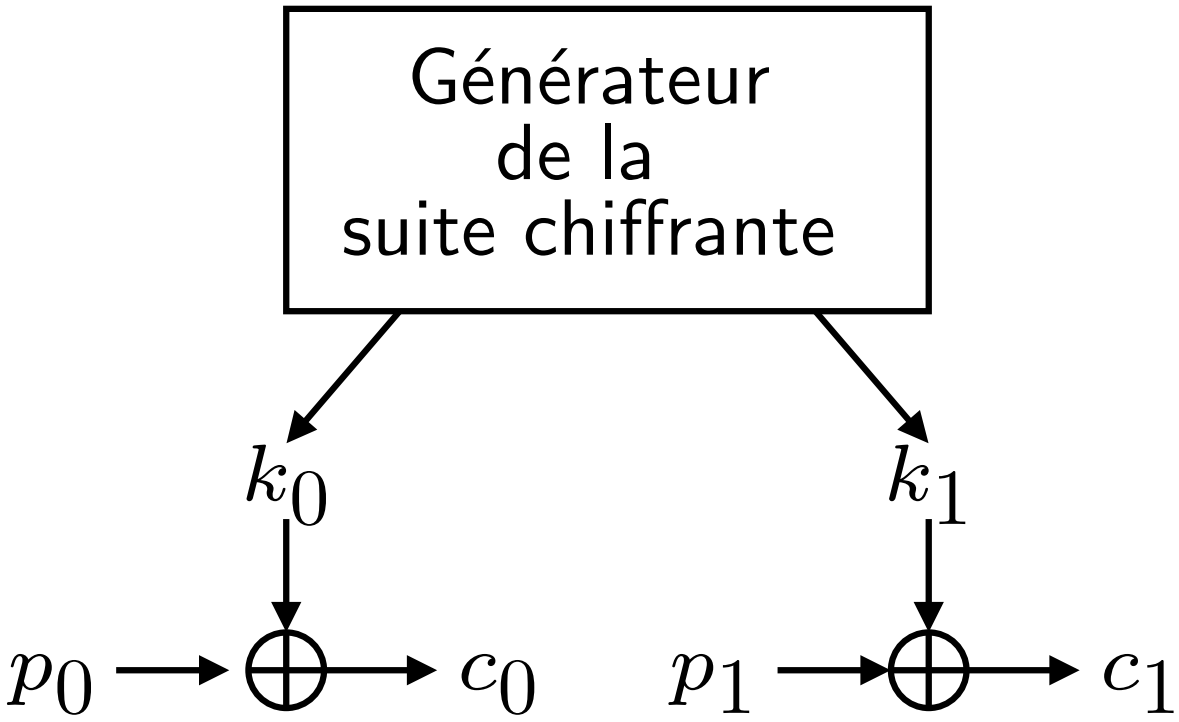
One-Time-Pad



One-Time-Pad

Sécurité parfaite

- $\ell(k) \geq \ell(p)$ implique
  - ▷ connaître $\ell(p)$ pour générer k,
  - ▷ ou communiquer k après p,
  - ▷ le canal de communication du message à la même bande passante que le canal de la clef.
- Il ne doit pas y avoir de répétition pour k :
  - ▷ on tire k au hasard
  - ▷ puis on teste pour éviter une répétition ($O(\log N)$).
- Le One-Time-Pad n'est utilisable en pratique !

Sécurité parfaite

- Quelque soit les moyens mis en œuvre par l'attaquant, il n'est pas possible de retrouver le texte clair ou la clef à partir du texte chiffré.

- **Conditions pour atteindre le secret parfait :**
  - $I(M;C) = 0$
  - $H(K) \geq H(M)$

- $H(K) \geq H(M)$ est la condition qui rend le système inutilisable.

- On va changer le modèle de sécurité !

De la sécurité parfaite. . .

à la sécurité calculatoire

- Un système de chiffrement est dit sûr au sens de la théorie de la complexité si la meilleure cryptanalyse nécessite n opérations avec n un nombre suffisamment grand pour que l'algorithme ne puisse être exécuté.
- On vient de limiter la puissance de l'attaquant.
- On parle de **sécurité calculatoire**.

Sécurité calculatoire

Précautions

- Il n'existe pas de système de chiffrement pour lequel on est une preuve mathématique que la meilleure cryptanalyse nécessite au moins n opérations.
- **Par contre**, il existe des systèmes pour lesquels la meilleure cryptanalyse **connue** est en n opérations
- En pratique, les concepteurs donne un $niveau\ de\ sécurité$ ($complexité\ de\ la\ cryptanalyse$). Toute personne qui trouve une cryptanalyse avec une complexité inférieure à celle annoncée **casse** l'algorithme.

Exemples

- GSM : A5/1
- IS95 : Long code generator
- CDMA and TDMA : Oryx
- Bluetooth : E0
- WEP : RC4
- DVDCSS : summation generator
- DVB : Common Scrambling Algorithm
- XBOX : RC4
- ECRYPT ESTREAM

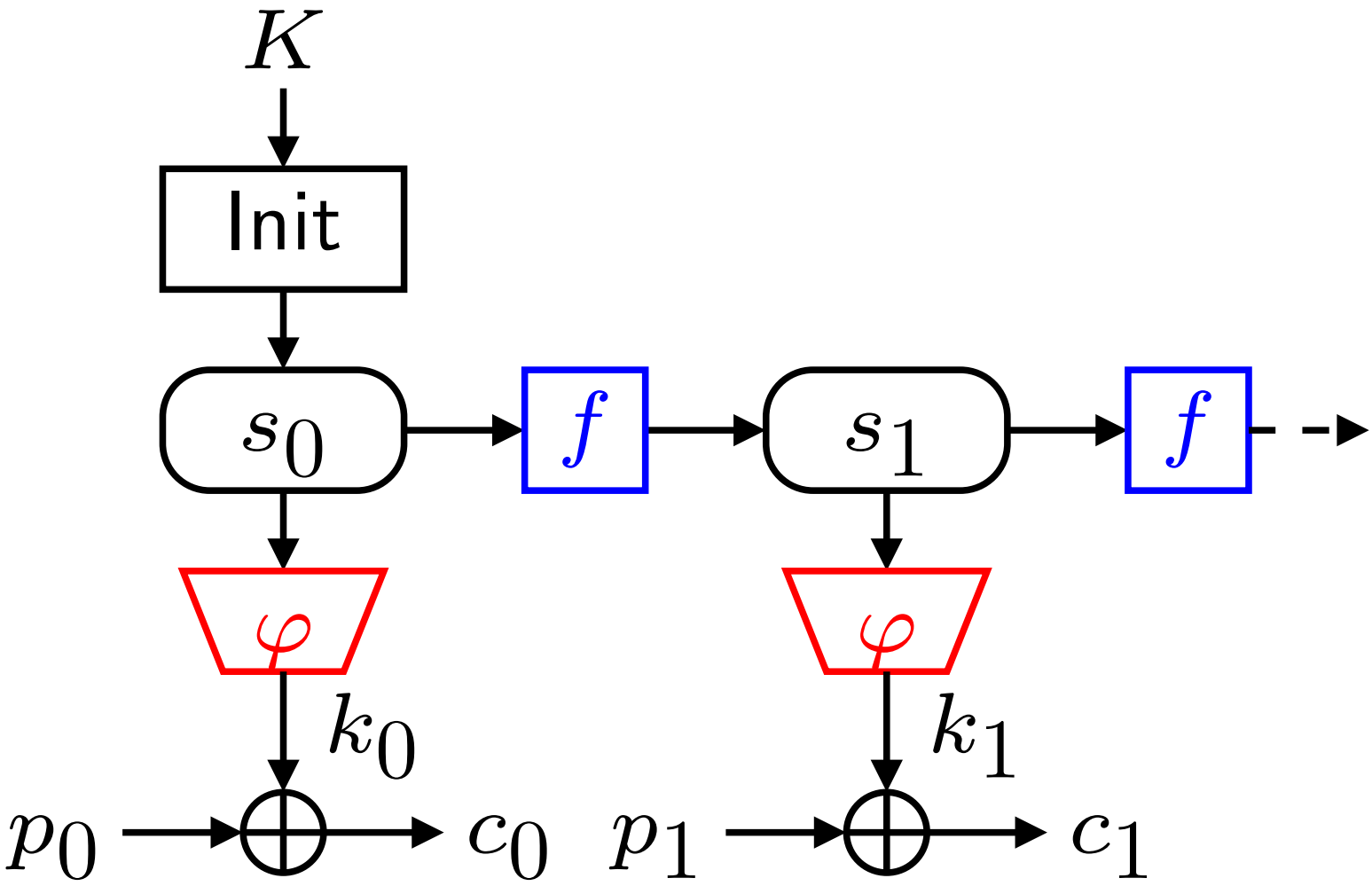
Vers le chiffrement par flot

- On veut :
  - ▷ une clef indépendante de la taille du message,
  - ▷ garder les bonnes propriétés du one-time-pad,
  - ▷ (dé)chiffrer vite (complexité en $O(1)$).

- On doit transformer une clef de taille fixe en une clef de longueur arbitraire pour (dé)chiffrer un message.

Comment faire ?

Générateur pseudo-aléatoire



Générateur pseudo-aléatoire

Vocabulaire

- Un générateur pseudo-aléatoire est une machine à états finis caractérisée par :
  - son état interne s (ℓ bits),
  - sa valeur s_0 est appelée la $graine$,
  - sa fonction de transition f,
  - sa fonction de filtrage φ (optionelle).

- Un générateur pseudo-aléatoire est déterministe : on a au maximum 2^ℓ états possibles.

Énumération

Fonction Booléenne

- Une fonction Booléenne est une fonction

$$g : \{0,1\}^{\ell} \to \{0,1\}.$$

- Une fonction Booléenne vectorielle est une fonction

$$h : \{0,1\}^{\ell} \to \{0,1\}^{m}.$$

- **Table de vérité** :

0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	1
0	1	0	1	1	1	0	0
0	1	1	1	1	1	1	0

Énumération

- On considère que la fonction de filtrage φ est une fonction Booléenne : $\varphi : \{0,1\}^\ell \to \{0,1\}$.
- *Combien existe-t-il de fonctions φ ?*

- On considère que la fonction de transition est une fonction Booléenne vectorielle $f : \{0,1\}^\ell \to \{0,1\}^\ell$.
- *Combien existe-t-il de fonctions f ?*
- *Que faut il en conclure ?*

Période

- Un générateur pseudo-aléatoire produit une suite infinie de valeurs.
- Une suite $(s_n)_{n\in\mathbb{N}}$ d'éléments d'un ensemble E est dite *périodique* s'il existe $p \in \mathbb{N}$ tels que $x_{n+p} = x_n$ pour tout n.
- Une suite $(s_n)_{n\in\mathbb{N}}$ d'éléments d'un ensemble E est dite *ultimement périodique* s'il existe $n_0 \in \mathbb{N}$ et $p \in \mathbb{N}$ tels que $x_{n+p} = x_n$ pour tout $n \geq n_0$.
- p est *une période* de cette suite.

Période

- La *période d'un générateur* est le plus petit entier positif $T > 0$ tel que :

$$s_{t+T} = s_t.$$

- Pour un état interne de n bits, on a $T \geq 2^\ell$.
- Si on utilise un générateur pseudo-aléatoire pour chiffrer des données on va avoir un problème à cause de la période.

Détection de cycle : Floyd (1)

Require: s_0

$\mu \leftarrow 1$

$y_0 \leftarrow s_0$

$s_1 \leftarrow f(s_0)$

$y_1 \leftarrow f(f(y_0))$

while $s_\mu \neq y_\mu$ **do**

$s_\mu \leftarrow f(s_{\mu-1})$ {Tortue}

$y_\mu \leftarrow f(f(y_{\mu-1}))$ {Lièvre}

$\mu \leftarrow \mu + 1$

end while

return μ

Détection de cycle : Floyd (2)

- Après cette étape, on a obtenu **un point sur le cycle** du générateur.
- De plus, μ est un multiple de T : $\mu = k \cdot T$.
- **Complexité :** $O(\mu)$.

Détection de cycle : Floyd (3)

- Pour trouver T, on cherche $\lambda \neq \mu$ tel que

$$s_\lambda = s_{2\lambda}.$$

On aura alors :

$$\begin{cases} \lambda \leq \quad \mu + T, \\ T = k \cdot (\lambda - \mu). \end{cases}$$

- On a donc $T = \lambda - \mu$.
- **Complexité :** $O(\lambda)$.

