: On utilise une entrée auxiliaire IV (vecteur d'initialisation)
- ✶ Une telle clé à usage unique, pour chiffrer un seul message, s'appelle: **clé de session**



Générer une clé de session

- ✱ Lorsqu'on partage déjà une clé symétrique avec le destinataire, on la combine avec un **nonce** à usage unique('number used once') appelé également **vecteur d'initialisation**
 - La clé de session est le résultat d'un hachage cryptographique à sens unique de:
la clé symétrique et du nonce
 - Le nonce est transmis en clair au destinataire
- ✱ Générer une clé de session aléatoire et l'échanger via un cryptosystème à clé publique

Mise en place: Fabrication d'un GPA

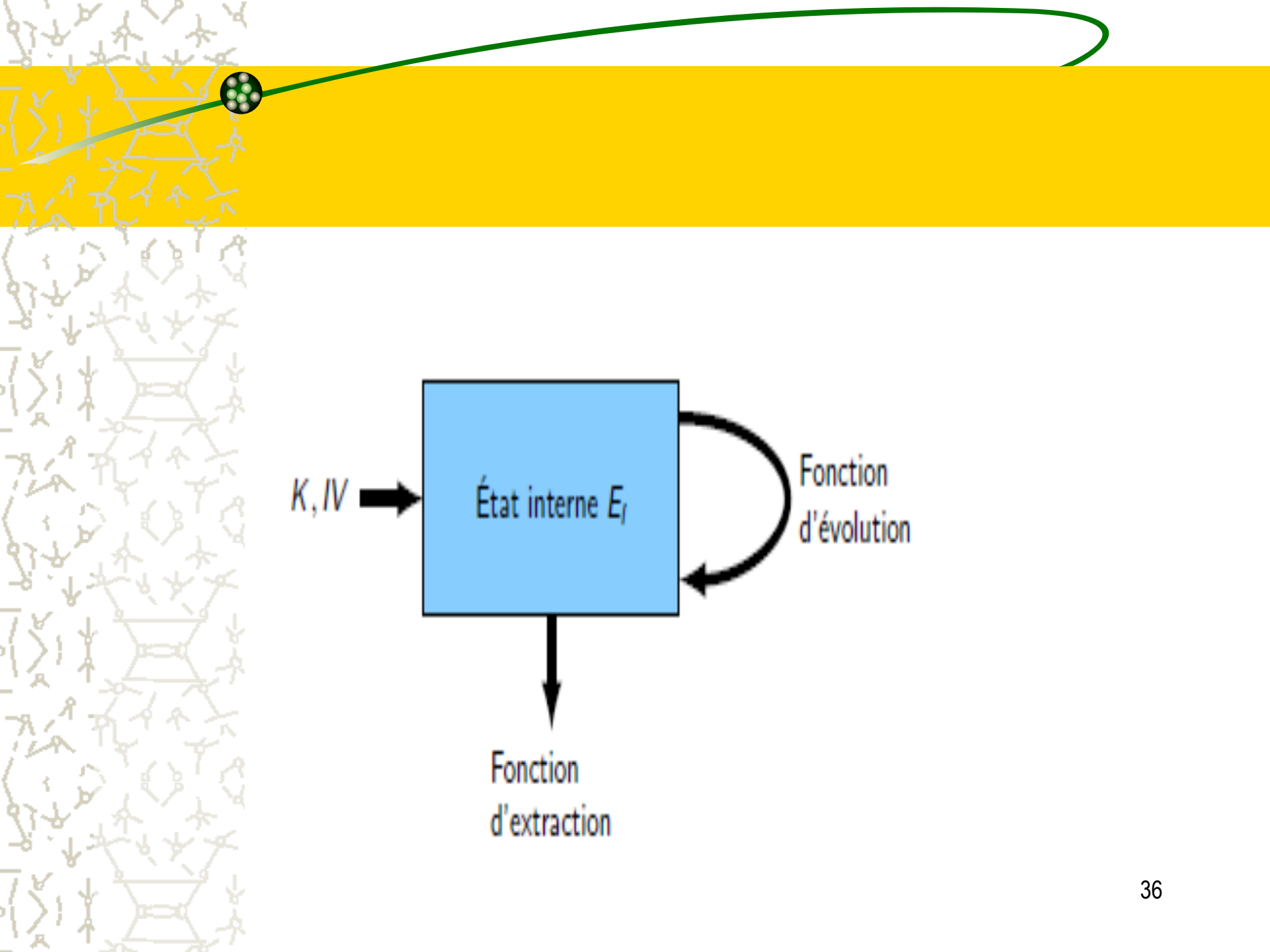
✶ Un **GPA** sera constitué de

- 1. Etat interne suffisamment grand
- 2. Fonction d'évolution
- 3. Fonction d'extraction

