Cryptanalyse : chiffrement par flot

Pierre-Alain Fouque

Département d'Informatique

Ecole normale supérieure

Plan

- Technique générale:
  - Piling up lemma
  - birthday paradox (et généralisation)
  - distingueur
- Attaques générales de streams:
  - recherche exhaustive et compromis temps/mem
  - attaque par corrélation (rapide)
- Attaques spécifiques de streams: A5, RC4,..

Piling-Up Lemma

- Th: Soit n VA binaires indépendantes $X_1,\ldots,X_n$ de biais ε_i, ie tq $Pr(X_i=1)=1/2(1+\varepsilon_i)$. Alors le biais ε de $\bigoplus_{i=1}^{n} X_i$ est $(-1)^n\prod_{i=1}^{n} \varepsilon_i$.
- Preuve: Récurrence sur n.
- $Pr(X=1)=Pr(\bigoplus_{i=1}^{n-1} X_i=1)Pr(X_n=0)+Pr(\bigoplus_{i=1}^{n-1} X_i=0)Pr(X_n=1)=1/4(1+(-1)^{n-1}\prod_{i=1}^{n-1} \varepsilon_i)\times(1-\varepsilon_n) + 1/4(1-(-1)^{n-1}\prod_{i=1}^{n-1} \varepsilon_i)\times(1+\varepsilon_n)=1/2-2^{n-1}\prod_{i=1}^{n} \varepsilon_i$.

Paradoxe des anniversaires

- Si on choisit indépendamment et au hasards N éléments dans D, alors Pr (collision)>1/2 si $N>\sqrt{D}$.
- Etant donné 2 listes L_1, L_2, d'éléments choisis uniformément et indépendamment dans $\{0,1\}^n$, trouver $x_1 \in L_1$ et $x_2 \in L_2$ tq $x_1 \oplus x_2 = 0^n$. Pr(solution x1 et x2)$=|L_1| \times |L_2|/2^n$

Paradoxe des anniversaires

- Version généralisée: étant donné k listes L_i, trouver $\forall i$, $x_i \in L_i$ tq $\bigoplus_{i=1}^{n} x_i = 0^n$.
- Existence d'une solution si $\prod_{i=1}^{n} |L_i| = 2^n$.
- Cas n=4. Existence solution si $|Li| \approx 2^{n/4}$. Trouver une telle solution peut se faire en temps et mémoire $2^{n/3}$.
- Construire 4 listes de $2^{n/3}$ élé. chacune et un arbre de collision: apparier les n/3 bits de L_1 et L_2 dans L'_1 et ceux de L_3 et L_4 dans L'_2. $|L'_i| \approx 2^{n/3}$. Donc, collision entre ces deux listes sur 2n/3 bits avec proba. 1/2.

Distingueur

- Test d'hypothèse: $H=\{1,2\}$
- Observation X dans ensemble E
- Deux distributions: $e \in E$, $D_i(e)=P(X=x|H=i)$
- $\sum_{e \in E} D_i(e)=1$, $E' \subset E$, $D_i(E')= \sum_{e \in E'} D_i(e)$
- Règle de décision: A distingueur défini par E_1= domaine d'acceptation où A retourne 1
- $p_{succ}(A)=p_1 \sum_{x \in E1} P(X=x|H=1)+p_2 \sum_{x \in E2} P(X=x|H=2)$
- $p_{succ}(A)=p_1 D_1(E_1)+p_2 D_2(E_2)$

Distingueur optimal

- Lemme de Neyman Pearson:
- $S=\{x \in E | p_1 D_1(x) > p_2 D_2(x)\}$
- Distingueur optimal (A_{opt}): $E_1=S$
- $\forall A$, $p_{succ}(A) \leq p_{succ}(A_{opt})$
- Rapport de vraisemblance: $D_1(x)/D_2(x) >? \lambda$ où $\lambda=p_2/p_1$.
- Si $\lambda=1$, A_{opt} retourne 1 si $D_1(x)>D_2(x)$
- Si $p_1=0.9$, A retourne toujours 1, $p_{succ}(A)=0.9$

Expériences multiples

- M-uplet au lieu d'une valeur
- D_1^M et D_2^M les deux distributions
- On peut montrer que si elles sont proches: si on fait $M=d/\sum_{x\in E} (D_1(x)-D_2(x))^2/D_1(x)$

  tirages, alors la proba de distinguer est $p_{succ}=1-1/\sqrt{2\pi}\int_{-\infty}^{\sqrt{d}/2} \exp(-u^2/2)\, du$.
- Si $E=\{0,1\}$, et D_2 dist. uniforme, $D_1(1)=0.5(1+\varepsilon)$, ε est le biais et on va mq $M\approx\varepsilon^{-2}$ pour distinguer

