Algorithmes de chiffrement par flot

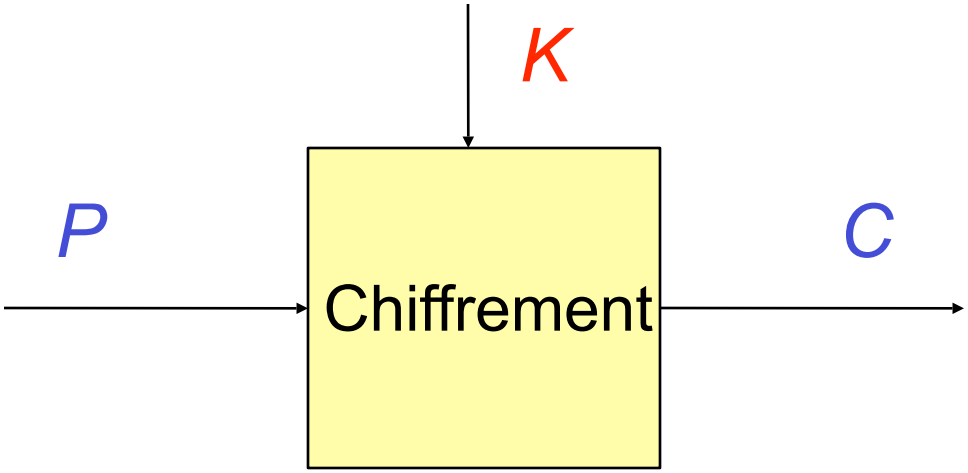
Fouque Pierre-Alain

Département d'Informatique

Ecole normale supérieure

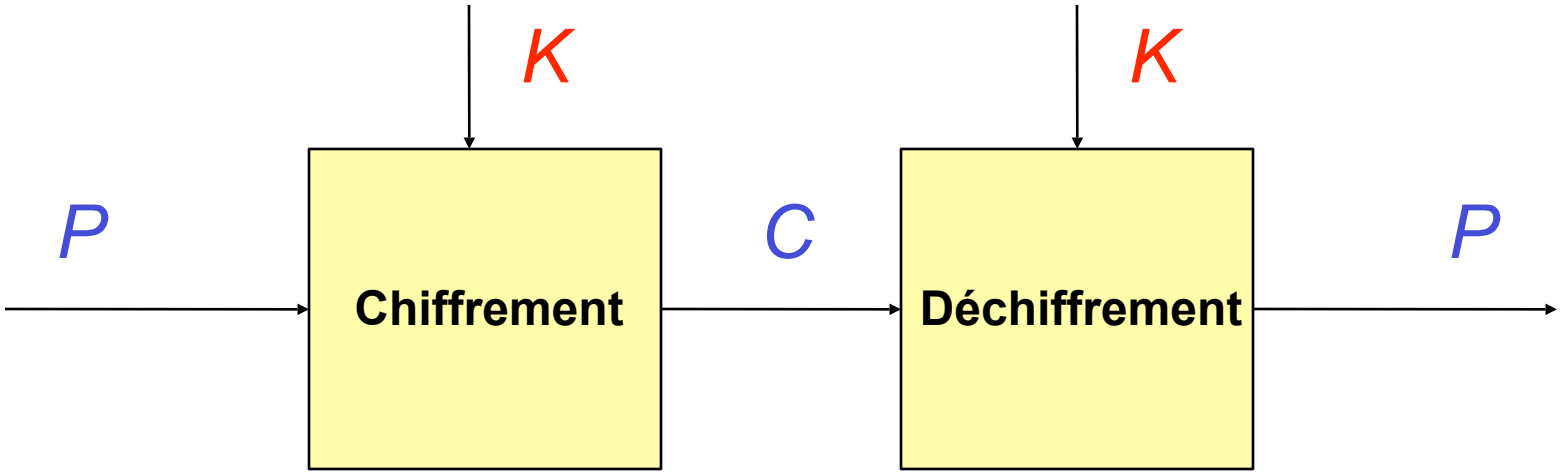
Chiffrement symétrique

Définition : *Un algorithme de chiffrement symétrique transforme un message en clair P avec une clé secrète K. Le résultat est un message chiffré C*



Chiffrement symétrique

La fonction de chiffrement doit être inversible



Deux grandes catégories

Chiffrement par bloc

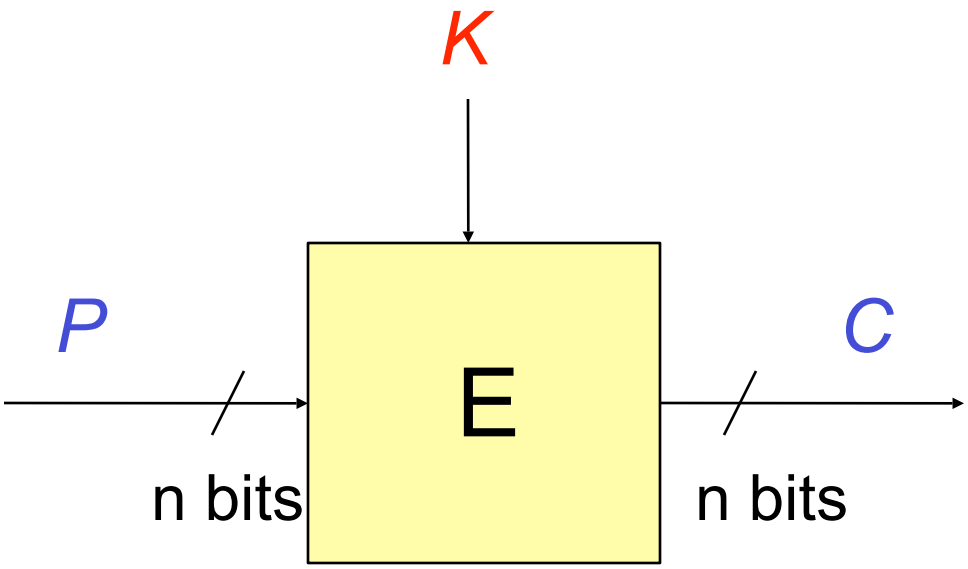
- P est traité par blocs de données (ex: 64 bits ou 128 bits)
- Algorithmes : DES, AES, IDEA, RC6, BLOWFISH, …