Détection de cycle : Floyd (4)

$\lambda \leftarrow 0$

$s_0 \leftarrow f(s_\mu)$

$y_0 \leftarrow s_0$

$s_1 \leftarrow f(s_0)$

$y_1 \leftarrow f(f(y_0))$

while $s_\lambda \neq y_\lambda$ **do**

$s_\lambda \leftarrow f(s_{\lambda-1})$ {Tortue}

$y_\lambda \leftarrow f(f(y_{\lambda-1}))$ {Lièvre}

$\lambda \leftarrow \lambda + 1$

end while

return $T \leftarrow \mu - \lambda$

Conclusion sur la période

- La complexité de l'algorithme de Floyd est $O(T)$.
- Il est donc impossible de choisir f au hasard si l'état interne est de taille conséquente ($\ell \geq 64$).
- On doit trouver des familles de fonctions f pour lesquels on peut prouver la période.

Générateur linéaire congruentiel

- Classe de générateurs proposée par Lehmer :

$$s_{t+1} = (a \cdot s_t) + c \bmod m$$

- Les premiers paramètres proposés étaient :
  — m premier,
  — $c = 0$.

- Il reste le choix de a.

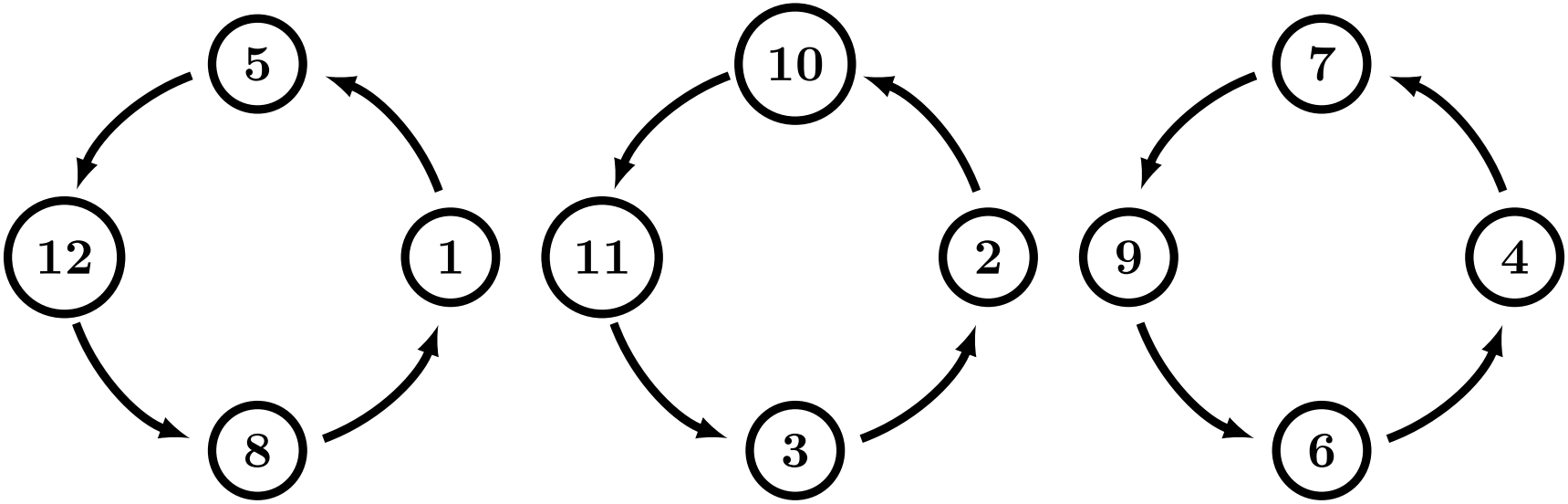
Point fixe

- $a = 0 \rightarrow \forall t,\ f(s_t) = 0 \rightarrow$ **point fixe**.
- $a = 1 \rightarrow \forall t,\ f(s_t) = s_0 \rightarrow$ **point fixe**.
- $a = m - 1$, alors on a :

$$s_1 = s_0 \cdot (m-1) \bmod m$$

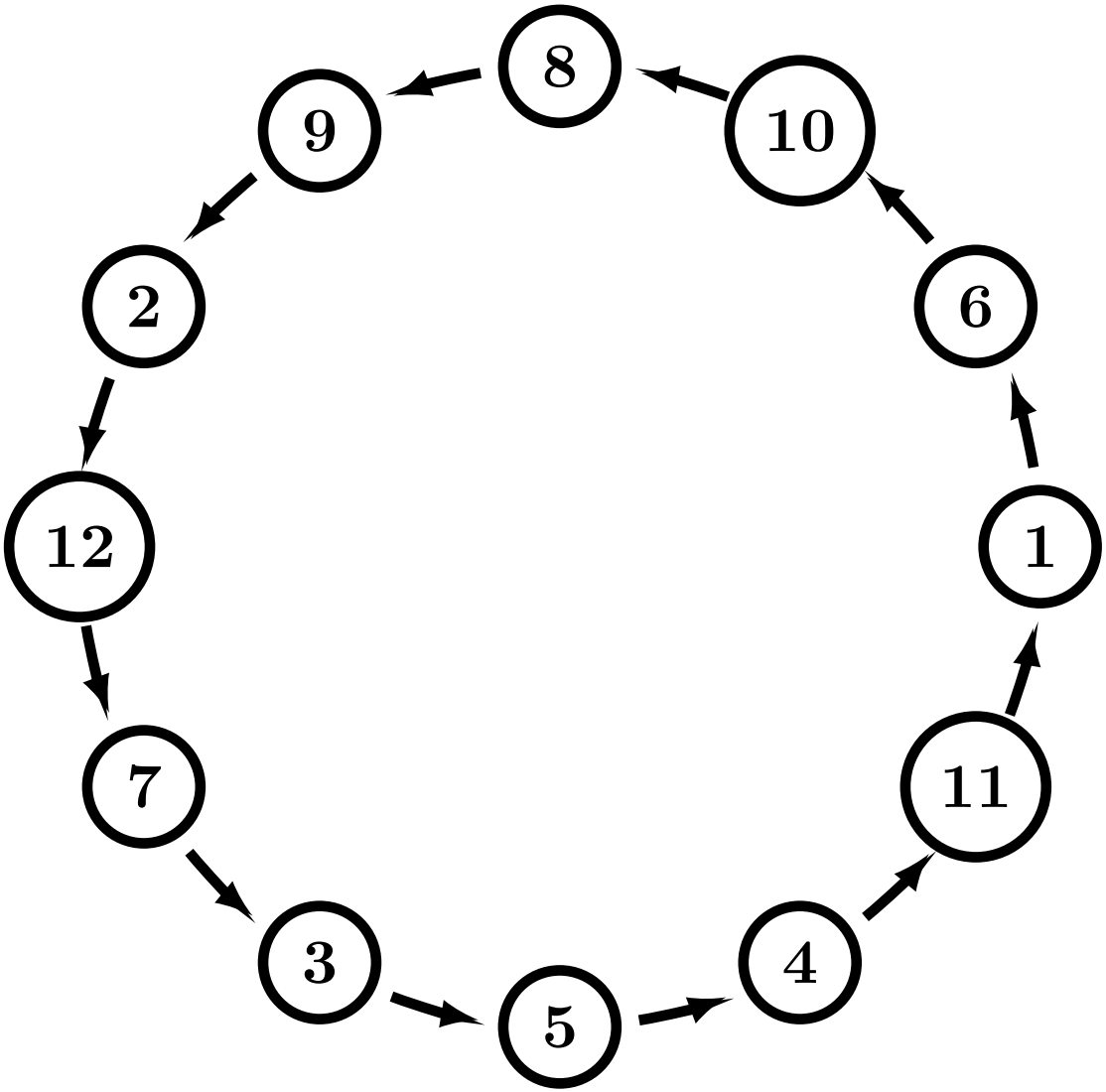
- $s_0 = 0$ est une $mauvaise\ graine$.

$$s_{t+1} = 5 \cdot s_t \bmod 13$$



- On a **3 cycles de longueur 4.**

$s_{t+1} = 6 \cdot s_t \bmod 13,$ **and** $(x_0 = 1)$



Générateur de Park-Miller

- Park et Miller ont proposé le générateur linéaire congruentiel suivant :

$$x_{t+1} = 16807 \times x_t \bmod 2^{31} - 1.$$

- Encore utilisé dans :
  — bibliothèque *Apple Carbon*,
  — Contiki.

Implantation

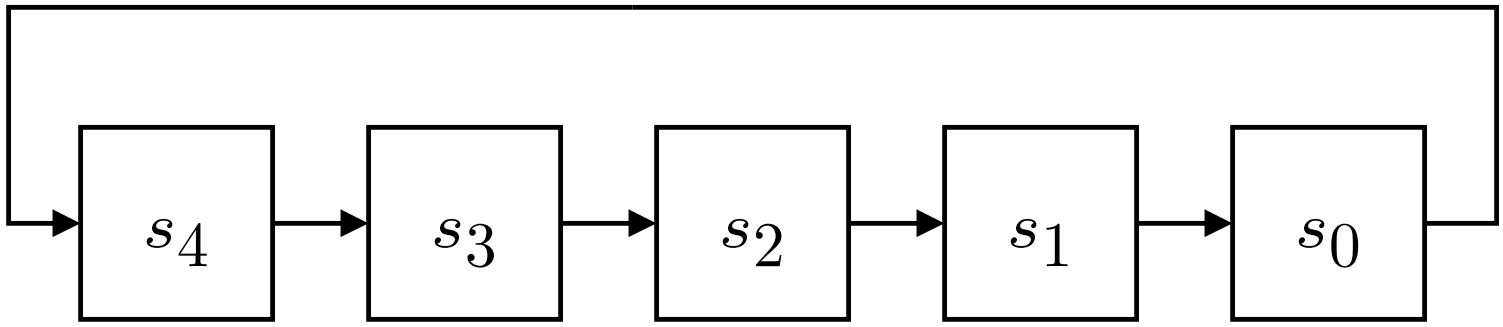
- Il faut faire attention :

$$x_{t+1} = 16807 \times x_t \bmod 2^{31} - 1.$$

- **Débordements**. Quand on multiplie 2 valeurs de m bits, on obtient un résultat sur $2m$ bits !

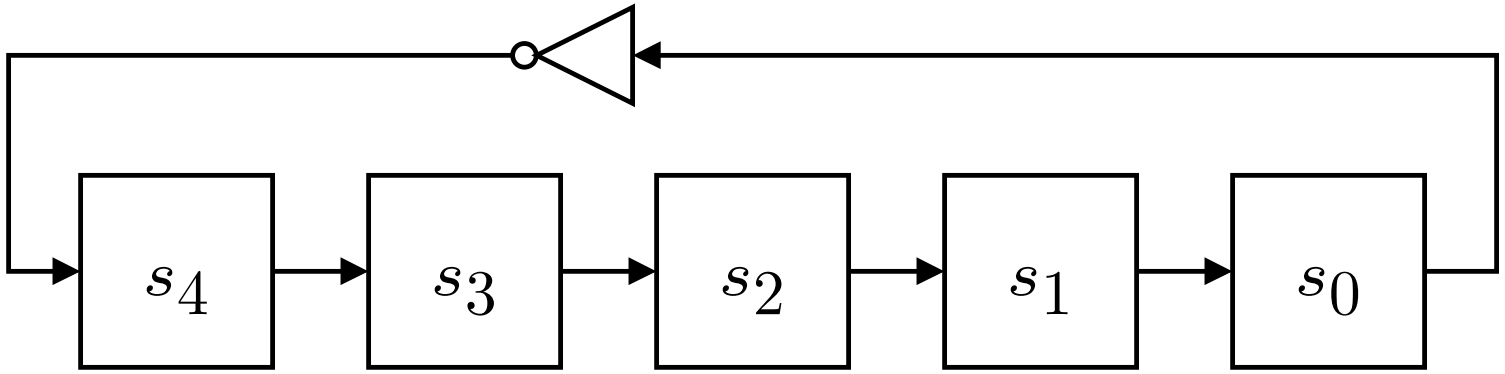
$$x_{t+1} = 16807 \times x_t \bmod 2^{31} - 1$$
$$\neq$$
$$x_{t+1} = (16807 \times x_t \bmod 2^k) \bmod 2^{31} - 1$$
$$\text{if } k \leq 46$$

Registre en anneau



	s_4	s_3	s_2	s_1	s_0
0	0	0	0	0	1
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0