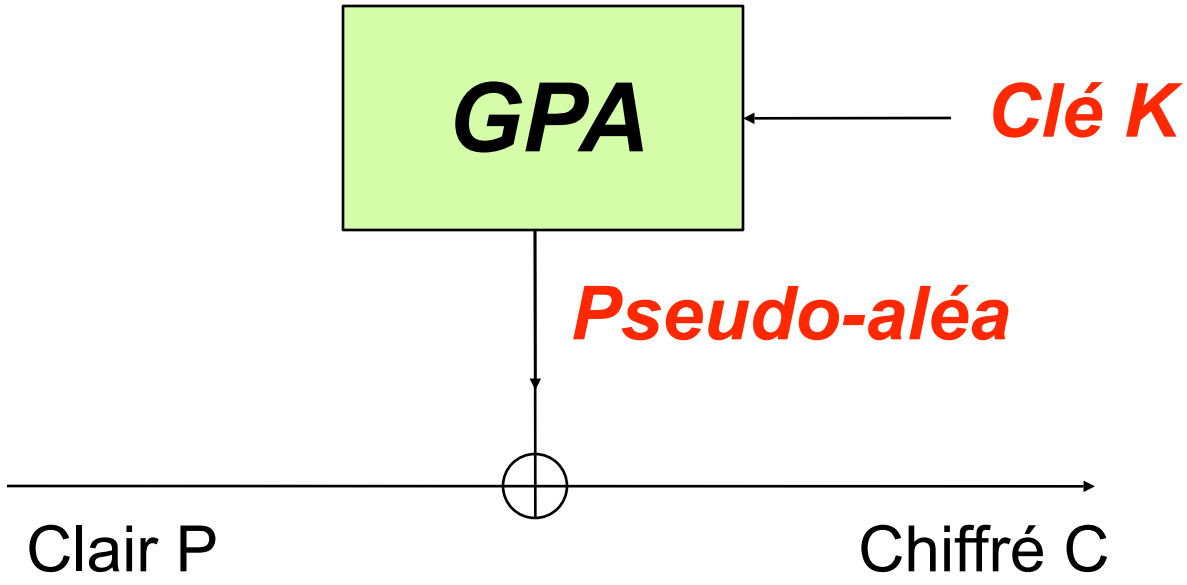
Approximation gaussienne

- Si D_2^M uniforme sur $\{0,1\}^M$, $D_2^M(e)=1/2^M$.
- T VA nombre d'indices i de e tq $e_i=1$. Si T=t, $D_1^M(e)=1/2^M(1+\varepsilon)^t(1-\varepsilon)^{M-t}$
- E_t l'ensemble des M-uplet tq T=t,
- $D_1^M(T=t)=D_1^M(E_t)=C^t_M\times 1/2^M(1+\varepsilon)^t(1-\varepsilon)^{M-t}$
- $D_2^M(T=t)=C^t_M\times 1/2^M$
- Pour M grand, D_1^M se comporte comme gaussienne, $Esp=M/2(1+\varepsilon)$, $\sigma\approx\sqrt{M}/2$.

Gaussienne (suite)

- De même pour $D_2^M(T)$ centrée en M/2.
- Elles sont approchées par $P(T \leq t)=1/(\sigma\sqrt{2\pi})\int_{-\infty}^{t} \exp(-1/2(x-Esp)^2/\sigma)dx$
- $P(T \leq Esp+n\sigma)=1/(\sqrt{2\pi})\int_{-\infty}^{n} \exp(-1/2u^2)du$
- Distingueur optimal: si $p_1=p_2=0.5$, alors $n=\varepsilon\sqrt{M}/2$, $p_{succ}=1-1/2erfc(n/\sqrt{2})$, où $p_{succ}(n)$ =0.5 (n=0), 0.84134 (n=1), 0.97723(n=2), 0.99999 (n=5), donc il faut prendre $M \approx \varepsilon^{-2}$.