Chiffrement par flot

- P est traité bit par bit
- Algorithmes : RC4, Bluetooth E0/1, GSM A5/1,

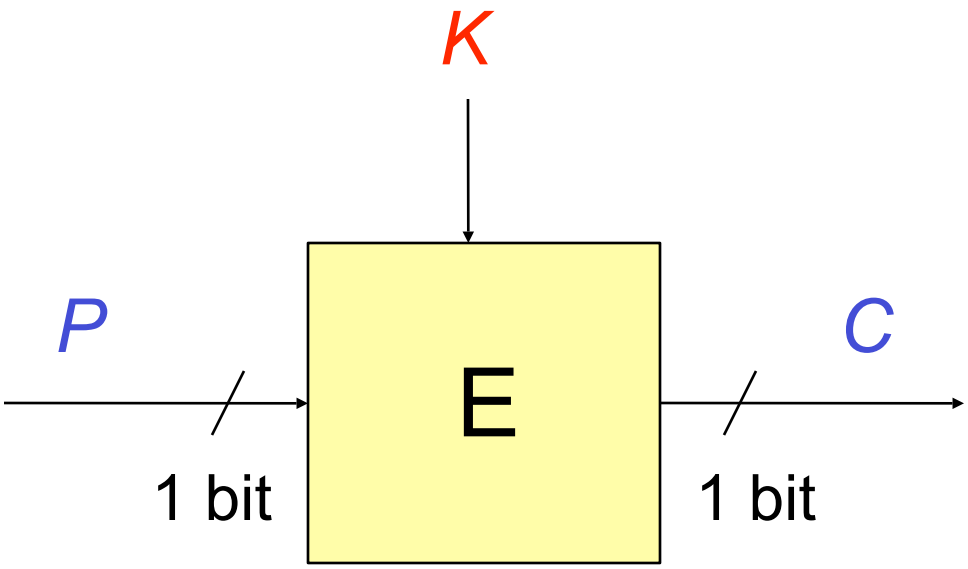
Chiffrement par bloc

- le message est traité par blocs de n bits
- E est une fonction fixe

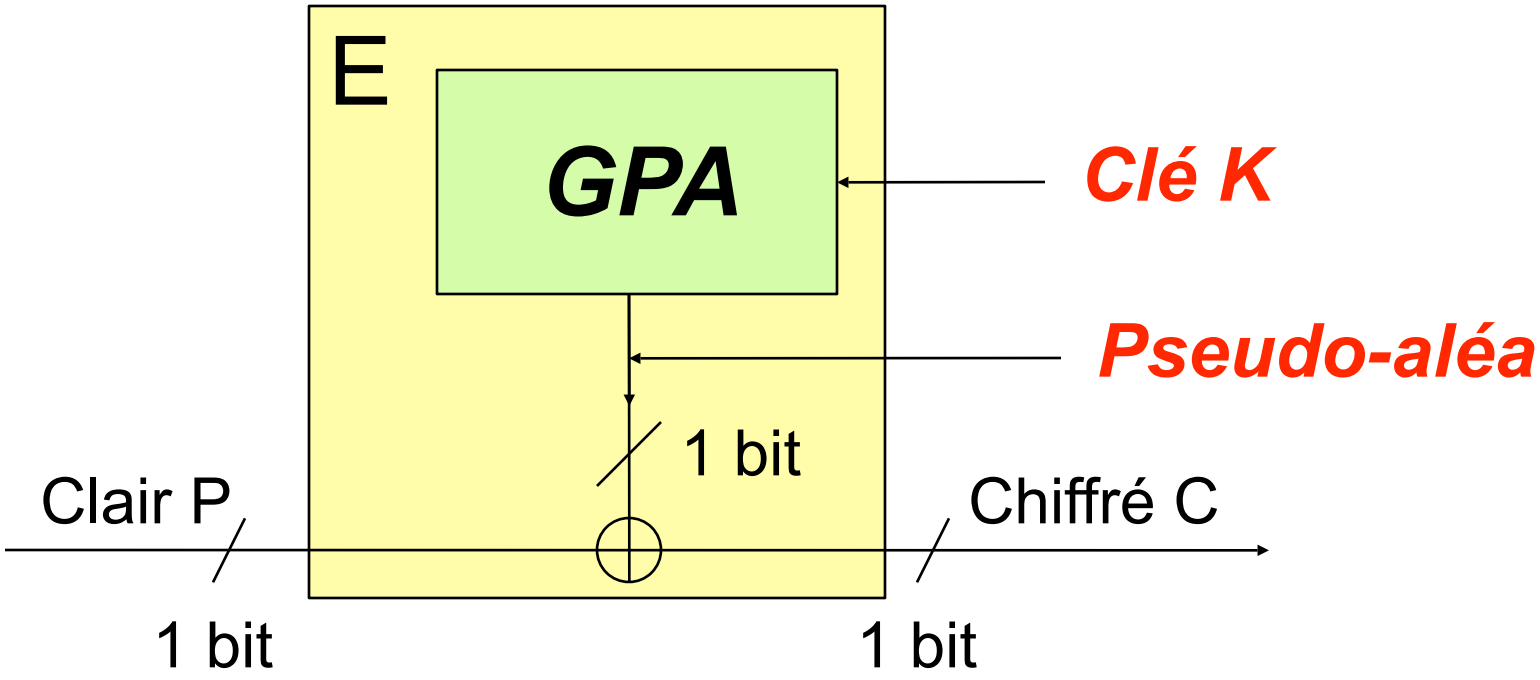


Chiffrement par flot

- le message est traité bit par bit
- E change en cours de chiffrement

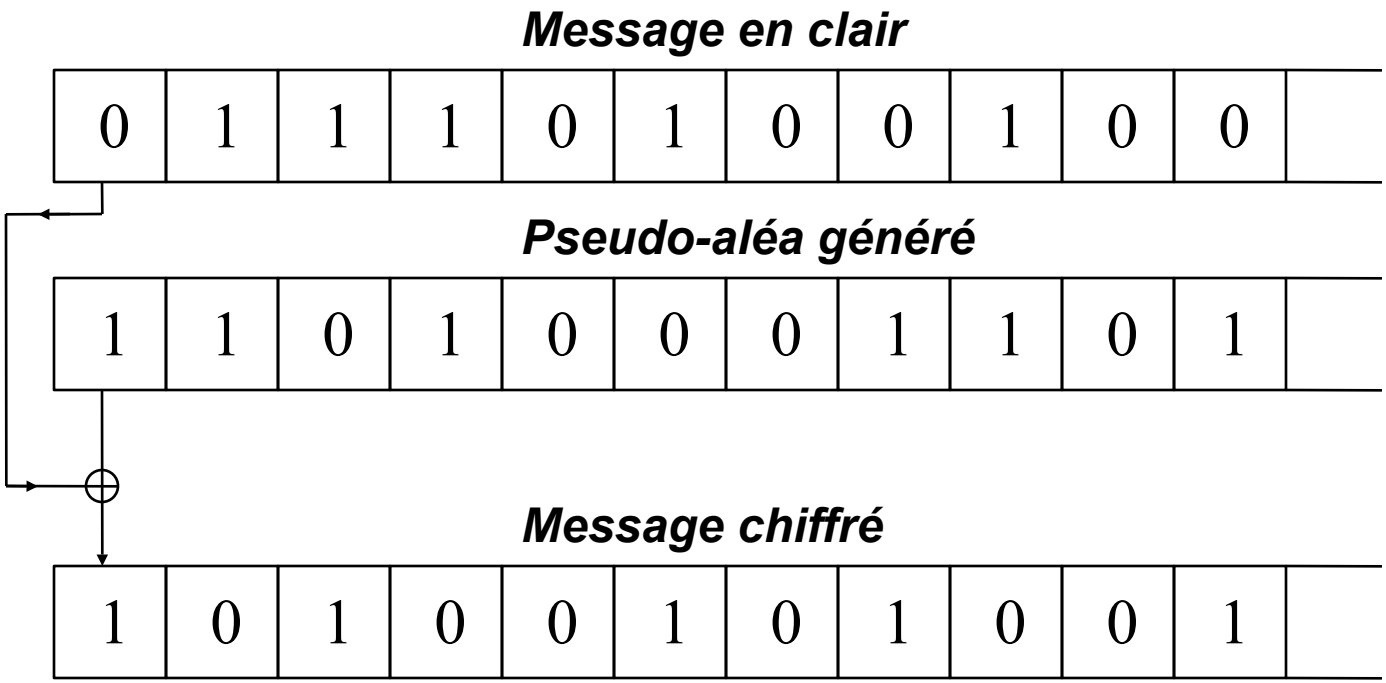


En général ...