Compteur de Johnson



	s_0	s_1	s_2	s_3	s_4
0	0	0	0	0	0
1					
2					
3					
4					

	s_0	s_1	s_2	s_3	s_4
5					
6					
7					
8					
9					

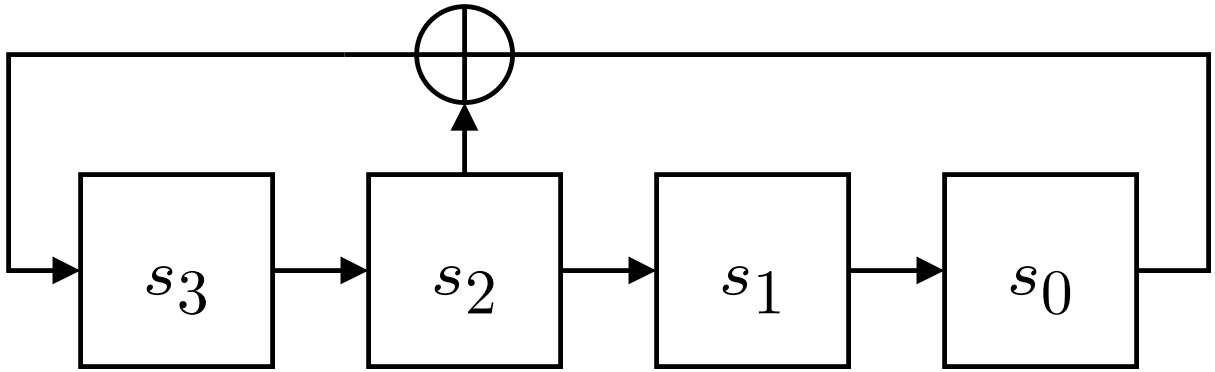
Théorème de Golomb

- Le(s) cycle(s) du générateur sera (seront) sans point de branchement ssi on peut écrire :

$$f(s_m, s_{m-1}, \cdots, s_{m-\ell+1}) = f_1(s_{m-1}, \cdots, s_{m-\ell+1}) \oplus s_m.$$

- On vient de passer de 2^{2^n} à $2^{2^{n-1}}$ fonctions de rétroaction possible.

Registre à rétroaction linéaire



	s_3	s_2	s_1	s_0
0	0	0	0	1
1				
2				
3				
4				
5				
6				

Registre à rétroaction linéaire

- On note : $(s_0, \cdots, s_{\ell-1})$ l'état initial du registre.
- $(a_1, \cdots, a_\ell)$ sont les coefficients de la rétroaction.
- $a_i \in \mathbb{F}_2$ et $s_i \in \mathbb{F}_2$ pour $i \in [1, n]$
- La fonction de transition f est définie par :

$$s_{m+\ell} = a_1 s_{m+\ell-1} \oplus a_2 s_{m+\ell-2} \oplus \cdots \oplus a_\ell s_m$$

Polynôme de rétroaction

- On appelle polynôme de rétroaction du registre, le polynôme définit par :

$$p(X) = 1 + \sum_{i=1}^{\ell} a_i X^i \ \in \mathbb{F}_2[X].$$

- Soit $(s_n)_{n\geq 0}$ la suite binaire ultimement periodique generer par notre registre. On associe a cette suite la serie formelle :

$$s(X) = \sum_{n\geq 0} s_n X^n \ \in \mathbb{F}_2[X]$$

Polynôme de rétroaction

- On peut écrire $s(X)$ tel que :

$$s(X) = \frac{g(X)}{p(X)},$$

avec $g(X) \in \mathbb{F}_2[X]$ et deg $g <$ deg p

Preuve

- On a $g(X) = s(X) \cdot p(X)$ d'où :

$$g(X) = \left(\sum_{n \geq 0} s_n X^n \right) \cdot \left(1 + \sum_{i=1}^{\ell} a_i X^i \right)$$

$$= \sum_{i=0}^{\infty} \left(\sum_{j=0}^{i} a_{i-j} s_j \right) X^i$$

On a $\sum_{j=0}^{i} a_{i-j} s_j = 0$ pour $i \geq \ell$, on obtient $g(X)$:

$$g(X) = \sum_{i=0}^{\ell-1} \left(\sum_{j=0}^{i} a_{i-j} s_j \right) X^i$$

Polynôme de rétroaction minimal

- Le polynôme de rétroaction minimal de la suite $(s_n)_{n\geq 0}$ est le polynôme de rétroaction de *plus bas degré* parmi les polynômes de rétroaction de tous les registres pouvant générer la suite $(s_n)_{n\geq 0}$.

- La complexité linéaire d'une suite est le *degré de son polynôme de rétroaction minimal*.

Polynôme de rétroaction minimal

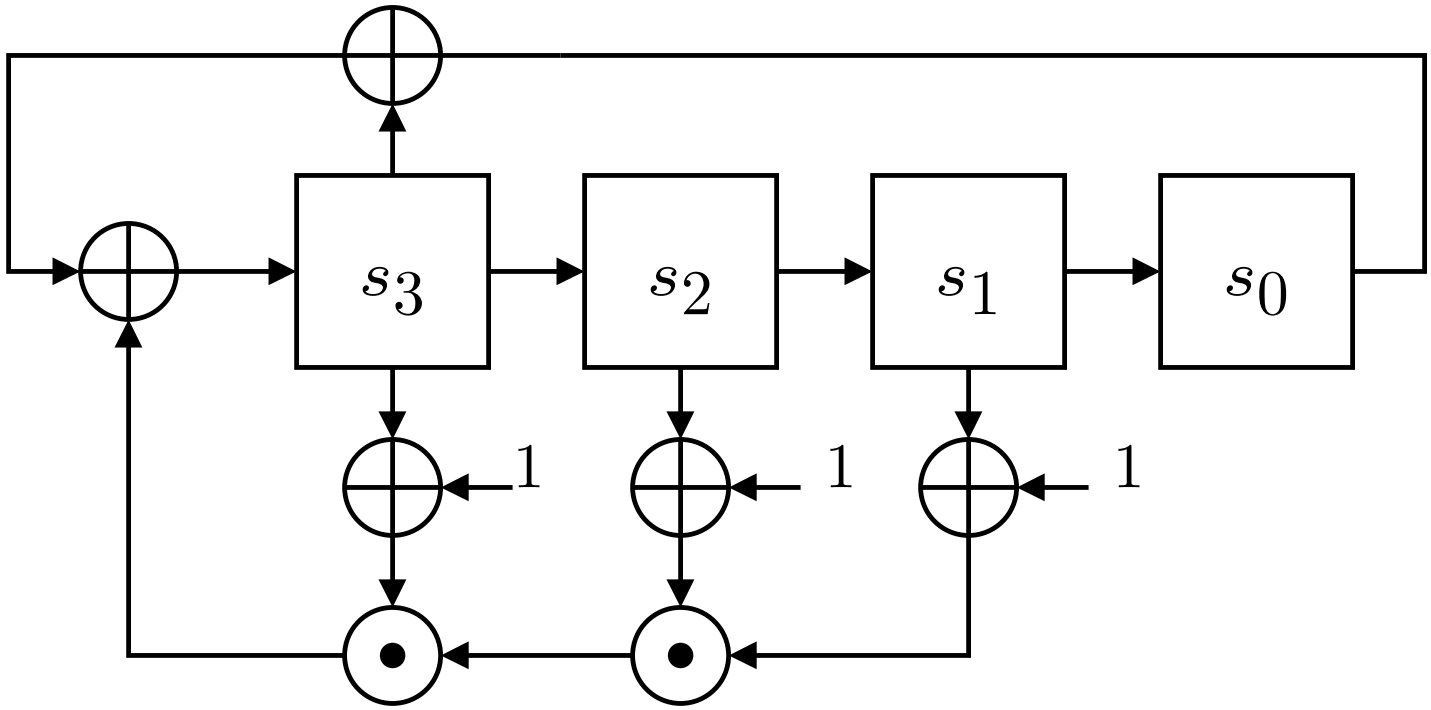
- Un polynôme $p(X)$ de degré $\ell > 0$ est dit *irréductible* si tout diviseur de p est une constante ou est produit de p par une constante.

- Un polynôme $p(X)$ de degré $\ell > 0$ défini sur $\mathbb{F}_2$ est *primitif* s'il est *irréductible* et

$$\min \{i > 0 | X^1 = 1 \bmod p\} = 2^\ell - 1.$$

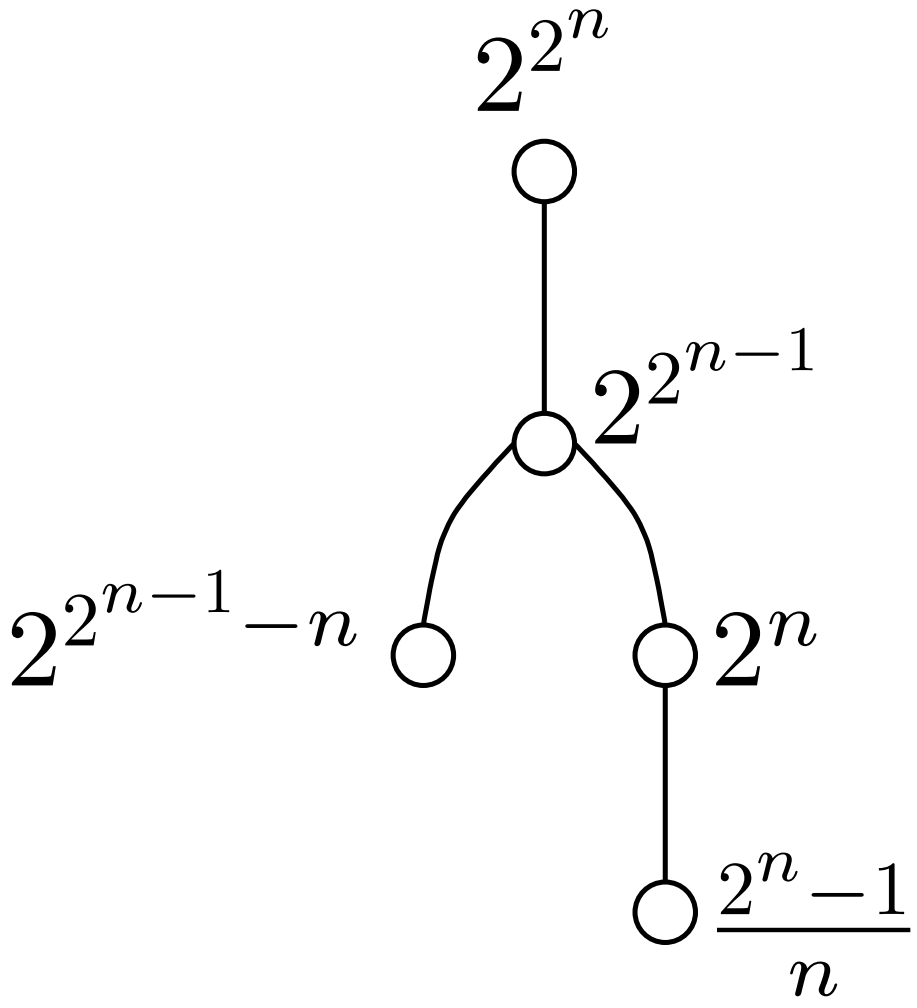
- Un registre à rétroaction linéaire de longueur ℓ a une période de $2^\ell - 1$ ssi son polynôme de rétroaction est primitif et son état initiale est non-nul.

Séquence de de Brujin



- Que pensez vous de ce registre à rétroaction non-linéaire ?

Séquence



Vision matricielle

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 \\ a_\ell & a_{\ell-1} & \cdots & a_3 & a_2 & a_1 \end{pmatrix}$$

- En considérant les états successifs R_i du registre comme des vecteurs colonnes, on a

$$R_{i+1} = AR_i.$$

Vision matricielle

- On remarque que l'on a :

$$R_{i+n} = A^n R_i.$$

- Le polynôme caractéristique de A est défini par :

$$\tilde{p}(X) = \det(Id - AX).$$

- De plus, on a : $\tilde{p}(X) = X^n p(X^{-1})$.

Propriétés statistiques

- Démontrées par Golomb en 1982 :
  — la séquence est équilibrée ;
  — les séries sont équitables réparties ;
  — la fonction d'autocorrélation prend 2 valeurs.

- Attention ces propriétés ne garantissent aucunement la sécurité de la séquence !

Equilibre

- Dans chaque période, le nombre de 0 est approximativement égal au nombre de 1 :

$$\left| \sum_{i=0}^{T-1} (-1)^{s_i} \leq 1 \right| .$$

- Dans une suite de longueur maximale $2^\ell - 1$, toute suite $(s_i, s_{i+1}, \cdots, s_{i+\ell-1})$ de n éléments non tous nuls apparaît **une et une seule fois par période**.