« Stream Cipher »



Générateur Pseudo-Aléatoire (GPA)
Traitement « bit à bit » des données

« Stream Cipher »

Suite binaire pseudo-aléatoire

$$GPA(K) = s_1\ s_2 \ldots s_n \ldots$$

Clair

$$P = p_1\ p_2 \ldots p_n \ldots$$

Chiffré

$$C = c_1\ c_2 \ldots c_n \ldots$$

On a :

$$\forall i,\ c_i = p_i \oplus s_i$$

Vigenère

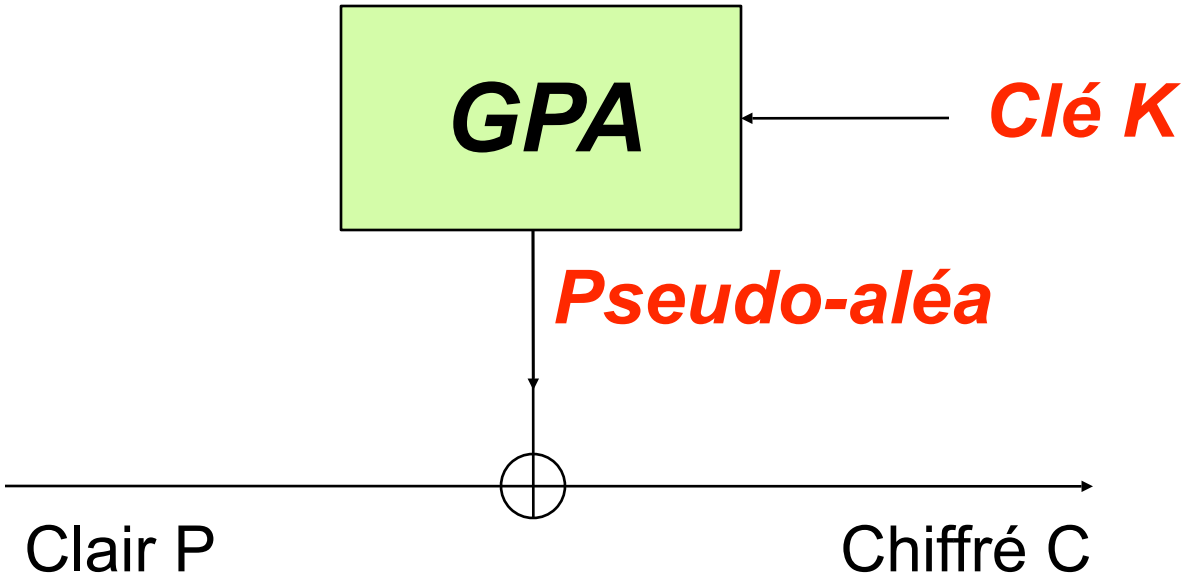
- clé: mot de passe key
- chiffrement: E_{key}(BONJOUR)=(B+k,O+e,N+y,J+k,O+e,U+y,R+k)
- Attaque shift: $\sum_{i=0}^{25} p_i^2 \approx 0.065$ sur le clair
- q_i proba de la i-ème lettre dans le chiffré
- Si la clé est k, on attend que q_{i+k} soit environ p_i pour tout i
- $I_j=\sum_{i=0}^{25} p_i.q_{i+j}$ pour $j \in \{0,...,25\}$, $I_k \approx 0.065$

Commentaires

- Le pseudo-aléa généré est de même taille que le message à chiffrer
- Pour la cryptanalyse, on se place toujours à clair connu

  $\Rightarrow$ L'attaquant connaît les sorties du GPA

But de l'attaque



1 – Trouver K à partir du Pseudo-aléa

2 – Distinguer le Pseudo-aléa de «vrai» aléa

Qualité de l'aléa

Si le pseudo-aléa généré n'est pas « bon » :

$$\text{Proba}[s_i = 0] = 0.5 + \varepsilon$$

Alors le chiffré fait « fuir » de l'information sur le clair :

$$c_i = s_i \oplus p_i$$

$$\text{Proba}[c_i = p_i] = 0.5 + \varepsilon$$

Différentes attaques

- Attaques génériques
  - Recherche exhaustive
  - Compromis Temps / Mémoire
- Attaques par Corrélation
- Attaques dédiées contre certains algorithmes

Attaques génériques

- **Attaques génériques**
  - **Recherche exhaustive**
  - **Compromis Temps / Mémoire**
- Attaques par Corrélation
- Attaques dédiées contre certains algorithmes

Attaques génériques

Clé de k bits $\Rightarrow 2^k$ clés possibles

56	256	Cycles/an processeur à 2 Ghz
56	256	Taille de clé DES
64	264	Calcul distribué (07/2002)
72	272	Calcul distribué en cours
128	2128	Taille de clé standard AES

Recherche exhaustive

- Est-ce toujours possible ? Il faut une condition pour identifier la bonne clé (ex : clair + chiffré correspondant)
- Pour un stream cipher, il faut connaître k bits de pseudo-aléa pour avoir une bonne chance de succès