Générateur Pseudo-Aléatoire (GPA)

GPA


Message en clair
0 1 1 1 0 1 0 0 1 0 0
Pseudo-aléa généré
1 1 0 1 0 0 0 1 1 0 1
Message chiffré
1 0 1 0 0 1 0 1 0 0 1

Idée sous-jacente

- Chiffrement de Vernam (masque jetable)
- Clé = pseudo-aléa
- Taille de clé = Taille du message
- Inconditionnellement sûr

Les algorithmes de chiffrement par flot s'en inspirent mais utilisent une suite pseudo-aléatoire générée à partir de quelques bits de clé réellement aléatoires

Sécurité parfaite

- Déf1: Sécurité parfaite. (Gen,Enc,Dec) sur un espace de clair M, est parfaitement sûr si pour tout $m \in M$, $c \in C$ tq $Pr(C=c)>0$,
- $Pr(M=m|C=c)=Pr(M=m)$
- ou, $Pr(C=c|M=m)=Pr(C=c)$
- Lemme: parfaitement sûr ssi $\forall M$,

$\forall m_0,m_1,\forall c$, $Pr(C=c|M=m_0)=Pr(C=c|M=m_1)$

One-Time Pad

- Th.: Le chiffrement de Vernam est parfaitement sûr
- Limitations:
  - Sécurité parfaite: $|K| \geq |M|$
  - Th. Shannon: Chiffrement tq $|K|=|M|=|C|$ est parfaitement sûr ssi
    - $\forall k \in K$, elle est choisie avec proba $1/|K|$
    - $\forall m \in M$, $\forall c \in C$, il existe une unique clé $k \in K$, $E_k(m)=c$

Pseudo-aléa

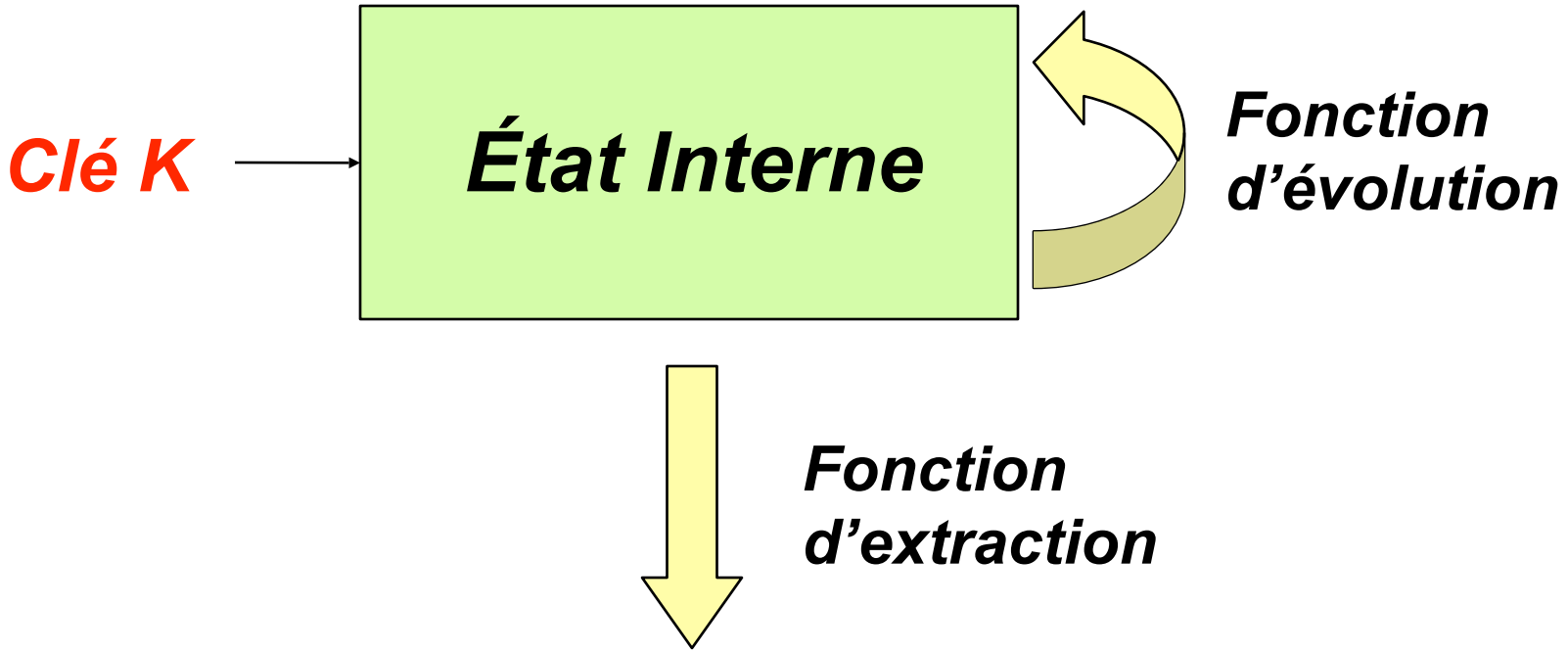
- Propriété attendue : impossible à distinguer de vrai aléa
- Plus précisément :
  - Équilibre : sortie = 0 ou 1 avec probabilité ½
  - Indépendance : chaque bit de sortie est indépendant des suivants et des précédents

Fabrication d'un GPA

Constitué de :

- État Interne suffisamment grand (recherche exhaustive)
- Fonction d'évolution
- Fonction d'extraction