Equilibre

- **La démonstration est simple :** un registre donné $R_i = (s_i, s_{i+1}, \cdots, s_{i+\ell-1})$ ne peut apparaître qu'une fois par période. Or il y a $2^\ell - 1$ états par période et les registres prennent au plus $2^\ell - 1$ valeurs. Donc ils prennent toutes les valeurs une fois.

- **Question :** combien y a t'il d'états ayant la valeur s_i à 1. Même question pour 0. Conclusion ?

Série

- Une série (de 0 ou de 1) est une succession de bits identiques, maximale (i.e. encadrée par des bits opposés).

- Soit S l'ensemble des séries. Dans chaque période, si $2^k \leq |S| < 2^{k+1}$, on trouve $|S|/2$ séries de longueur 1, $|S|/4$ séries de longueur 2, . . . , $|S|/2^k$ séries de longueur k, et pour chaque longueur, autant de séries de 0 que de séries de 1.

Série

- **Démonstration.** Comptons le nombre d'occurence d'une série d'exactement k zéros. Cela revient à compter les occurences de $(1, \underbrace{0, \cdots, 0}_{k}, 1)$. Comme tous les états $R_i = (s_i, s_{i+1}, \cdots, s_{i+\ell-1})$ apparaissent une et une seule fois, un registre composé de

$$(1, \underbrace{0, \cdots, 0}_{k}, 1, s_{i+k+2} \cdots, s_{i+\ell-1}),$$

apparaît $2^{\ell-k-2}$.

Cryptanalyse

- Les **attaques par recouvrement de clef**, qui visent à **retrouver la clef secrète** du générateur à partir de la connaissance d'un certain nombre de bits de sa sortie.
- Attaque la plus puissante !

Cryptanalyse

- Les **attaques par recouvrement de l'état initial**, dont l'objectif est de retrouver l'initialisation du générateur (ou de façon équivalente un état interne complet).
- La connaissance de la clef secrète suffit naturellement pour retrouver l'initialisation, mais la réciproque n'est pas nécessairement vraie.

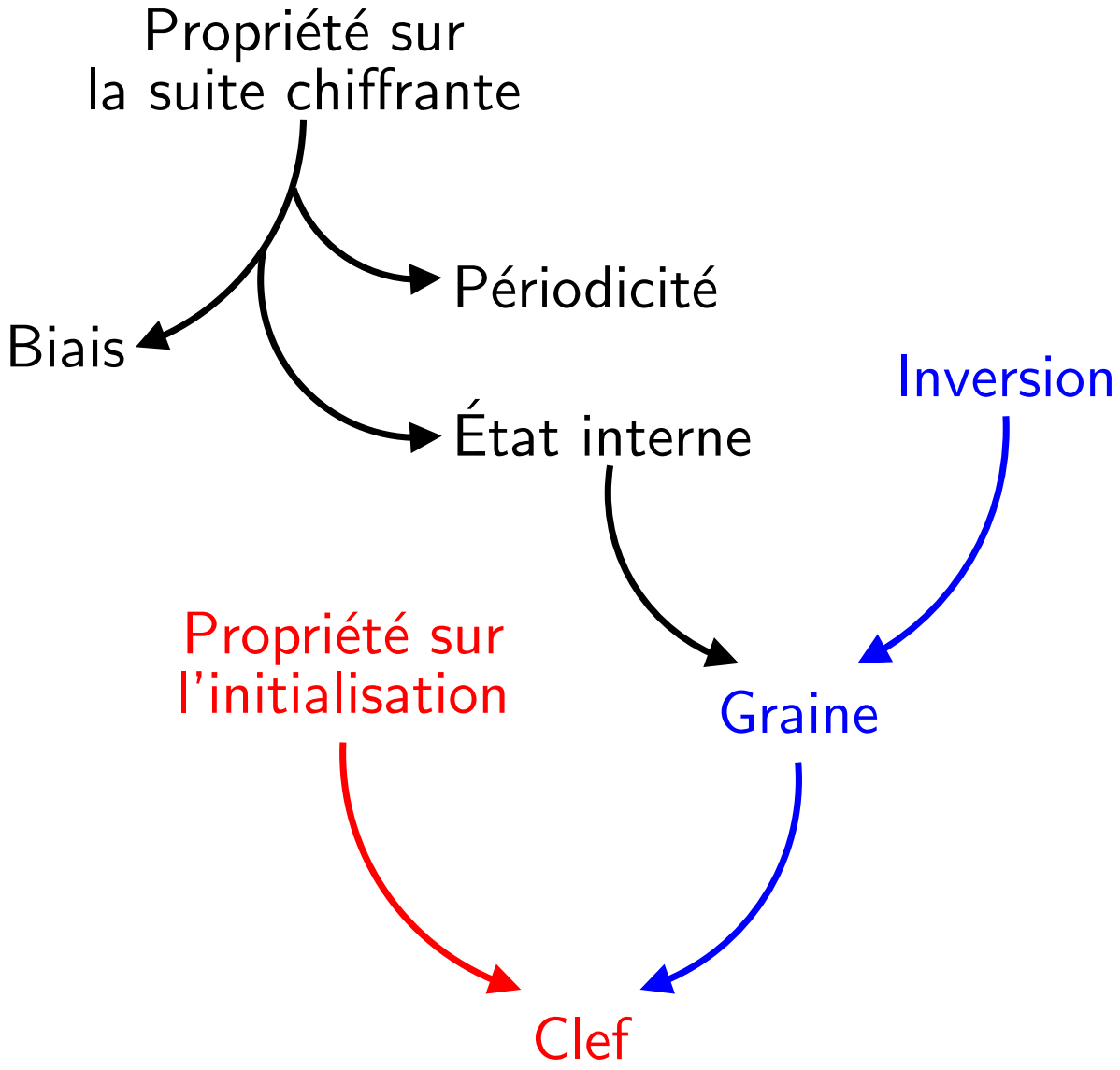
Cryptanalyse

- Les **attaques par prédiction du bit suivant** consistent à partir de la connaissance des n premiers bits de la suite engendrée par le générateur pour une certaine clef, à prédire la valeur du bit suivant

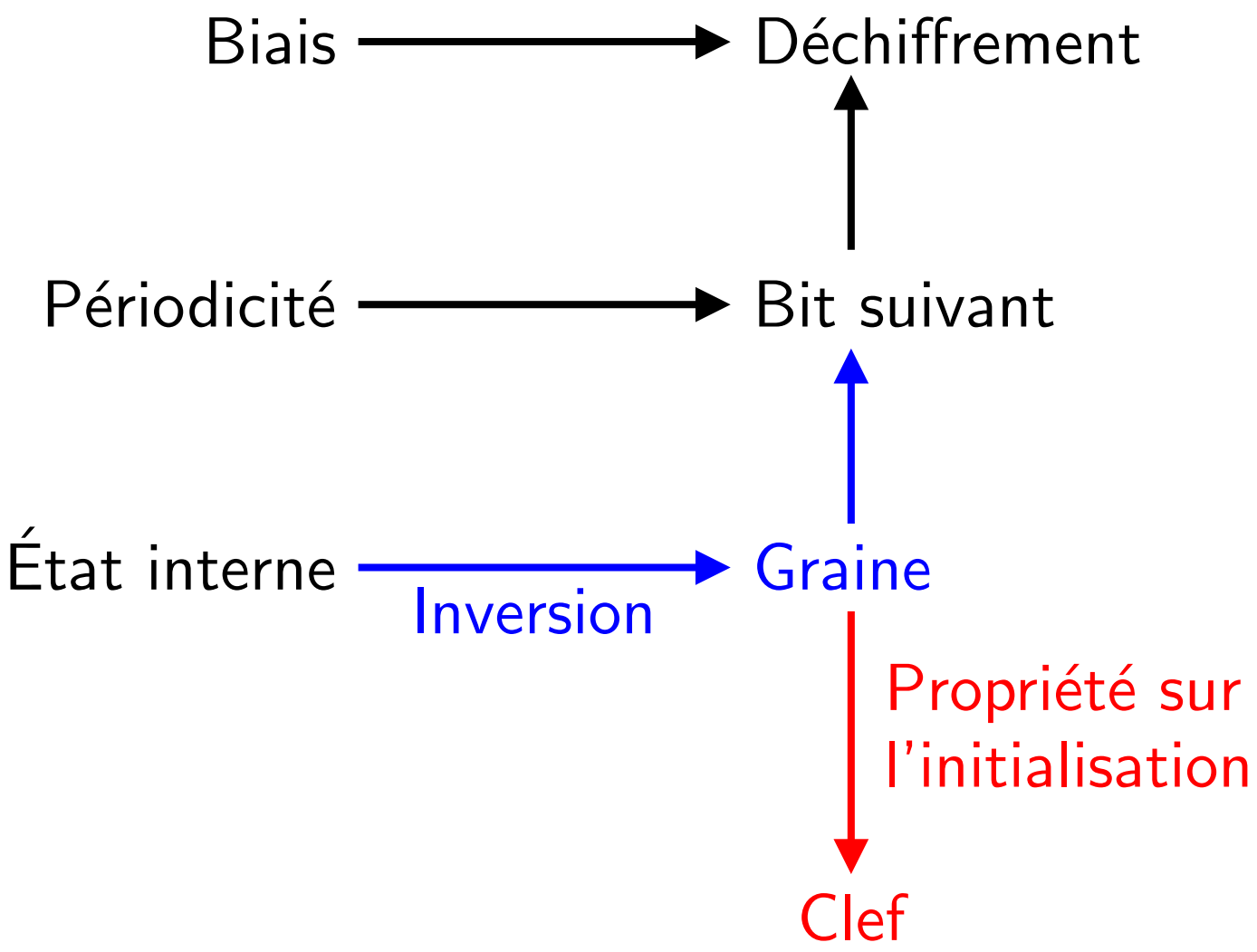
Cryptanalyse

- Les **attaques par distingueur** déterminent si une suite de n bits correspond à la sortie du générateur pseudo-aléatoire considéré ou s'il s'agit d'une suite véritablement aléatoire.

Attaques sur les chiffrements par flot



Impact des attaques



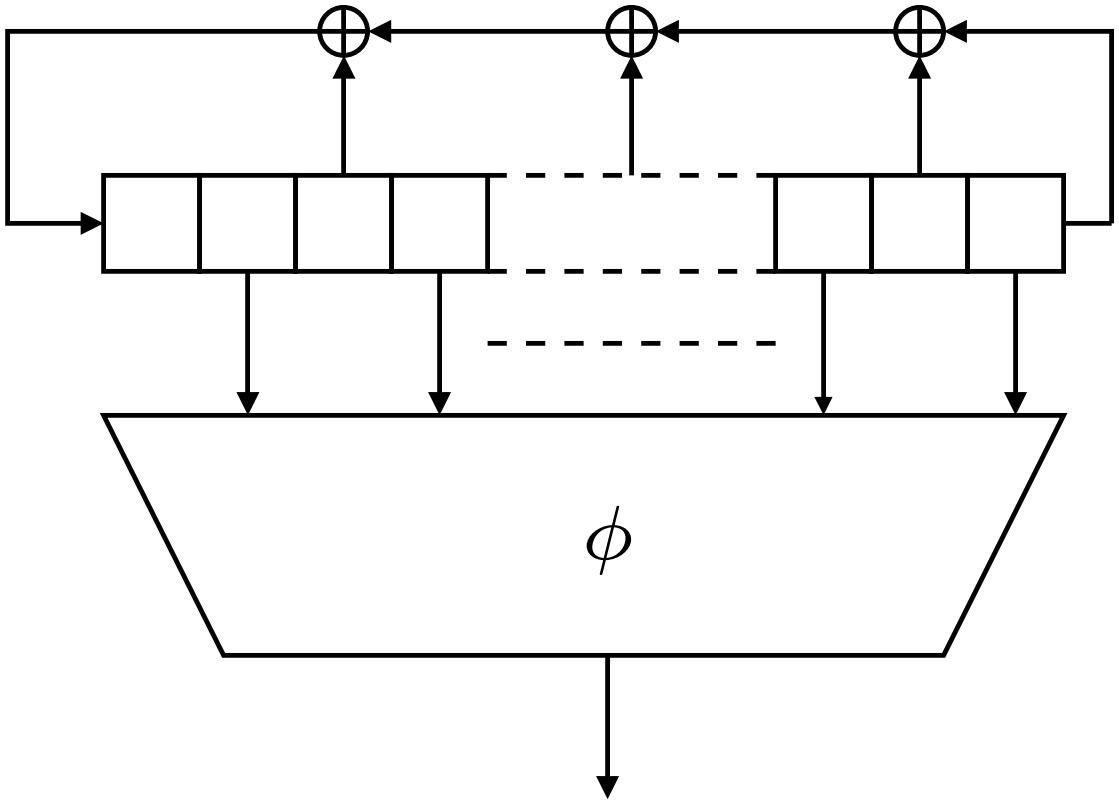
Reconstruction par élimination de Gauss

- On peut construire le système suivant :

$$\begin{aligned}
s_\ell &= c_1 s_{\ell-1} \oplus c_2 s_{\ell-2} \oplus \cdots \oplus c_\ell s_0 \\
s_{\ell+1} &= c_1 s_\ell \oplus c_2 s_{\ell-1} \oplus \cdots \oplus c_\ell s_1 \\
s_{\ell+2} &= c_1 s_{\ell+1} \oplus c_2 s_\ell \oplus \cdots \oplus c_\ell s_2 \\
&\vdots \quad \vdots \quad \vdots \\
s_{2\ell-1} &= c_1 s_{2\ell-2} \oplus c_2 s_{2\ell-1} \oplus \cdots \oplus c_\ell s_{\ell-1}
\end{aligned}$$

- **Complexité :** $O(\ell^3)$.
- Plus rapide : Berlekemp-Massey $O(\ell^2)$.