Recherche exhaustive

- On cherche une clé inconnue K_{vraie}
- On suppose connus $s_1 \ldots s_m$
- for $K_{candidate}$ = 1 to 2^k
  - Calculer GPA($K_{candidate}$) = $s'_1 \ldots s'_m$
  - Si ($s_1 \ldots s_m$) == ($s'_1 \ldots s'_m$), on prédit que

$$\mathbf{K_{vraie} = K_{candidate}}$$

Recherche exhaustive

- On cherche une clé inconnue K_{vraie}
- On suppose connus $s_1 \ldots s_m$
- for $K_{candidate}$ = 1 to 2^k
  - Calculer GPA($K_{candidate}$) = $s'_1 \ldots s'_m$
  - Si ($s_1 \ldots s_m$) == ($s'_1 \ldots s'_m$), on prédit que

$$\mathbf{K_{vraie} = K_{candidate}}$$

Probabilité de fausse alerte = 2^{-t}

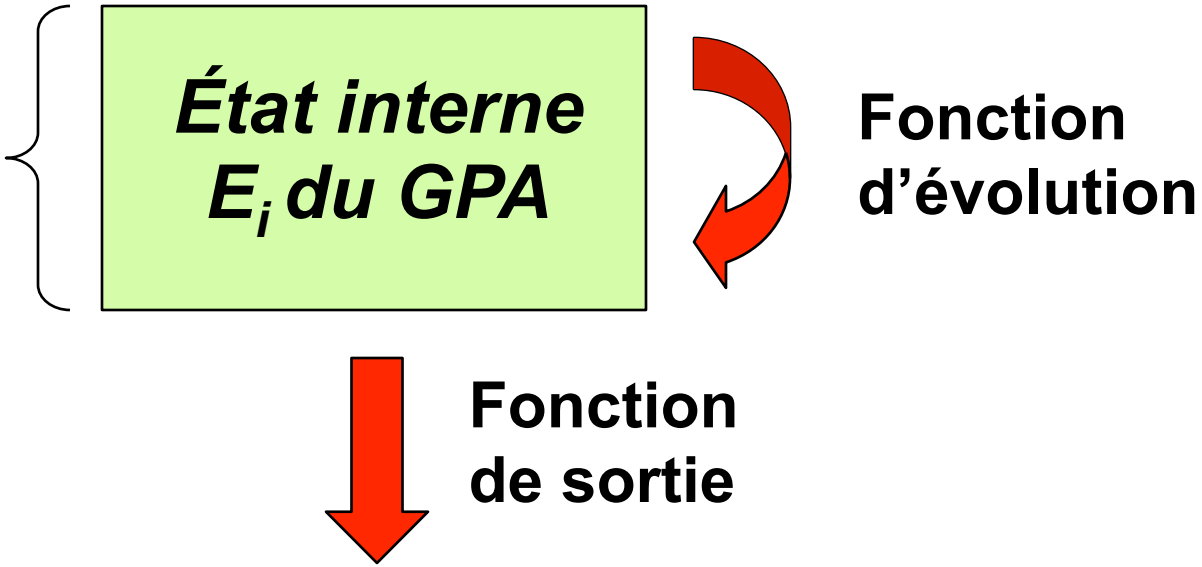
Nombre de fausses alertes $\approx 2^k * 2^{-t}$

Dès que t>k, on a de bonnes chances de succès

État interne du GPA

En général E_0 est équivalent à la clé

L'état interne contient t cases mémoires de 1 bit



État interne du GPA

- La sorties du GPA pour $i \geq i_0$ (notées s_i) ne dépendent que de l'état interne E_i à l'instant i

- On peut donc effectuer une recherche exhaustive sur E_i en vérifiant le bon candidat grâce aux $\{s_i\}_{i \geq i_0}$