Fabrication d'un GPA

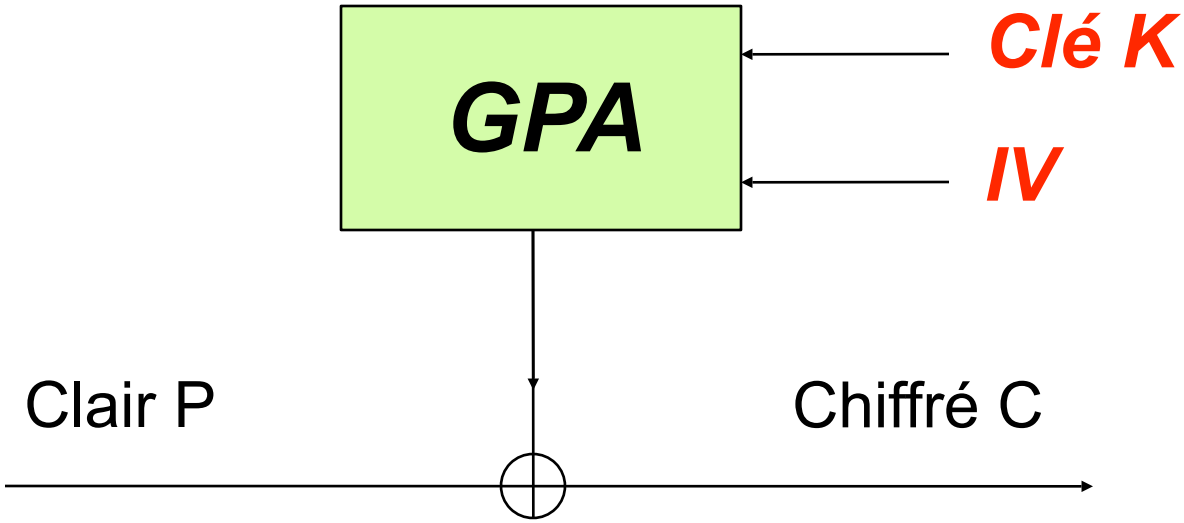


Vecteur d'initialisation

- En pratique, problème pour chiffrer plusieurs messages avec la même clé
  - Même clé ⇒ Même pseudo-aléa
  - $E_K(M1) \oplus E_K(M2) = M1 \oplus M2$
- Solution : entrée auxiliaire IV

  (Vecteur d'initialisation ou Marquant)

Vecteur d'initialisation



Génération de plusieurs séquences de sortie avec 1 seule clé

Tailles critiques

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

Caractéristiques

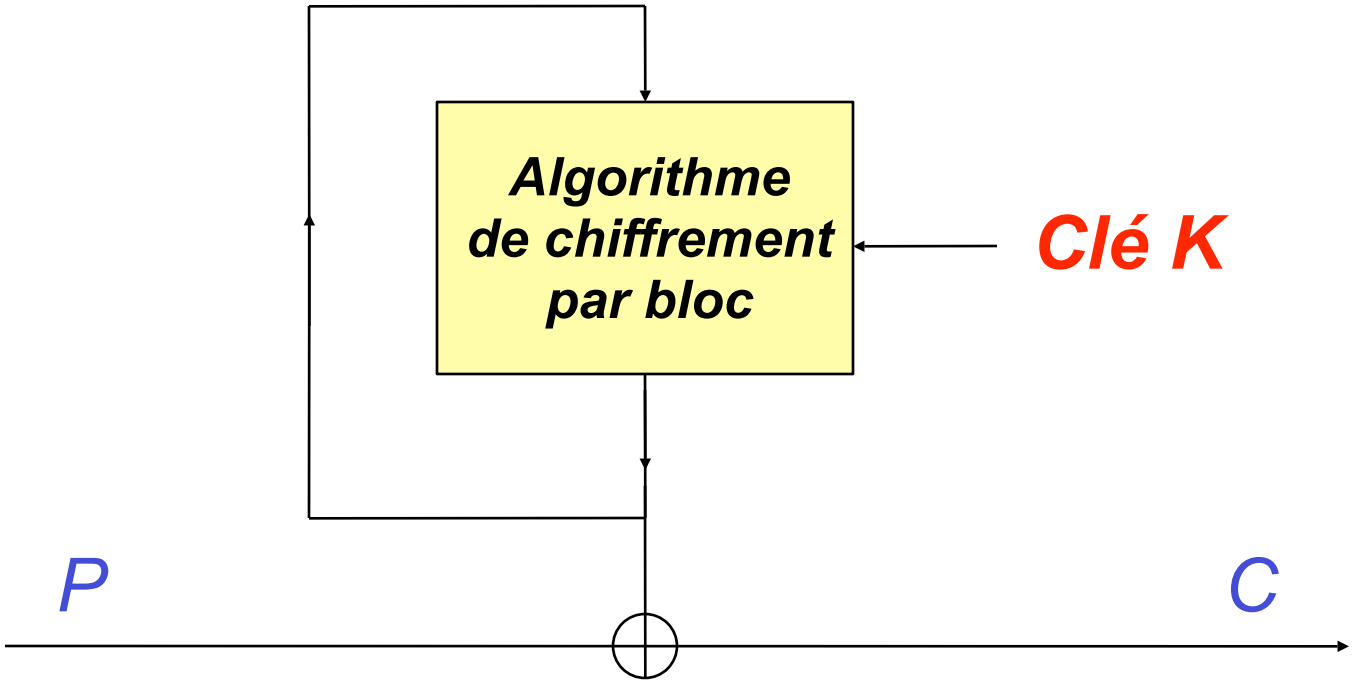
Algorithmes de chiffrement par flot :

- Rapidité (software - hardware)
- Peu de portes logiques (surface matérielle)
- Problèmes : synchronisation en cas d'erreur
  - Théorie : message arbitrairement longs
  - En pratique : trames
    (GSM : 224 bits, Bluetooth : 2700 bits)

Exemples d'algorithmes

- RC4 (norme 802.11)
- E0 (norme Bluetooth)
- A5/1 et A5/2 (norme GSM)
- Basé sur un algorithme de chiffrement par bloc