Générateur filtré



- Quels critères doient satisfaire φ ?

Fonction Booléenne

Fonctions élémentaires

négation

a	$\neg a$
0	1
1	0

$a \vee b$

a/b	0	1
0	0	1
1	1	1

$a \wedge b$

a/b	0	1
0	0	0
1	0	1

$a \oplus b$

a/b	0	1
0	0	1
1	1	0

Fonction Booléenne

Propriétés

Associativité : $(a \wedge b) \wedge c = a \wedge (b \wedge c),$

$(a \vee b) \vee c = a \vee (b \vee c).$

Commutativité : $a \wedge b = b \wedge a,$

$a \vee b = b \vee a.$

Distributivité :

$a \wedge (b \vee c) = (a \wedge c) \vee (a \wedge b)$

$a \vee (b \wedge c) = (a \vee c) \wedge (a \vee b)$

Idempotence : $a \wedge a = a$

$a \vee a = a$

Fonction Booléenne

Propriétés

Double négation : $\neg\neg a = a$

Loi de DeMorgan : $\neg(a \wedge b) = \neg a \vee \neg b,$

$\neg(a \vee b) = \neg a \wedge \neg b.$

Absorption : $a \vee (a \wedge b) = a,$

$a \wedge (a \vee b) = a.$

Constantes : $a \wedge 0 = 0,$

$a \vee 1 = a.$

Opposées : $a \wedge \neg a = 0,$

$a \vee \neg a = 1.$

Polynômes dans $\mathbb{F}_2[X_1, \cdots, X_n]$

- Un **polynôme de** $\mathbb{F}_2[X_1, \cdots, X_n]$ est une somme finie de termes monômiaux de la forme :

$$X_1^{a_1} X_2^{a_2} \cdots X_n^{a_n} \text{ avec } a_1, \cdots, a_n \geq 0.$$

- **Exemple :** $X_1^3 + X_1^2 X_2 + X_1^5 X_3^7$ est un élément de $\mathbb{F}_2[X_1, X_2, X_3]$.

Polynômes dans $\mathbb{F}_2[X_1, \cdots, X_n]$

- On définit le degré d'un monôme $X_1^{a_1} X_2^{a_2} \cdots X_n^{a_n}$ par :

$$\deg(X_1^{a_1} X_2^{a_2} \cdots X_n^{a_n}) = \sum_{i=1}^{n} a_i.$$

- Le **degré d'un polynôme** comme le maximum des degrés de ses termes monômiaux.
- Déterminer le degré du polynôme $X_1^3 + X_1^2 X_2 + X_1^5 X_3^7$.

Polynômes dans $\mathbb{F}_2[X_1, \cdots, X_n]$

- Un polynôme dans $\mathbb{F}_2[X_1, \cdots, X_n]$ permet de construire une fonction Booléenne de $\{0,1\}^n$ en considérant 0 et 1 comme des éléments de $\mathbb{F}_2$.

Forme Algébrique Normale

- Un polynôme réduit a tous ces termes monomiaux de la forme

$$X_1^{a_1} X_2^{a_2} \cdots X_n^{a_n} \text{ avec } a_1, \cdots, a_n = 0 \text{ ou } 1.$$

- Le polynôme suivant est réduit :

$$X_1 X_3 + X_2 X_3 + X_3.$$

- Toute fonction Booléenne peut s'exprimer de manière unique comme polynôme réduit.

Forme Algébrique Normale

- La *forme algébrique normale* d'une fonction Booléenne f est un polynôme de la forme :

$$f(x_0, \cdots x_{n-1}) = \sum_{j \in \mathbb{F}_2^n} a_i \prod_{i=0}^{n} x_i^{j_i},$$

avec

— $a_i \in \mathbb{F}_2$,

— $j = (j_0, \cdots, j_{n-1})$ et $\forall i < n, j_i \in \mathbb{F}_2$.

Forme Algébrique Normale

- Soit $\mathbf{A} = (a_0, \cdots, a_{n-1})$, avec $a_i \in \mathbb{F}_2$ le vecteur définissant les monômes de la FAN.

Require: f la table de vérité de f.

$f_{0,b} \leftarrow f(b), \forall b \in \mathbb{F}_2^n$

for $k = 0$ **to** n **do**

for $c = 0$ **to** $2^{n-k-1} - 1$ **do**

$f_{k+1,c} = (f_{k,2c} f_{k,2c+1} + f_{k,2c})$

end for

end for

return $\mathbf{A} = f_{n,0}$

Poids de Hamming

- Le *poids de Hamming* w_H d'un vecteur $x = (x_0, \cdots, x_{n-1})$ est défini par :

$$w_H(x) = \sum_{i=0}^{n-1} x_i.$$

- La *distance de Hamming* d entre 2 vecteurs x et y est définie par :

$$d(x, y) = w_h(x \oplus y).$$

Support

- Le support de x est l'ensemble des indices de ces composantes non-nulles :

$$\mathrm{supp}(x) = \{i \in \{0, \cdots, n-1\}, x_i \neq 0\}.$$

- Toutes les définitions s'appliquant aux vecteurs peuvent être appliquées aux fonctions.

Application aux fonctions

- Le support de f est l'ensemble des indices de ces composantes non-nulles :

$$\mathrm{supp}(x) = \{x \in \mathbb{F}_2^n, f(x) \neq 0\}.$$

- La *distance* d entre 2 fonctions f et g est définie par :

$$d(f, g) = w_h(f + g).$$

Exemple

- Soit f une fonction Booléenne à 3 variables dont la forme algébrique est :

$$f(x) = f(x_0, x_1, x_2) = x_1 + x_2 + x_2x_3 + x_1x_2x_3.$$

- On a les propriétés suivantes :
  — $deg(f) = 3$,
  — $w_H(f) = 3$,
  — $\text{supp}(f) = \{(1,0,0),(0,1,0),(1,0,1)\}$.

Poids d'une fonction

- Le poids $\mathcal{F}$ d'une fonction est définie par :

$$\begin{aligned} \mathcal{F}(f) &= \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} \\ &= 2^n - 2w_H(f). \end{aligned}$$

Transformée de Walsh-Hadamard

- La **transformée de Walsh** de f est une fonction notée $\mathcal{F}(f+\varphi(\cdot))$ définie par :

$$\begin{aligned}\mathcal{F}(f+\varphi(\cdot)):\ \mathbb{F}_2^n &\to \mathbb{Z}\\ a &\to \sum_{x\in\mathbb{F}_2^n}(-1)^{f(x)\oplus a\cdot x}\end{aligned}$$

- $\mathcal{F}(f+\varphi_a)$ est le *coefficient de Walsh* de f en a. L'ensemble $\{\mathcal{F}(f+\varphi_a), a\in\mathbb{F}_2^n\}$ est le *spectre de Walsh* de f.

Attaque par distingueur

- Si la table de vérité de φ contient plus de 0 que de 1 alors on peut monter une **attaque par distingueur**.
- Si φ a un **biais** ϵ alors un adversaire peut distinguer le suite chiffrante d'une suite réellement aléatoire.
- $\rightarrow$ notion d'équilibre pour φ.

Équilibre

- Une fonction f est dite **équilibrée** si et seulement si $\mathcal{F}(f) = 0$.

- Les fonctions linéaires sont toutes équilibrées :

$$f(s_0, \cdots, x_{\ell-1}) = a_0 s_0 \oplus \cdots \oplus a_{\ell-1} x_{\ell-1}.$$

Attaque algébrique

Shannon

- A partir de la suite chiffrante, on peut construire le système suivant :

$$\begin{aligned} k_1 &= \varphi(s_0, \cdots s_{\ell-1}) \\ k_2 &= \varphi(f(s_0, \cdots s_{\ell-1})) \\ k_3 &= \varphi(f(f(s_0, \cdots s_{\ell-1}))) \\ &\vdots \qquad\qquad \vdots \end{aligned}$$

- Que peut on faire de ce système ?

Linéarisation

- On a des monômes de degré supérieurs à 1 donc on ne peut pas appliquer l'élimination de Gauss directement.

- A chacun de ses monômes, on peut associer une variable et re-linéariser le système !

- Combien y a t'il de variables à re-linéariser ?

Exemple

- On se place dans $\mathbb{F}_2[x_1, x_2]$.
  On a le système :

$$\begin{aligned} x_1 \cdot x_2 &= 0 \\ x_1 \cdot x_2 + x_1 &= 0 \\ x_2 + x_1 &= 1 \end{aligned}$$

- On renomme :

$$(w_1, w_2, w_3) = (x_1, x_2, x_1 \cdot x_2)$$

Exemple

- On obtient :

$$\begin{aligned} w_3 &= 0 \\ w_3 + w_1 &= 0 \\ w_2 + w_1 &= 1 \end{aligned}$$

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$$

Linéarisation

Complexité

- La complexité en temps est de l'ordre de

$$\left(\sum_{i=0}^{d} \binom{\ell}{i}\right)^3 .$$

- Complexité en données de l'attaque ?
- Complexité en mémoire de l'attaque ?

Linéarisation

Critère

- L'attaque de linéarisation est plus efficace que la recherche exhaustive si

$$\deg(\varphi) \leq 0.42 \left(\frac{|K|}{1 + \log_2 |K|} \right)$$

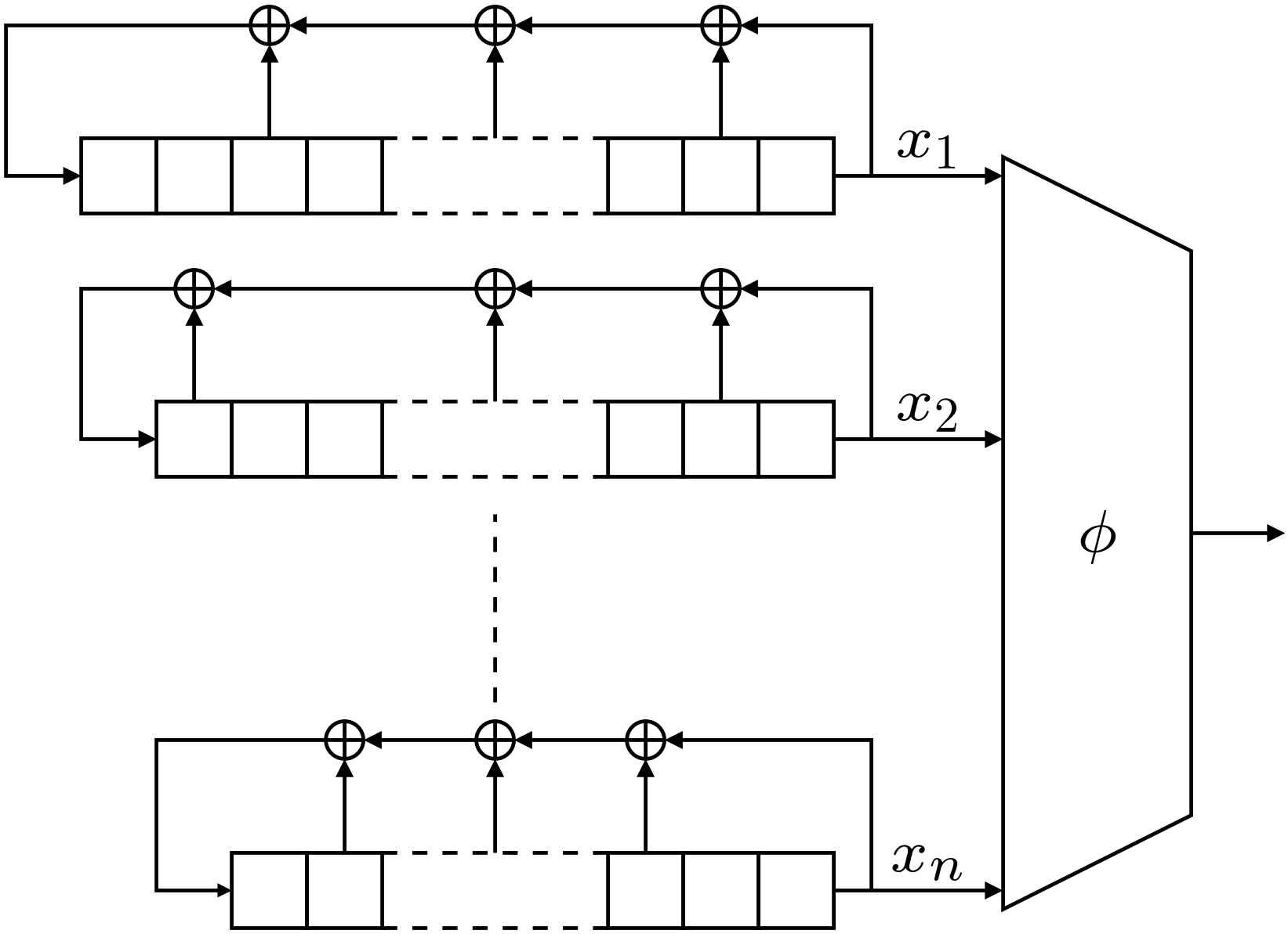
Non-linéarité

- La non-linéarité de f est définie par :