- Cela coûte 2^t opérations

Conclusion

Il faut que

- La clé K
- L'état interne du GPA

soient assez grands pour se protéger de la recherche exhaustive

Typiquement

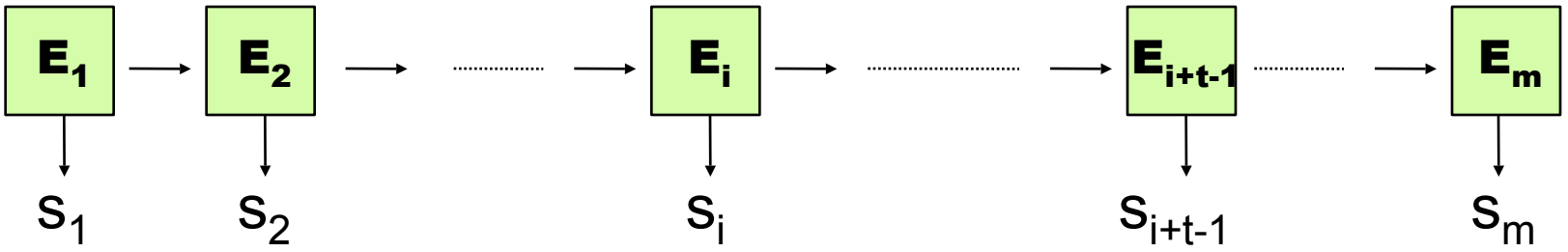
$$t \geq k \geq 128$$

Compromis Temps / Mémoire

- On s'autorise
  - à faire un précalcul (pendant un temps P)
  - à stocker des données en mémoire à l'issue du précalcul (M bits de stockage)
- Est-il alors possible de réduire le temps $T = 2^t$ passé dans une attaque ?

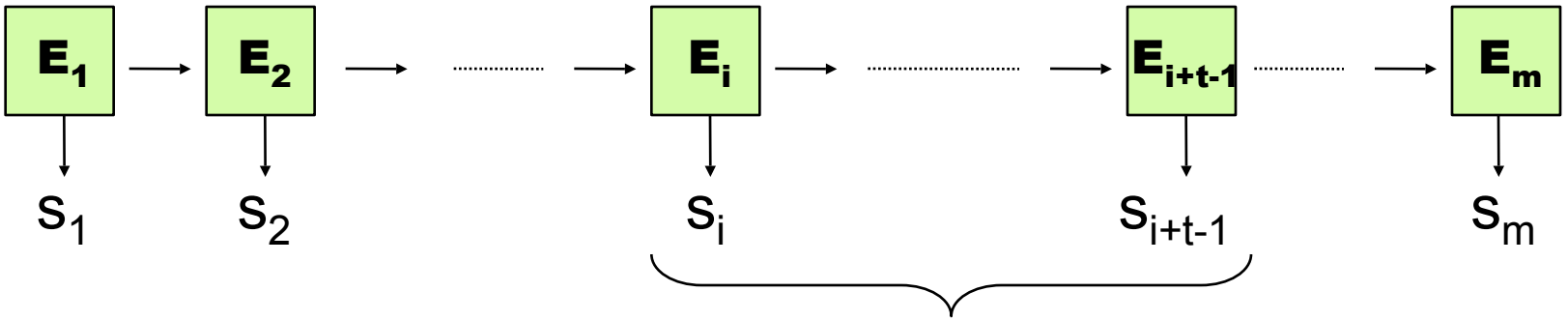
Compromis en $2^{t/2}$

Supposons que l'on connaisse m bits de sortie



Compromis en $2^{t/2}$

Supposons que l'on connaisse m bits de sortie



$\Phi(E_i)$ = *fenêtre de t bits*

Définissons la fonction :

$$\Phi: \{0,1\}^t \rightarrow \{0,1\}^t$$
$$E_i \rightarrow (s_i, \ldots, s_{i+t-1})$$

Compromis en $2^{t/2}$

- Φ associe à un état interne de t bits, la fenêtre des t prochains bits de pseudo-aléa générés
- Φ n'est pas forcément inversible !
- Toutefois, pour une fenêtre donnée, il n'y a en moyenne que 1 antécédent par Φ

Idée

- On précalcule l'image par Φ de $2^{t/2}$ états internes quelconques
- On connaît l'image par Φ de $m = 2^{t/2}$ états internes différents (pseudo-aléa connu)

$$2^{t/2} * 2^{t/2} = 2^{t}$$

- On s'attend donc à une collision (identifiable en regardant les sorties du GPA)

Compromis en $2^{t/2}$

- **Précalcul**
  - Choisir $2^{t/2}$ états internes $x_1, \ldots, x_{2^{t/2}}$ au hasard
  - Calculer $y_i = \Phi(x_i)$ pour tout les i
- **Stockage**
  - Stocker (tableau T) les paires (x_i, y_i) triées selon y_i
- **Attaque**
  - Pour i=1 … m, chercher à identifier la fenêtre de sortie connue $(s_i, \ldots, s_{i+t-1})$ avec un élément y_a de T
  - Retourner $E_i = x_a$

Analyse

- Temps de précalcul ≈ $2^{t/2}$ opérations
- Mémoire utilisée ≈ $2^{t/2}$ bits
- Temps de l'attaque ≈ $2^{t/2}$ opérations

Conclusion

- Il faut toujours ***prendre un état interne plus grand que le niveau de sécurité attendu***