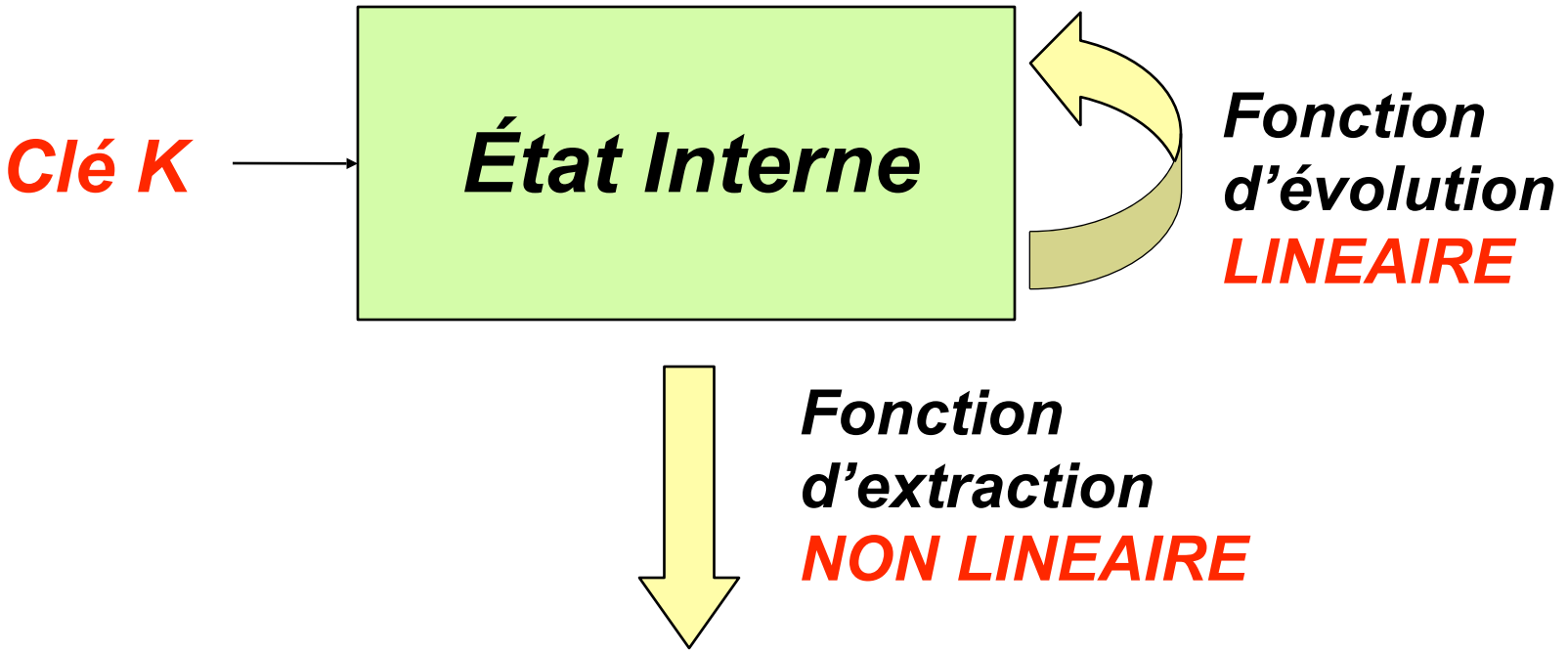
Le mode OFB



Revenons au GPA …

- Schéma général
  - Évolution linéaire (cela permet d'étudier des propriétés mathématiques comme la période)
  - Extraction non-linéaire («briser» la linéarité de la construction)
  - La clé K constitue l'état initial (même s'il arrive qu'elle soit réutilisée ensuite)

Schéma général



Critères

- Soit F la fonction d'évolution
  - F doit avoir une grande période (sinon les sorties vont se répéter)
- Soit G la fonction d'extraction
  - Doit résister à toutes les attaques connues :
  - Non-linéarité
  - Attaque par corrélations (approximation linéaire)
  - Attaques algébriques
  - Guess-and-determine

Non-linéarité

- Supposons que le pseudo-aléa s'exprime comme une fonction linéaire de l'état initial
- On est ramené à un problème d'algèbre linéaire

$$\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} = M \cdot \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_n \end{pmatrix}$$

Bits de pseudo-aléa

Clé $K = (k_1 \ldots k_n)$

Algorithme de Gauss → inverser la matrice M
complexité en n^3

Attaques par corrélation

- Idée générale :
  - L'algorithme est non-linéaire
  - Mais on le représente par des équations linéaires vraies avec probabilité

$$p = 0{,}5 + \varepsilon$$

  - Plus ε est grand, mieux c'est pour l'attaquant

Attaques algébriques

- Idée générale
  - L'algorithme est non-linéaire
  - Mais on le représente par des équations polynômiales

$$z_i = P(k_1 \ldots k_n) \text{ de degré } d$$

  - Plus d est petit, mieux c'est pour l'attaquant

Guess-and-determine

- Idée générale
  - L'algorithme est non-linéaire
  - Mais certains bits de sortie ne dépendent pas de toute la clé :

$$z_i = P(k_1 \ldots k_t) \text{ avec } t < n$$

  - Plus t est petit, mieux c'est pour l'attaquant

Remarque

- Cette liste d'attaque n'est pas exhaustive !
- Il existe de nombreux critères à satisfaire
- Certains critères sont contradictoires !

Période de F ?

- Peut-on construire des fonctions d'évolution telles que
  - Période

  et
  - Efficacité

  soient satisfaisants ?

Observation

- Période de F ?
- état interne de n bits ➔ période $\leq 2^n$
  - état initial $x_0 \in \{0,1\}^n$
  - itérés $x_i = F^i(x_0)$
- Il existe forcément $(a,b) < (2^n+1)$ t.q. $x_a = x_b$

  sinon 2^n+1 états différents de n bits !!!

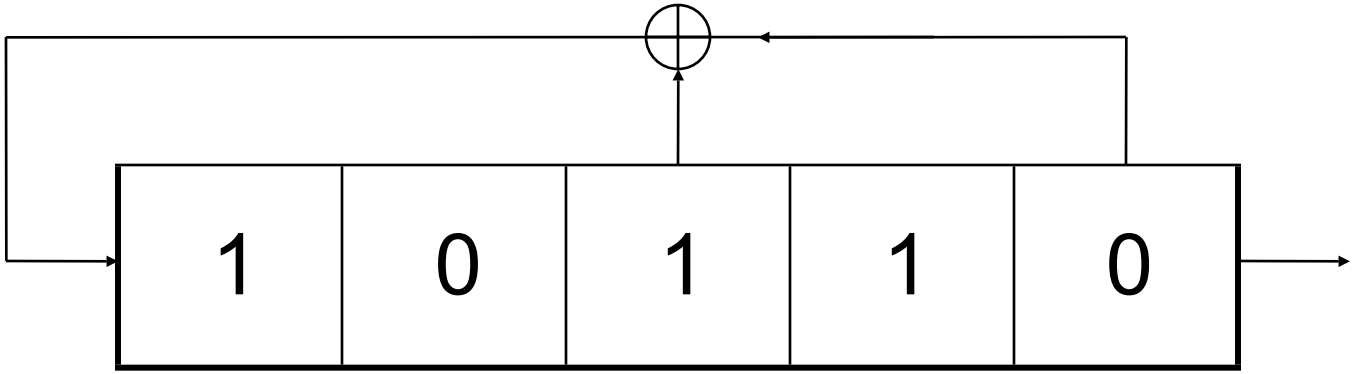
Registres à décalage

- Linear Feedback Shift Register (LFSR)
- Registre à Décalage à Rétroaction Linéaire (RDRL)
- Outil de base très souvent utilisé pour fabriquer des GPA
  - Propriétés mathématiques simples à étudier
  - Implémentation efficace en hardware