La qualité d'un **GPA** est définie par sa période de retour

✶ Paramètres Importants:

- Taille de la clé: recherche exhaustive
- Taille de l'état interne: recherche exhaustive
- Taille de 'IV': recherche en collisions



Algorithmes de chiffrement par Flux

☀ Deux grandes catégories:

- Chiffrements **synchrones**: la sortie du GPA ne dépend que de son état interne
 - Problème de synchronisation
 - Exemples: modes OFB et CTR d'un chiffrement par bloc
- Chiffrements **asynchrones**: la sortie du GPA dépend de son état interne et de plusieurs symboles du message
 - Exemples: mode CFB d'un chiffrement bloc

Chiffrement synchrone(1)

Caractéristiques:

- l'expéditeur et le destinataire calculent les flux de clés de façon synchrone
- Le flux de clés dépend seulement de la clé secrète choisie mais pas du message clair et du chiffré.

Exemples:

- **Exemple de générateur de flux de clés:** soit n un entier positif et

$K = \{0,1\}^n$ l'espace des clés; $\Sigma = \{0,1\}$ l'alphabet des messages

Chiffrement synchrone(2)

Si $K = k_1 k_2 \dots k_n$ alors on peut définir un générateur de flux de clés comme suit :

pour $1 \leq i \leq n$ $s_i = k_i$

pour $i > n$ $s_i = \sum_{j=1}^n c_j s_{i-j} \mod 2$

Où c_1, c_2, \dots, c_n sont des coefficients fixés

Voir exemple concret dans ce qui suit.

Chiffrement synchrone (4)

Formellement: l'algorithme de chiffrement en continu opère de la manière suivante:

- Etant donné une clé K , l'algorithme calcule le flux de clés:

$$g(K) = S_1 S_2 \dots$$

- Puis il utilise des fonctions

$$e_s: \Sigma \rightarrow \Sigma \quad (s \text{ appartient à } \Sigma)$$

qui chiffrent le message en clair:

$$M = X_1 X_2 \dots X_m$$

Chiffrement synchrone(5)

$$E(M) = e_{s1}(X_1)e_{s2}(X_2) \dots e_{sm}(X_m)$$

• Exemple concret:

Chiffrement synchrone(6)

✚ Concrétisation de l'exemple:

$n = 4$ avec $K = 1000$ comme clé initiale

et $c_1 = c_2 = 0, c_3 = c_4 = 1$

- Avec la récurrence linéaire d'ordre n définie ci-dessus, on obtient le flux de clés:

1,0,0,0,1,0,0,1,1,0 1,0,1,1,1,1,0,0,0,

- Ce flux de clés est périodique de période 15.
- Pour $\Sigma = \{0,1\}$

Chiffrement synchrone(7)

✚ On définit :

- $e_s(X) = S \oplus X$

- Ainsi le message : $M = X_1 X_2 \dots X_m$ sera chiffré par le biais de la clé $s_1 s_2 \dots s_m$: pour obtenir:

$$Ek(M) = Y_1 Y_2 \dots Y_m \text{ où}$$

$$Y_1 = s_1 \oplus X_1 ; Y_2 = s_2 \oplus X_2 \dots$$

Exemple: Algorithme RC4

- ✱ Il est développé par Ronald Rivest, en 1987
- RC= Rivest Cipher(ou encore Ron's Code)
- Sa clé de taille variable:de 1 à 256 octets
- Breveté et tenu secret par la société RSA
- Détails postés anonymement sur Usenet en 1994
- Mais jamais approuvés officiellement par RSA
- Période du générateur est supérieure à 10^{100}

Algorithme RC4 (1)

✿ Largement utilisé:

- WEP(Wired Equivalent Privacy), WPA(Wi-Fi Protected Access), SSL/TLS, Oracle SQL...
- Grande simplicité, facilité de mise en œuvre
- Excellentes performances(vitesse de chiffrement)