- En général

$$t \geq 2 * k$$

Remarque

Autre compromis possible :

$$T = 2^{2t/3}$$

$$P = 2^{2t/3}$$

$$M = 2^{t/3}$$

avec $m = 2^{t/3}$ bits de sortie connus

Amélioration du compromis de Hellman

Attaques par corrélation

- Attaques génériques
  - Recherche exhaustive
  - Compromis Temps / Mémoire
- **Attaques par Corrélation**
- Attaques dédiées contre certains algorithmes

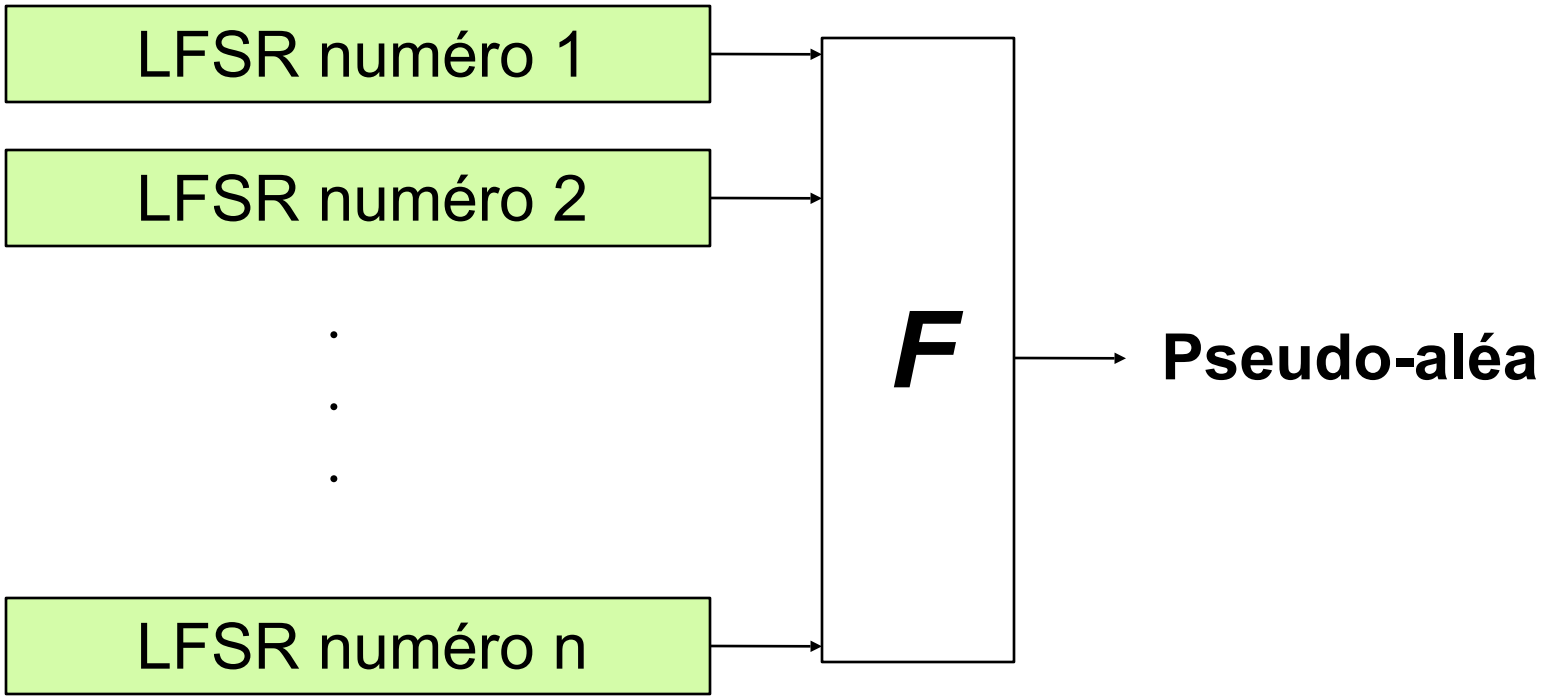
Attaques par corrélation

Attaques dédiées contre les algorithmes basés sur des Linear Feedback Shift Registers (LFSR)