Les LFSR

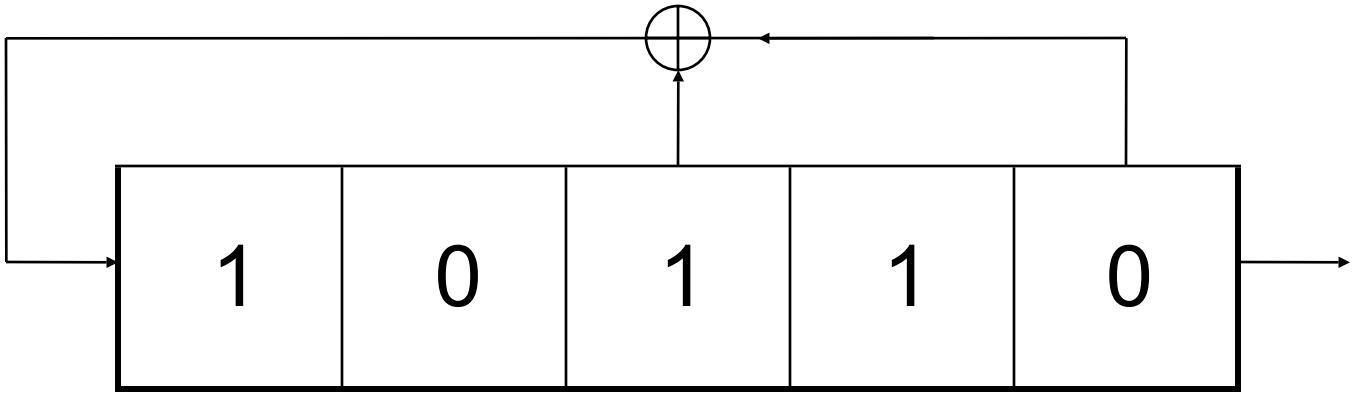
L cases mémoires contenant 1 bit d'information

Rebouclage linéaire

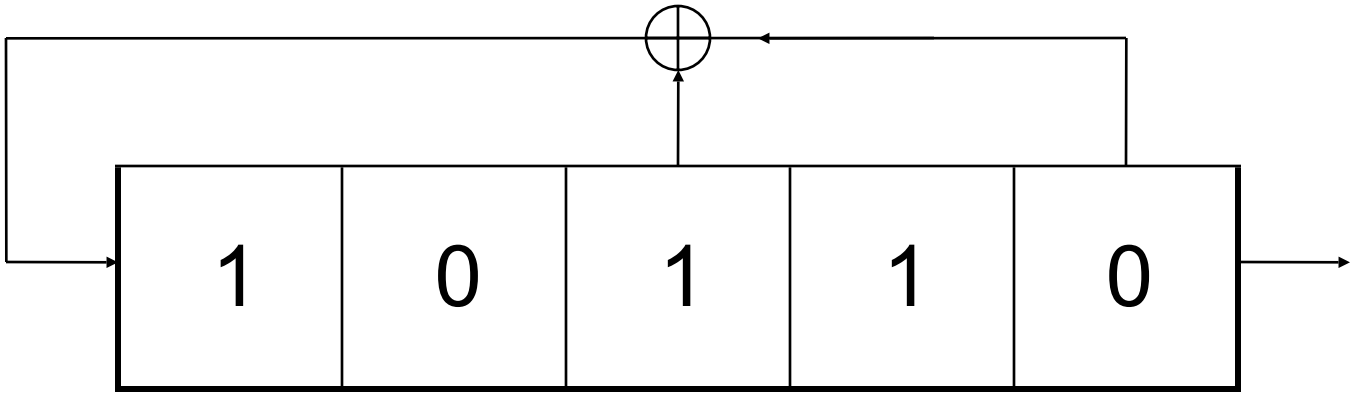


L = 5

Exemple

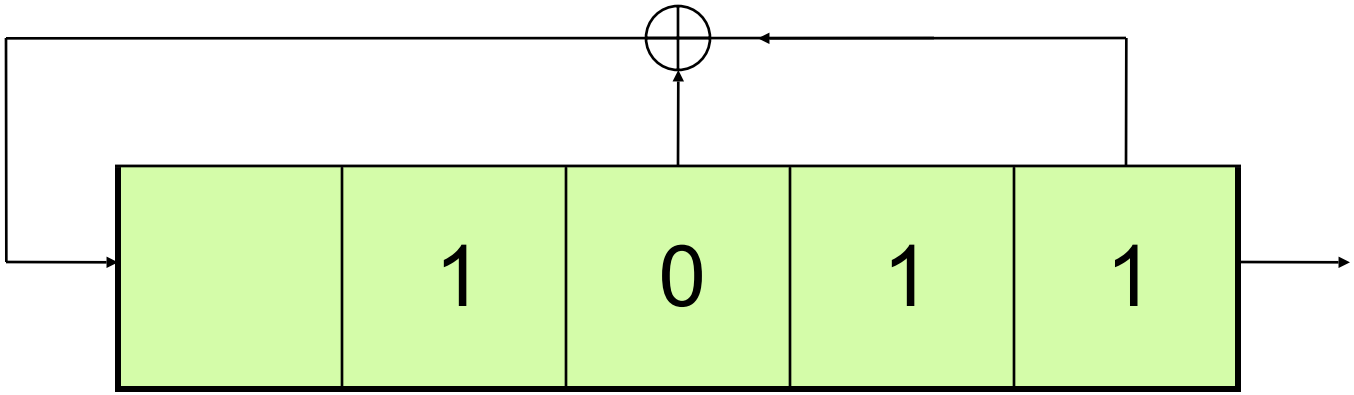


Exemple



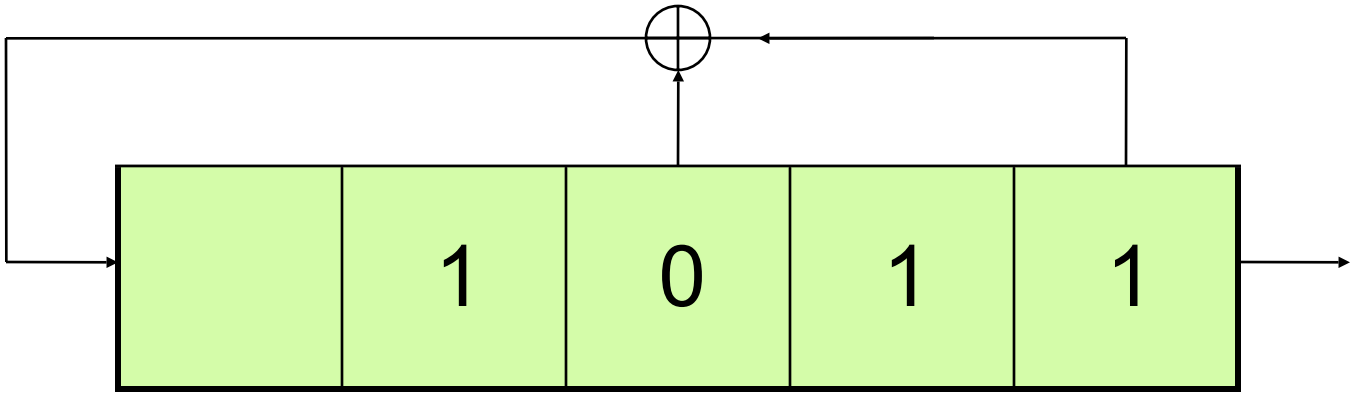
1 – Décalage (bit de sortie = 0)

Exemple



1 – Décalage (bit de sortie = 0)