$$\begin{aligned}\mathcal{N}(f) &= \min_{a\in\mathbb{F}_2^n}(w_H(f+\varphi_a), w_H(f+\varphi_a+1)) \\ &= 2^{n-1} - \frac{\mathcal{L}(f)}{2}.\end{aligned}$$

- avec $\mathcal{L}(f) = \max_{a\in\mathbb{F}_2^n}|\mathcal{F}(f+\varphi_a)|$

Générateur combiné



Attaque par corrélation

- Attaque de type **diviser-pour-régner**.
- L'état interne du générateur à l'instant t est divisible en deux parties x_t et y_t de tailles respectives ℓ et $(n - \ell)$
- On essaye de retrouver x_t.

Attaque par corrélation

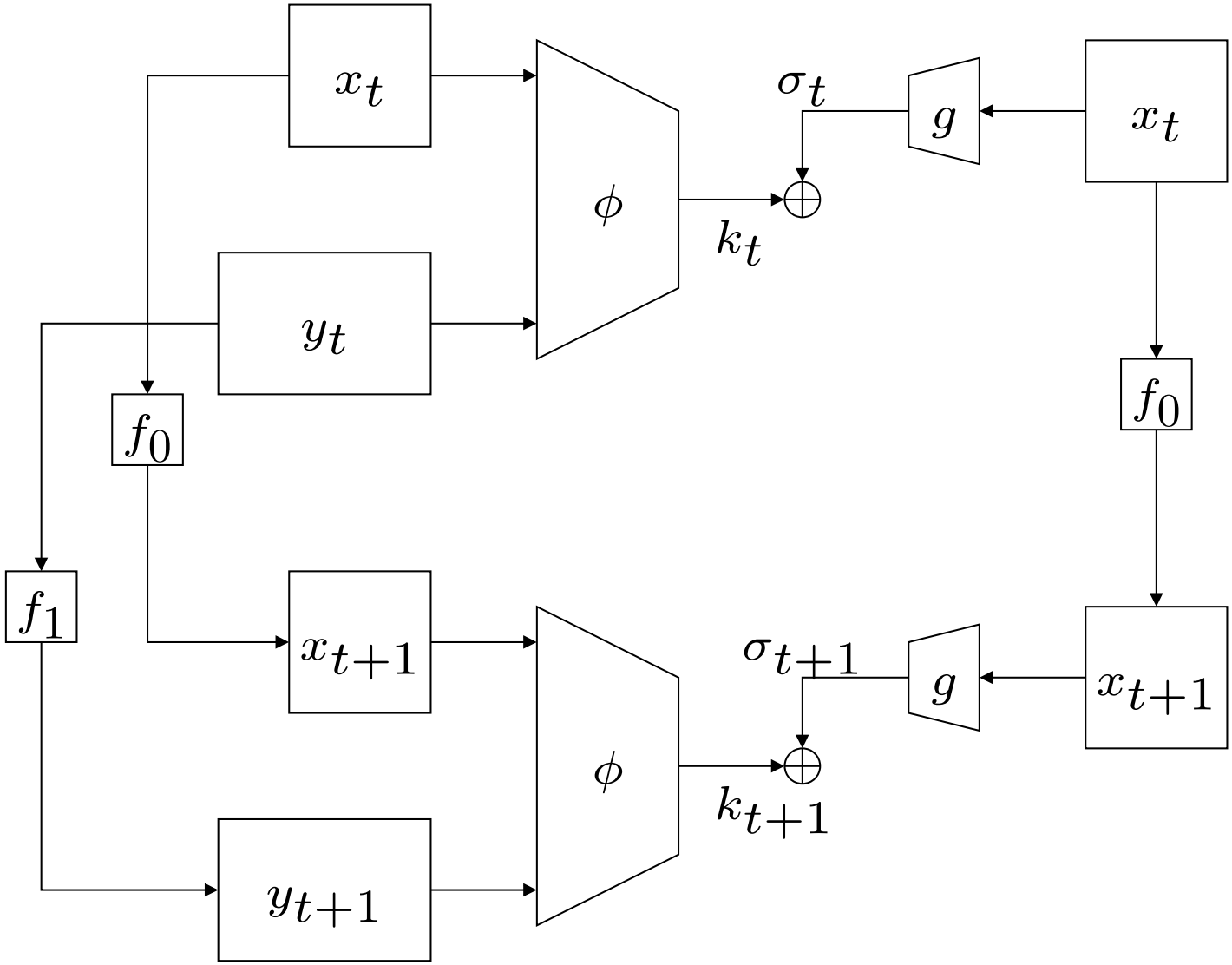
- L'attaque s'applique s'il existe une fonction g à ℓ variables qui coïncide avec la sortie de f dans plus de la moitié des cas :

$$p_g = Pr_{XY}[\varphi(X, Y) = g(X)] > 1/2$$

- De plus, on a :

$$Pr[k_t = \sigma_t] = p_g > 1/2$$

Attaque par corrélation



Attaque par corrélation

- Pour tout vecteur $x_0 \in \{0,1\}^\ell$, on calcul $\sigma_t = g \circ f_0(x_0)$
- On calcule $c(k_t, \sigma_t) = \sum_{i=0}^{N-1} (-1)^{\sigma_i + k_i}$
- Retourner x_0 qui maximise $c(k_t, \sigma_t)$.

Attaque par corrélation

Complexité

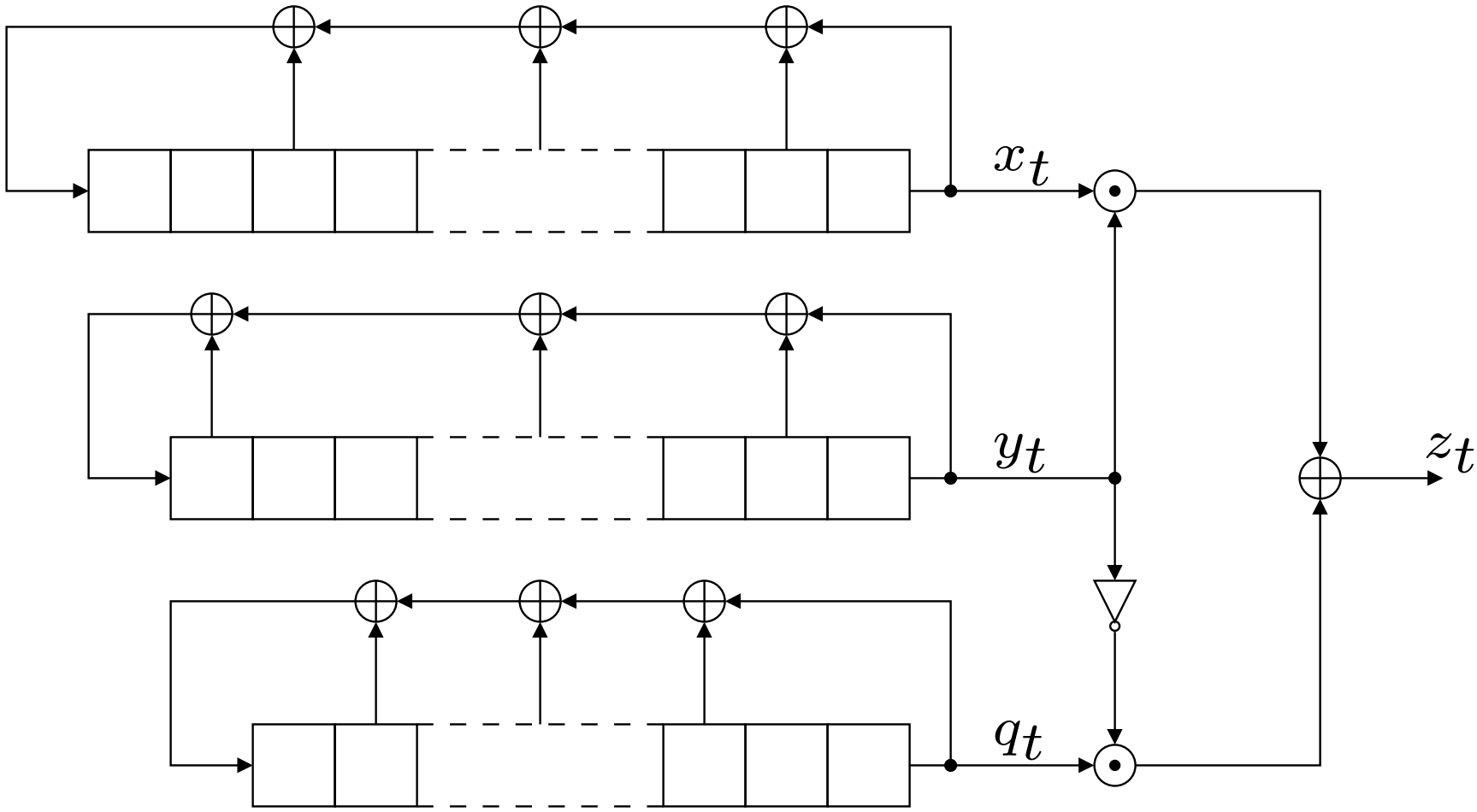
- Le nombre de bits N de suite chiffrante nécessaires pour retrouver la valeur correcte de x_0 est de l'ordre de :

$$\left(\frac{1}{p_g - 0.5}\right)^2$$

- La complexité en temps pour retrouver les ℓ bits x_0 est $\ell 2^\ell$.

Générateur combiné

Geffe



Immunité aux corrélations

- f est sans corrélation d 'ordre t si la distribution de probabilités de sa sortie est inchangée quand on fixe n'importe quel ensemble de t variables d'entrées et que les $(n-t)$ autres forment un ensemble de variables aléatoires indépendantes uniformément distribuées.

t-résilience

- f est t-résiliente si f est sans corrélation d'ordre t et équilibrée.

- Une fonction Booléenne f est t-résiliente si et seulement si

$$\forall a \in \mathbb{F}_2^n,\ 0 \leq w_H(a) \leq t, \mathcal{F}(f + \varphi + a) = 0$$

Borne de Siegenthaler

- Soit f sans corrélation d'ordre t et de degré algébrique d, alors on a la relation suivante :

$$d + t \leq n.$$

Si f est équilibrée et si $t < n - 1$ alors

$$d + t \leq n - 1.$$

RC4 : Rivest Cipher 4

Généralités

Système	Type	Création
RC2	Bloc	1987
RC4	**Flot**	**1987**
RC5	Bloc	1994
RC6	Bloc	1997

- RC4 est le chiffrement par flot le plus utilisé au monde : WEP, WPA, XBOX, Skype. . .
- **Malheureusement, RC4 est très faible !**

RC4 : caractéristiques

- **Clef :** longueur $40 \leq \ell \leq 128$ bits.
- **État interne :**
  — RC4 travaille sur un tableau S de 256 octets.
  — 2 compteurs.
- **Fonction d'initialisation** qui dépend de la clef K.
- **Fonction de génération :**
  — Fonction de mise à jour f.
  — Fonction de filtrage ϕ.