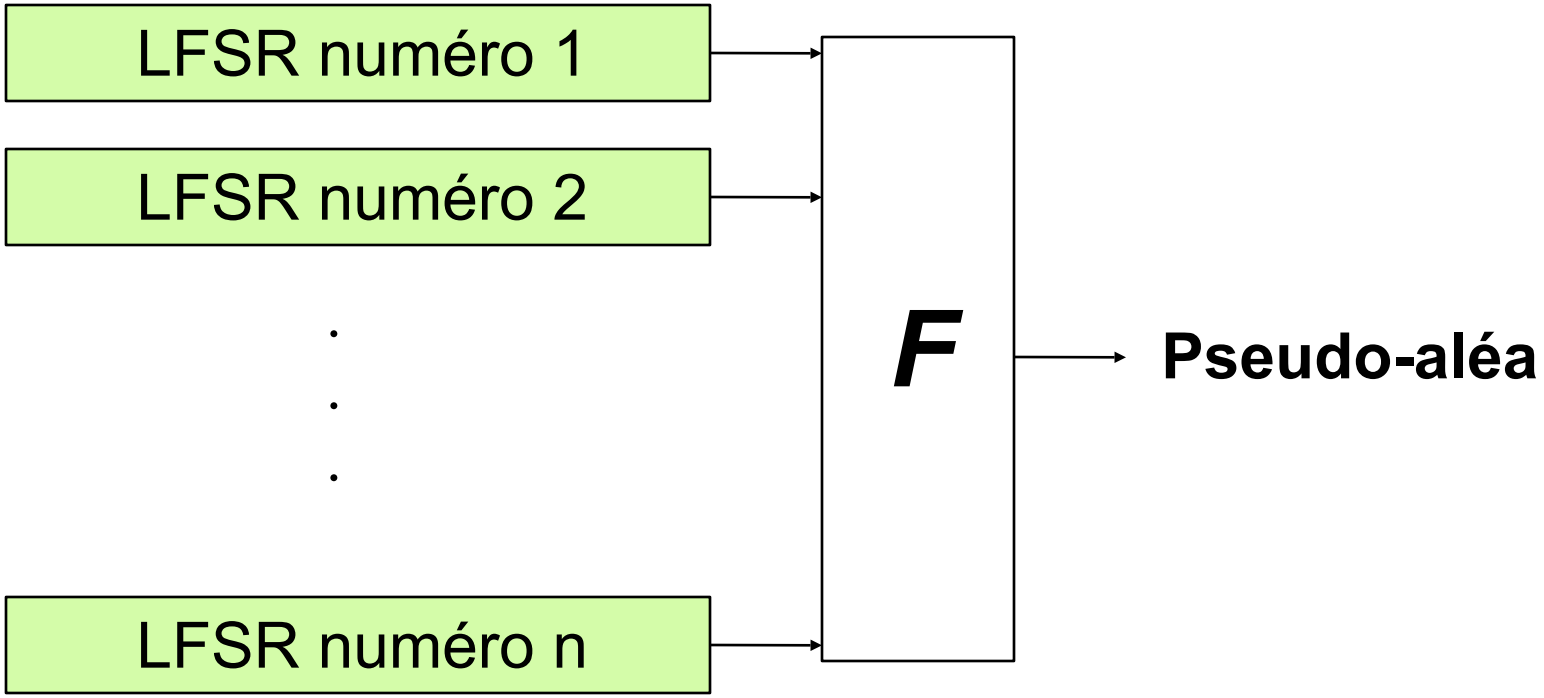
Les LFSR

- Linear Feedback Shift Register (LFSR)
- Registre à Décalage à Rétroaction Linéaire (RDRL)
- Très utilisés dans les algorithmes de chiffrement par flot