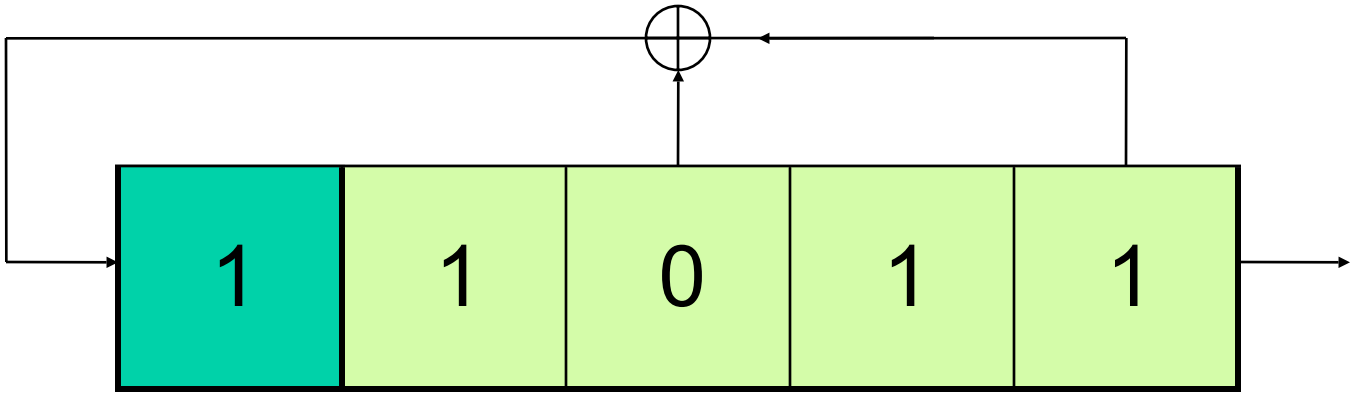
Exemple



1 – Décalage (bit de sortie = 0)

2 – Nouvelle valeur : 0 ⊕ 1 = 1

Exemple



1 – Décalage (bit de sortie = 0)

2 – Nouvelle valeur : 0 ⊕ 1 = 1

Propriétés

- La fonction de rebouclage s'écrit :

$$X(t) = c_1 X(t-1) \oplus \ldots \oplus c_L X(t-L)$$

où $\{ X(i), i \geq 0 \}$ est la suite des bits de sortie du LFSR

- Le Polynôme de rebouclage associé :

$$P(X) = 1 + c_1 X + \ldots + c_L X^L$$

- Les propriétés du LFSR dépendent fortement de P

Propriétés

- L'état (0, …, 0) est un point fixe
- Impossible d'atteindre la meilleure période possible (2^L)
- Si P est primitif, alors un seul cycle de période maximale $2^L - 1$ (critère important pour éviter les collisions d'état interne)
- P polynôme dense $\Rightarrow$ efficacité

Les LFSR

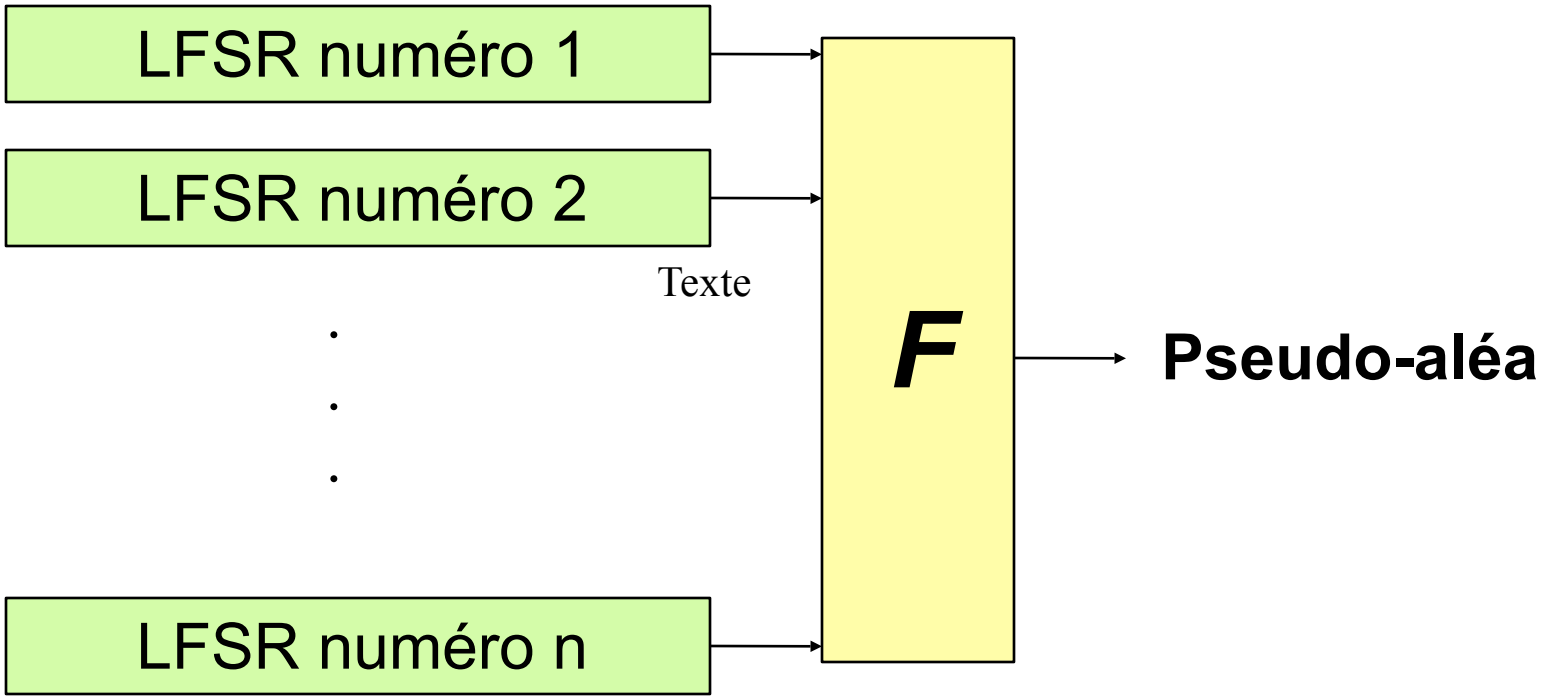
- Très utilisés pour construire des algorithmes de chiffrement par flot
- Comportement purement linéaire : ne pas utiliser seul !
- L'algorithme de Berlekamp-Massey permet de reconstruire P et l'état interne