Algorithme RC4(3)

✱ Du point de vue technique:

1. RC4 est un chiffrement par flux (octet par octet)

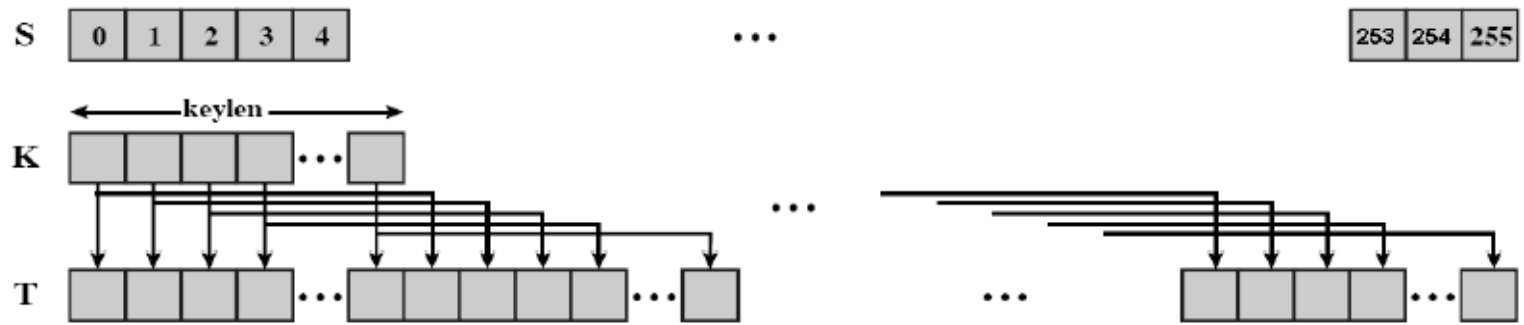
- Permet d'atteindre des performances logicielles

2. Implémente un générateur pseudo-aléatoire

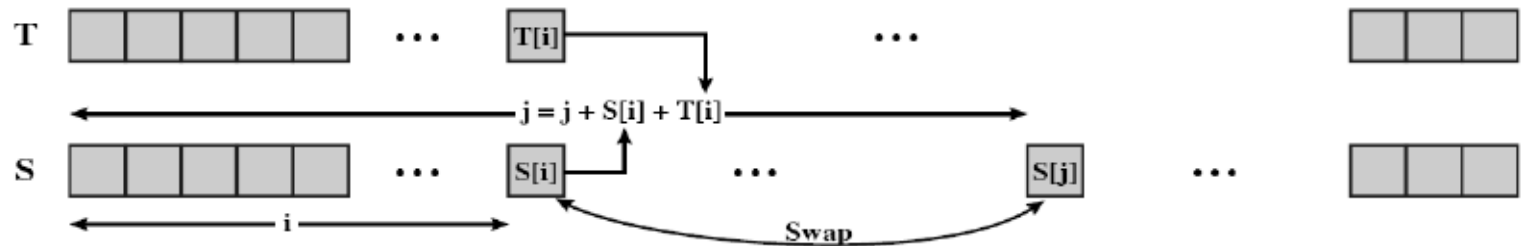
- État interne = un tableau codant une permutation de 256 octets
- Deux pointeurs i et j sur des cases du tableau.

✱ La clé est utilisée pour mélanger le tableau le mieux possible, puis le tableau (la permutation) évolue à chaque tour.

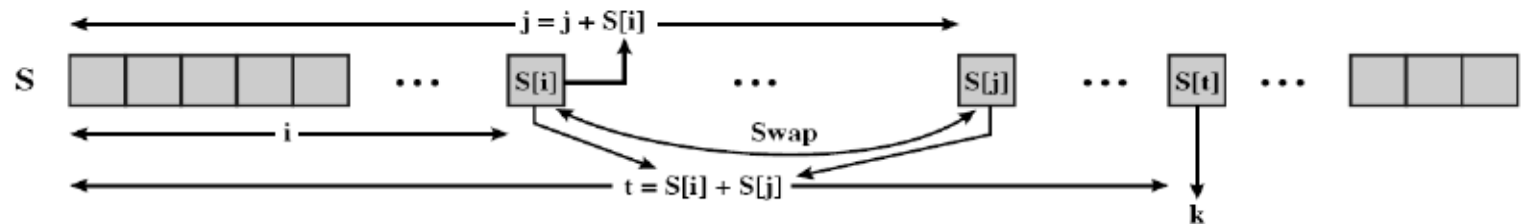
Schéma :RC4



(a) Initial state of S and T



(b) Initial permutation of S



(c) Stream Generation

Description formelle de RC4(1)