- *Pour simplifier la description de l'algorithme, on va reduire RC4 à un tableau de 8 octets (TinyRC4).*

TinyRC4

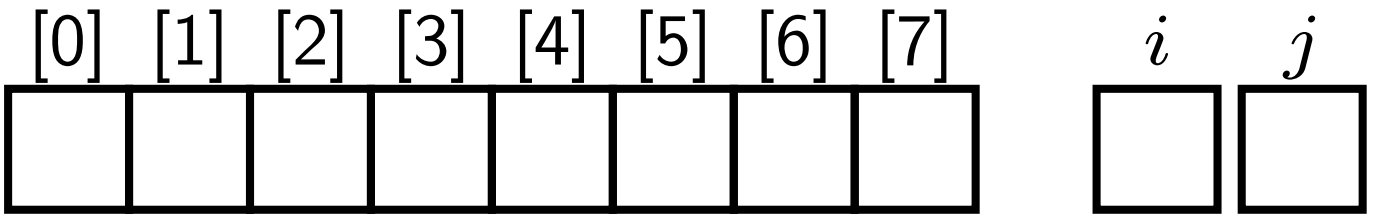
Fonction d'initialisation

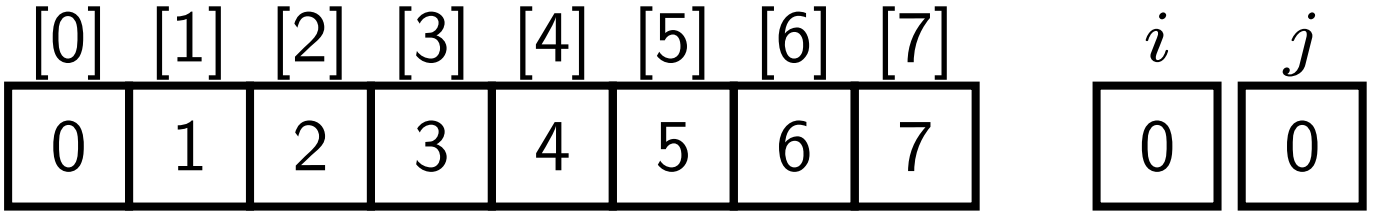
- La fonction de d'initialisation a pour objectif d'initialiser l'état interne avec une **permutation pseudo-aléatoire** qui dépend de la clef des entiers positifs plus petit que 8.

- Résultat attendu (exemple) : $\{0, 1, 3, 7, 4, 6, 5, 2\}$

TinyRC4

Fonction d'initialisation





TinyRC4

Fonction d'initialisation

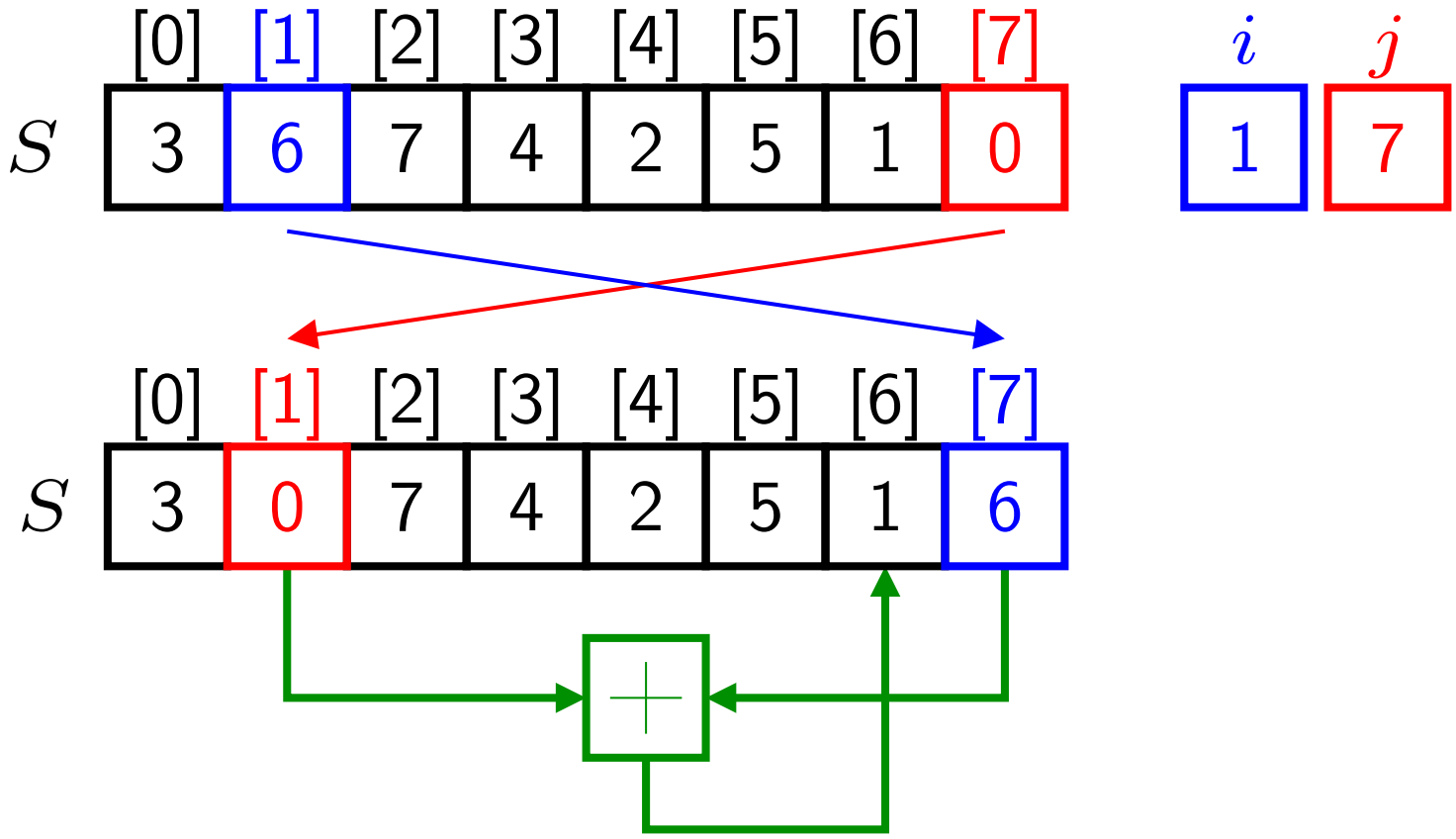
	[0]	[1]	[2]	[3]	[4]	[5]
K	2	114	84	0	201	48

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

i	j
0	0

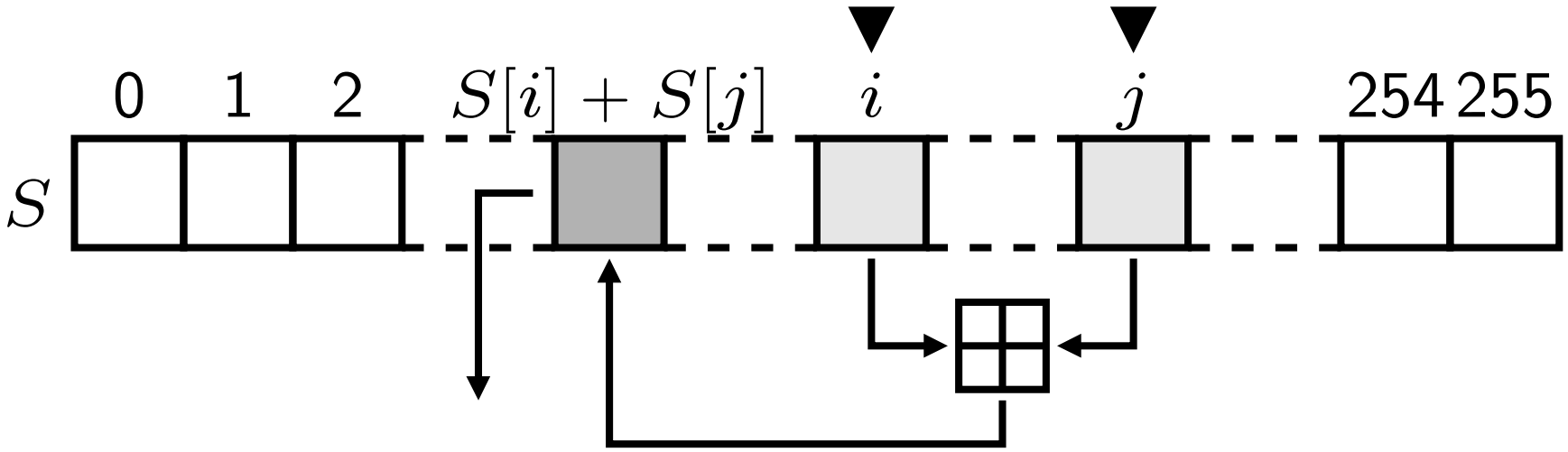
- Transition :
  - $i_{t+1} = i_t + 1$
  - $j_{t+1} = j_t + S[i_t] + K[i_t] \bmod 8$
  - $\texttt{swap}(S[i_t], S[j_{t+1}])$

Génération de la suite chiffrante



- Tout ce qui s'applique à TinyRC4 s'appliquera à RC4.

Génération de la suite chiffrante



Code C de RC4

- Code très simple !

```
int i=0,j=0,x,t;
for (x=0; x < len; ++x)
{
  i = (i + 1) % 256;
  j = (j + state[i]) % 256;
  t = state[i];
  state[i] = state[j];
  state[j] = t;
  out[x] = state[(state[i] + state[j]) % 256];
}
```

Code C de RC4

Initialisation

```
int i,j=0,t;
for (i=0; i < 256; ++i)
  state[i] = i;
for (i=0; i < 256; ++i)
{
  j = (j + state[i] + key[i % len]) % 256;
  t = state[i];
  state[i] = state[j];
  state[j] = t;
}
```

Justification de la construction

RC4

- Code très simple !

```
int i=0,j=0,x,t;
for (x=0; x < len; ++x)
{
  i = (i + 1) % 256;
  j = (j + state[i]) % 256;
  out[x] = state[(state[i] + state[j]) % 256];
}
```

- On vient d'enlever le `swap`.

Justification de la construction

swap

- On note i_t, j_t et z_t les valeurs respectives de i, j et de la suite chiffrante après t itérations.
- Donner les formules pour i_{t+256}, j_{t+256}, j_{t+512}, z_{t+256} et z_{t+512}. Simplifier les : que constatez vous ?

Période de RC4 simplifié

- Soit i_t et j_t les indices des compteurs après t itérations. On a pour i_t et z_t :

$$\begin{aligned} i_{t+256} &= i_t \bmod 256 \\ z_{t+256} &= S[S[i_{t+256}] + S[j_{t+256}]] \\ &= S[S[i_t] + S[j_{t+256}]] \end{aligned}$$

- Il faut simplifier $S[j_{t+256}]$.