Constructions

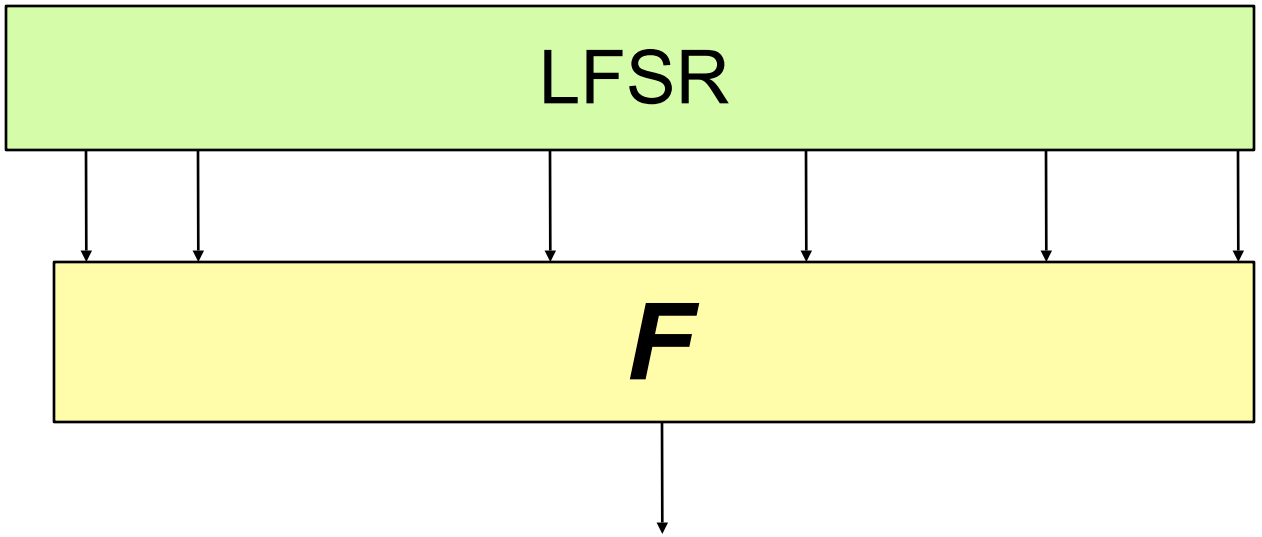
Avec des LFSR, 3 techniques :

- Combinaison
- Filtrage
- Contrôle de l'horloge

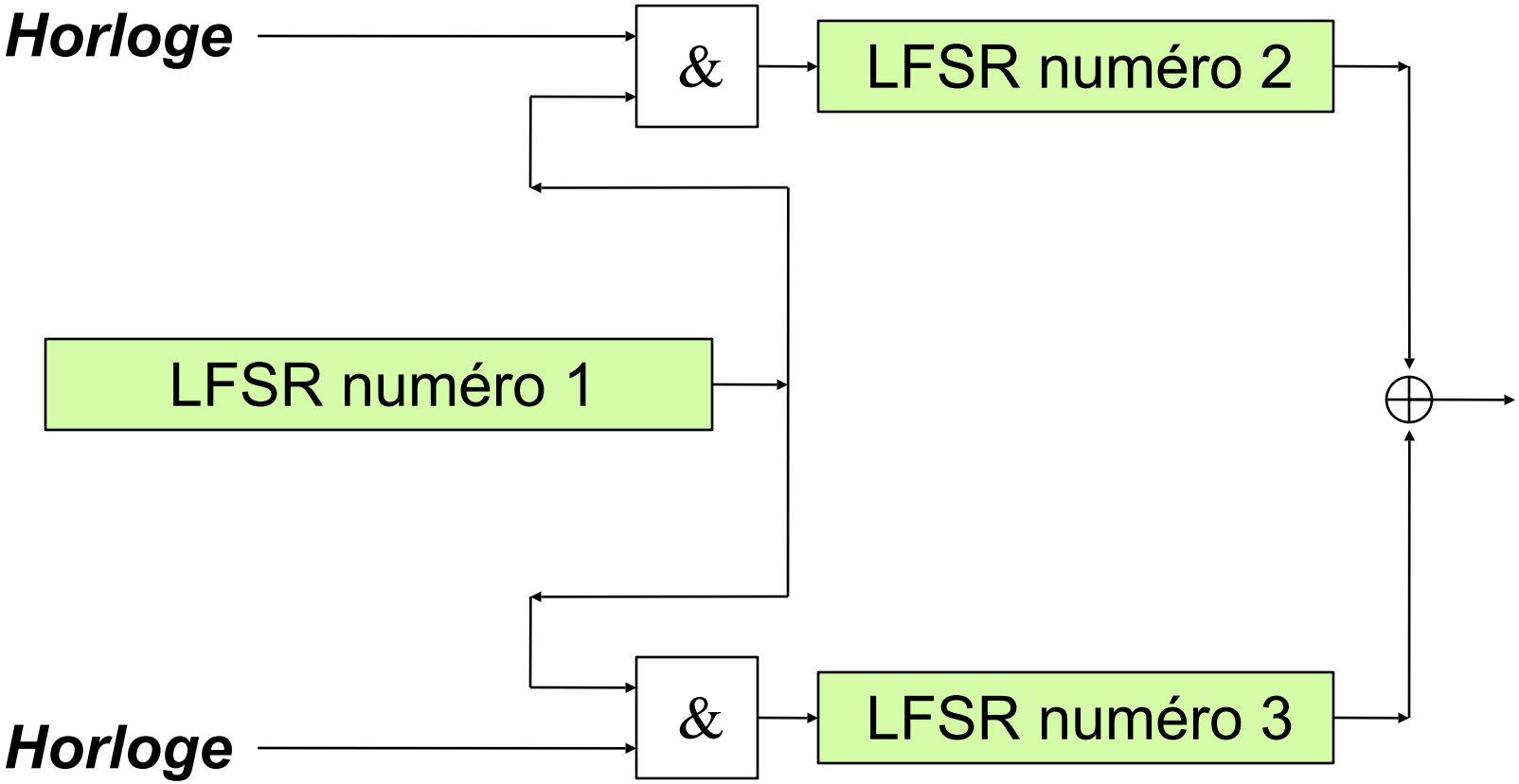
Combinaison



Filtrage



Contrôle de l'horloge



Synthèse LFSR

- Les registres à décalage sont utilisés historiquement dans les algorithmes
- Efficacité en hardware, beaucoup moins en software
- Problèmes de sécurité
  - Linéarité de la construction
  - Attaques par corrélation
  - Attaques algébriques

Synthèse Flot

- Problèmes liés au chiffrement par flot
  - Sécurité (pas de standard)
  - Traitement «par paquet» de l'information
  - Critères d'efficacité
- Le présent
  - Les algorithmes de chiffrement par flot sont progressivement remplacés par du chiffrement par bloc (ex : GSM)
  - Standardisation (NESSIE, eCRYPT, …)

Synthèse Flot (2)

- Le futur
  - Besoin d'algorithmes «hardware», très «légers» (puces RFID).
  - Besoin d'algorithmes «software», très rapides (routeurs IP).
  - Solution = chiffrement par flot ?
- Quels algorithmes existent aujourd'hui ?

Algorithmes actuels

- RC4 : utilisé dans de nombreux logiciels + norme 802.11
- A5/1 et A5/1 : norme GSM
- E0 : norme Bluetooth
- Algorithmes basés sur les LFSR
- Algorithmes avec contrôle de l'horloge
- Autres constructions plus récentes (Helix, SNOW, SEAL, SOBER, Rabbit, …)