- ✱ La clé est utilisée pour initialiser un vecteur T de 256 octets.
- ✱ A tout moment S contient une permutation de toutes les cellules le composant.
- ✱ Etape 1: Initialisation(pour K de longueur l)
- ✱ Les cellules de S reçoivent une valeur égale à leur position
- ✱ Un vecteur temporaire de longueur T(égale à S) est destiné à recevoir la clé.

Description formelle de RC4 (2)

- ✱ K est simplement transféré dans T.
- ✱ Si $|K|$ est inférieur à 256 octets, elle est copiée dans T jusqu'à atteindre la taille de T.
- ✱ Ce procédé est illustré au point (a).

For $i = 0$ to 255 Do

$S[i] = i;$

$T[i] = K[i \bmod |K|]$




Description formelle de RC4(3)



Etape 2: Permutation initiale de S

- T est ensuite utilisé pour produire la permutation initiale de S
- Chaque cellule $S[i]$ de S, sera échangée avec une autre cellule de S selon un calcul basé sur la valeur de $T[i]$ correspondante
- Le point (b) illustre cette phase.

Description formelle de RC4 (4)

 $j=0;$

FOR $i=0$ to 255 Do

$j = (j + S[i] + T[i]) \bmod 256$


SWAP ($S[i], S[j]$);

Description formelle de RC4(5)

➤ Etape 3: Génération du flux:

- A partir de cet instant la clé initiale n'est plus utilisée
- Chaque $S[i]$ sera échangé avec un autre octet de S , selon un schéma basé sur la configuration courante de S . Le processus redémarre à la cellule $S[0]$, une fois arrivé à $S[255]$
- La valeur de sortie (output) du générateur de flux est alors utilisée pour le chiffrement (du prochain octet du texte clair), ou pour le déchiffrement.
- Le point © illustre cela

Description formelle de RC4(6)

 $i := 0;$

$J := 0;$

While génération do

$i := i + 1 \quad \text{mod } 256$

$J := j + S[i] \quad \text{mod } 256$

Swap ($S[i]$, $S[j]$)

Output $S[S[i] + S[j] \text{ mod } 256]$



Chiffrement/déchiffrement

Etape 4: Chiffrement (ou déchiffrement)

L'octet output (étape 3) sera Xoré avec l'octet d'entrée du message clair (ou chiffré) pour obtenir le chiffré correspondant

Sécurité de RC4 (1)

- ✦ En 2001 Adi & Shamir ont découvert que les valeurs des premiers octets générés du flux de clés ne sont pas équiprobables.

En effet: dans le second octet du flux

- Valeur 0 avec probabilité $1/128$ au lieu de $1/256$
- Sous certaines conditions (clés faibles)

Sécurité de RC4 (2)

⚡ Concernant le Vecteur d'initialisation: il existe des faiblesses

Fluhrer, Mantin et Shamir ont montré que:

- Certains motifs de la clé se « répercutent » dans l'état interne
- Et sont « détectables » dans le flux pseudo-aléatoire de sortie
- On en déduit un distingueur (par rapport à un aléa vrai)



Attaque de la Sécurité du **WEP**

Sécurité du RC4 (3)

- ✶ En effet : le **WEP** utilise une simple concaténation de l'IV avec la clé
 - Donc les motifs de l'IV sont directement concernés
 - De plus l'attaquant connaît l'IV (il peut même le choisir)

Attaque concrète:

Avec 6 millions de chiffrés (interceptés via une attaque passive), on a pu retrouver complètement la clé secrète.

Sécurité du RC4 (4)

✶ Concernant : l' état interne

- Le nombre de permutations de 256 valeurs est exponentielle : $256!$ de l'ordre de 2^{1684}
- donc un état interne de 1700 bits environ

✶ Mais cela n'exclut pas les **faiblesses statistiques**

- Certaines clés faibles réduisent l'espace des états qui seront atteints

C/C: remplacé par, Spritz (2014): « spongy RC4-like stream cipher and hash function », par Ronald Rivest & al..