Période de RC4 simplifié

j_{t+256}

$$
\begin{aligned}
j_{t+256} &= j_t + \sum_{k=0}^{255} S[i_{t+k}] \bmod 256 \\
&= j_t + \sum_{k=0}^{255} S[k] \bmod 256
\end{aligned}
$$

S étant une permutation, on peut simplifier :

$$
\begin{aligned}
\sum_{k=0}^{255} S[k] &= \frac{255 \cdot 266}{2} \bmod 256 \\
&= 128 \bmod 256
\end{aligned}
$$

Période de RC4 simplifié

j_{t+256}

On obtient donc :

$$\begin{aligned} j_{t+256} &= j_t + 128 \bmod 256 \\ j_{t+512} &= j_t + 256 \bmod 256 \\ &= j_t \bmod 256 \end{aligned}$$

On a donc la relation :

$$\begin{aligned} z_{t+512} &= S[S[i_{t+512}] + S[j_{t+512}]] \\ &= S[S[i_t] + S[j_t]] \\ &= z_t \end{aligned}$$

Attaques sur RC4

- Pour attaquer un algorithme de chiffrement par flot, on se concentre donc sur trois faiblesses :

  ▷ **Propriétés sur la suite chiffrante**

  ▷ **Propriété d'inversion**

  ▷ **Propriété sur l'initialisation**

Inversion de TinyRC4

et donc de RC4

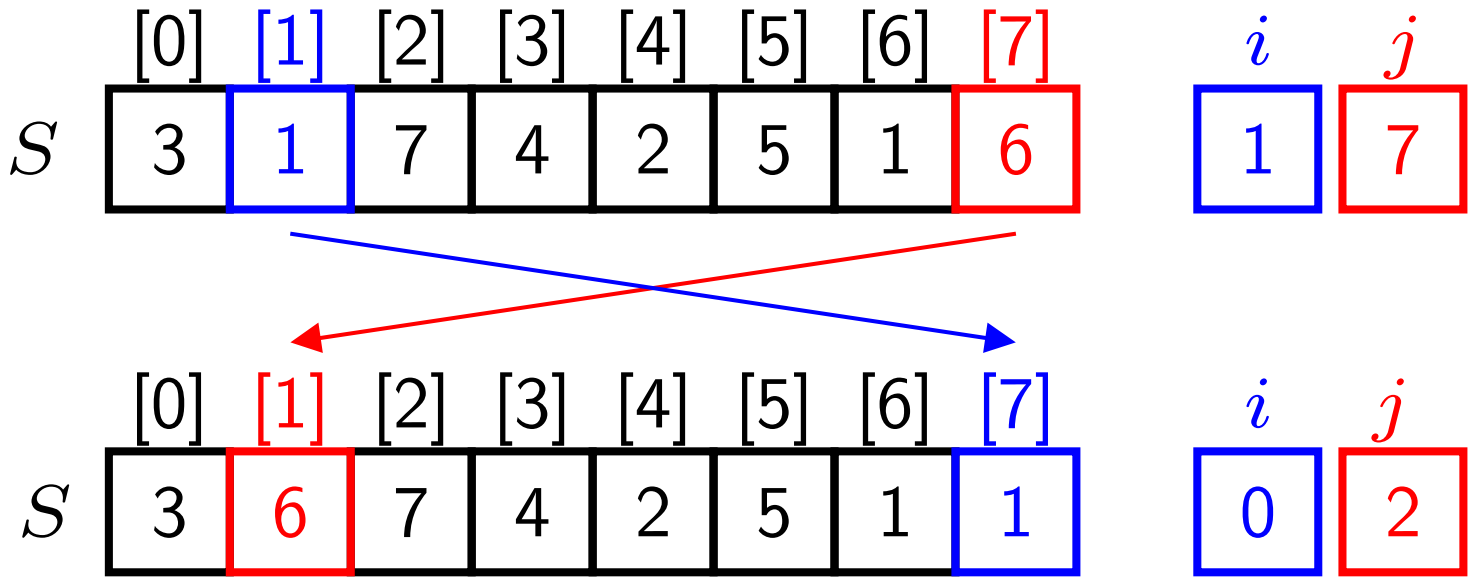
- On suppose que l'on connaît l'état interne après t tours ainsi que le dernier octet de la suite chiffrante.
- On connaît donc S et $i_t = t$.
- On peut retrouver facilement j_t car :

$$k_t = S[S[i_t] + S[j_t]].$$

On connaît k_t, i_t et S, on peut retrouver j.

Inversion de TinyRC4

Exemple



- Fonction d'inversion :
  - $\texttt{SWAP}(S[i],S[j])$
  - $i_{t-1} = i_t - 1 \bmod 8$
  - $j_{t-1} = j_t - S[i_{t-1}] \bmod 8$

Propriété d'initialisation

Biais de Roos

- Si l'initialisation de S était effectuée par une permutation aléatoire alors on aurait :

$$\forall y \in \mathbb{Z}/256\mathbb{Z}, \mathbf{Pr}(S[y] = c) = 1/256$$

avec $\forall c \in \mathbb{Z}/256\mathbb{Z}$.

- L'initialisation de RC4 se comporte-t-elle comme une permutation aléatoire ?

Propriété d'initialisation

Biais de Roos

- On définit f_y par

$$\begin{aligned} f_y &= \sum_{x=0}^{y} K[x] + x \\ &= \frac{y(y+1)}{2} + \sum_{x=0}^{y} K[x]. \end{aligned}$$

- La valeur la plus probable pour $S[y]$ à la fin de l'initialisation est $S[y] = f_y$. **[Roos 1995]**

Propriété d'initialisation

Biais de Roos

y	$\mathbf{Pr}(S[y] = f_y)$							
0-7	0.370	0.368	0.362	0.358	0.349	0.340	0.330	0.322
8-15	0.309	0.298	0.285	0.275	0.260	0.245	0.229	0.216
16-23	0.203	0.189	0.173	0.161	0.147	0.135	0.124	0.112
24-31	0.101	0.090	0.082	0.074	0.064	0.057	0.051	0.044
32-39	0.039	0.035	0.030	0.026	0.023	0.020	0.017	0.014
40-47	0.013	0.012	0.010	0.009	0.008	0.007	0.006	0.006

En notant que $\frac{1}{256} = 0.00390625$

Biais de Roos

- On prend une clef de 40 bits :

$$K = \{106, 59, 220, 65, 34\}$$

y	0	1	2	3	4	5	6	7
f_y	106	166	132	200	238	93	158	129
$S_{256}[y]$	230	166	87	48	238	93	68	239
y	8	9	10	11	12	13	14	15
f_y	202	245	105	175	151	229	21	142
$S_{256}[y]$	202	83	105	147	151	229	35	142

Biais de Roos

- Pour $S[1]$ et $S[2]$ on a :

$$\mathbf{Pr}(S[1] = f_1) \approx \left(\frac{256-1}{256}\right)^{256}$$

$$\mathbf{Pr}(S[2] = f_2) \approx \left(\frac{256-1}{256}\right)^{256}$$

$$f_1 = K[0] + K[1] + 1$$

$$f_2 = K[0] + K[1] + K[2] + 3$$

- On a des équations sur la clef à partir de la graine.

Généralisation

- On a :

$$\mathbf{Pr}(S[y] = f_y) = \left(\frac{256 - y}{256}\right) \cdot \left(\frac{256 - 1}{256}\right)^{256 + \frac{(y+1)y}{2}} + \frac{1}{256}$$

Pour la preuve voir le résumé de cours.

Recouvrement de la clef

- On connaît S et $K = \{106, 59, 220, 65, 34\}$

y	0	1	2	3	4	5	6	7
f_y	106	166	132	200	238	93	158	129
$S[y]$	230	166	87	48	238	93	68	239
y	8	9	10	11	12	13	14	15
f_y	202	245	105	175	151	229	21	142
$S[y]$	202	83	105	147	151	229	35	142

- Chaque fois que l'on a $S[y] = f_y$, on obtient une équation sur la clef. On peut donc former des systèmes d'équations.

Recouvrement de la clef

$$1 + \sum_{x=0}^{1} K[x] = 166 \quad (1)$$

$$10 + \sum_{x=0}^{4} K[x] = 238 \quad (2)$$

$$15 + \sum_{x=0}^{5} K[x] = 93 \quad (3)$$

$$36 + \sum_{x=0}^{8} K[x] = 202 \quad (4)$$

Attaque FMS

Fluhrer-Mantin-Shamir

- Soit $S_j[i]$ la valeur du $i^{\text{ème}}$ octet de la permutation S après la $j^{\text{ème}}$ itération.
- On fait l'hypothèse que l'on a :
  — $S_0[2] = 0$,
  — $S_0[1] \neq 2$.

  Posons $x = S_0[2]$ et $y = S_0[x]$. Avec

$$\begin{cases} i_1 = 1 \\ j_1 = 0 + S_0[i_1] = S_0[1] = x \end{cases}$$

On échange $S_0[1]$ et $S_0[x]$.

Attaque FMS

Biais sur la suite chiffrante

- La suite chiffrante est obtenue par :

$$\begin{aligned} z_1 &= S_1[S_1[1] + S_1[x] \bmod 256] \\ &= S_1[x + y] \end{aligned}$$

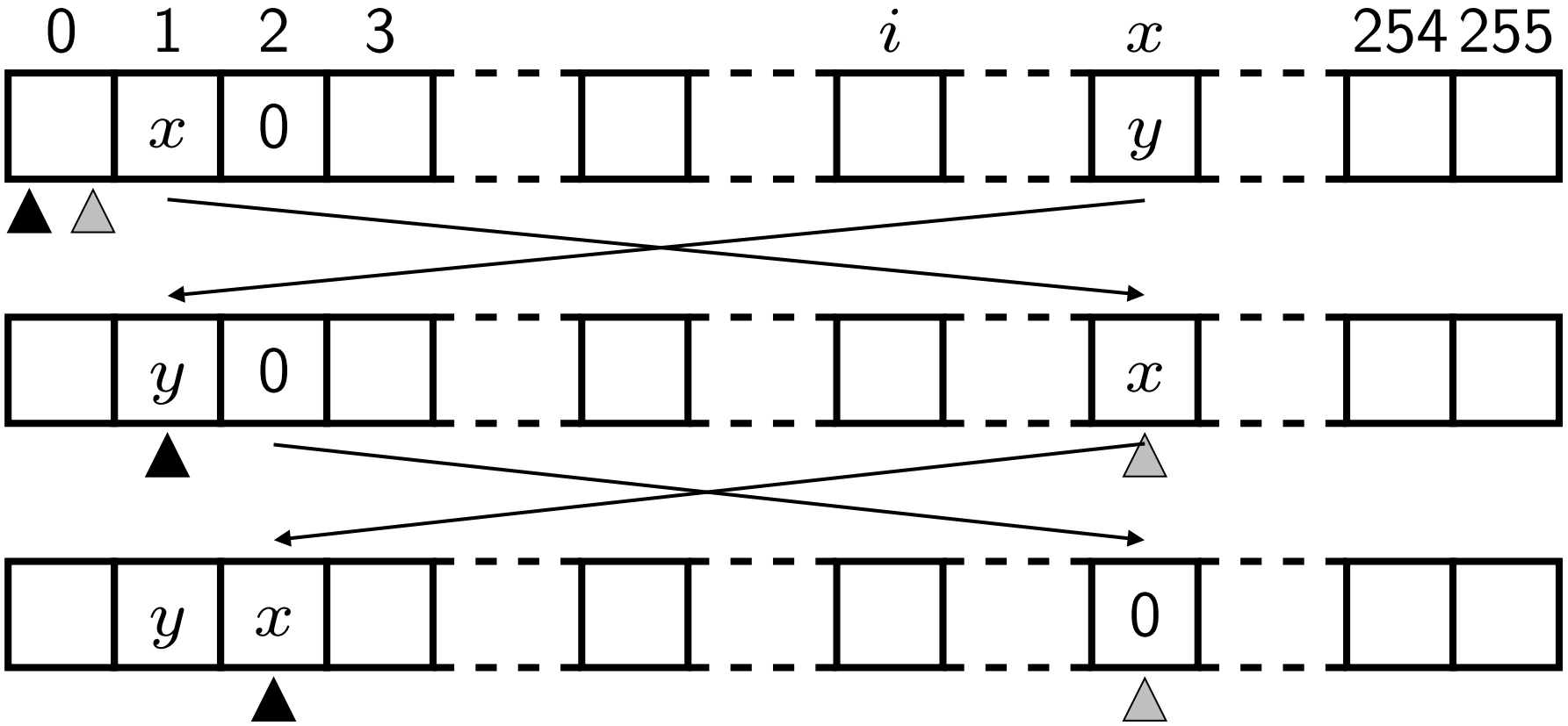
On a : $\begin{cases} i_2 = 2 \\ j_2 = x + S_1[i_2] = x + S_1[2] = x \end{cases}$

- Le deuxième octet de la suite chiffrante est donnée par :

$$z_2 = S_2[S_2[2] + S_2[x] \bmod 256] = 0$$

Attaque FMS

Mécanique



Attaque FMS

Biais

Une permutation aléatoire a les propriétés :

— $S_0[2] = 0$,

— $S_0[1] \neq 2$,

avec probabilité $\frac{1}{256} \cdot \frac{255}{256}$.

Pour RC4, la probabilité d'avoir $z_2 = 0$ est donnée :

$$\begin{aligned} \mathbf{Pr}(z_2 = 0) &= 1 \cdots \frac{255}{256} + \frac{1}{256} \cdot \frac{256^2 - 255}{256^2} \\ &\approx \frac{1}{128} \end{aligned}$$

Sources

- **RC4 Stream Cipher and Its Variants.** *P. Goutam et S. Maitra, 2011 CRC Press.*
- **Weaknesses in the Key Scheduling Algorithm of RC4.** *S. R. Fluhrer, I. Mantin et A. Shamir, SAC 2001, Springer-Verlag.*
- **A Class of Weak Keys in the RC4 Stream Cipher.** *Andrew Roos, posté sur sci.crypt, 1995.*