Combinaison

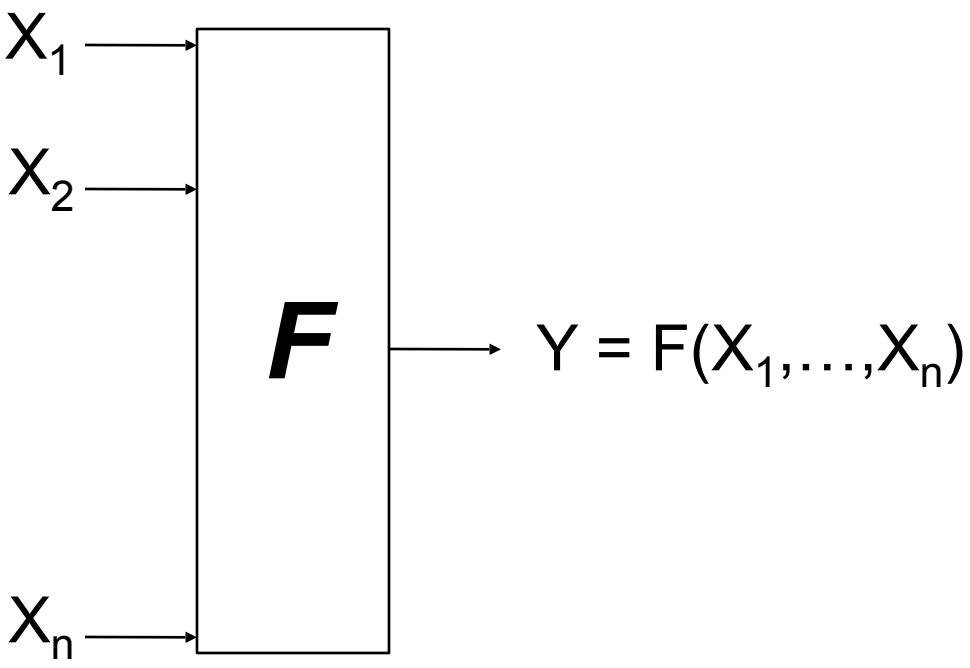


Combinaison



La fonction F n'est pas parfaite

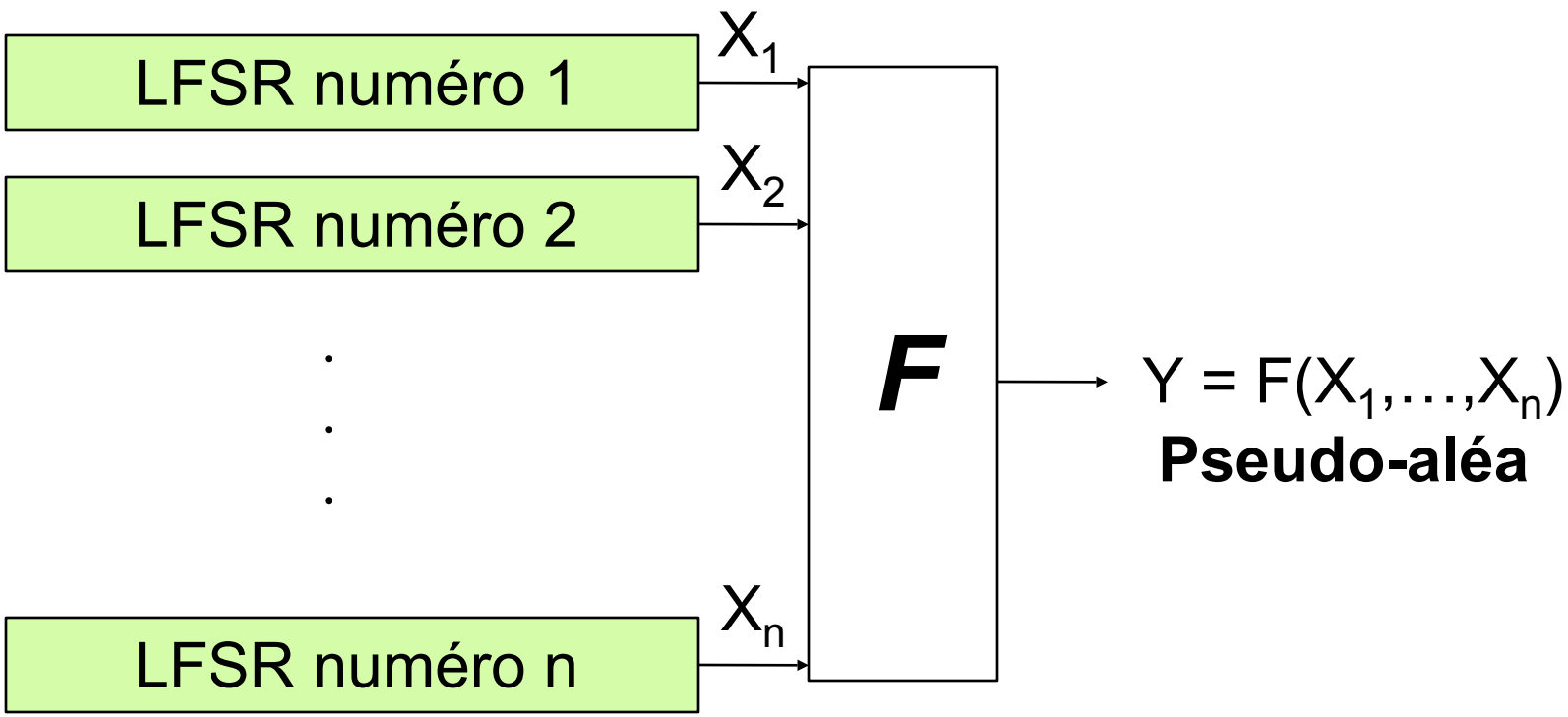
Corrélation ou Biais



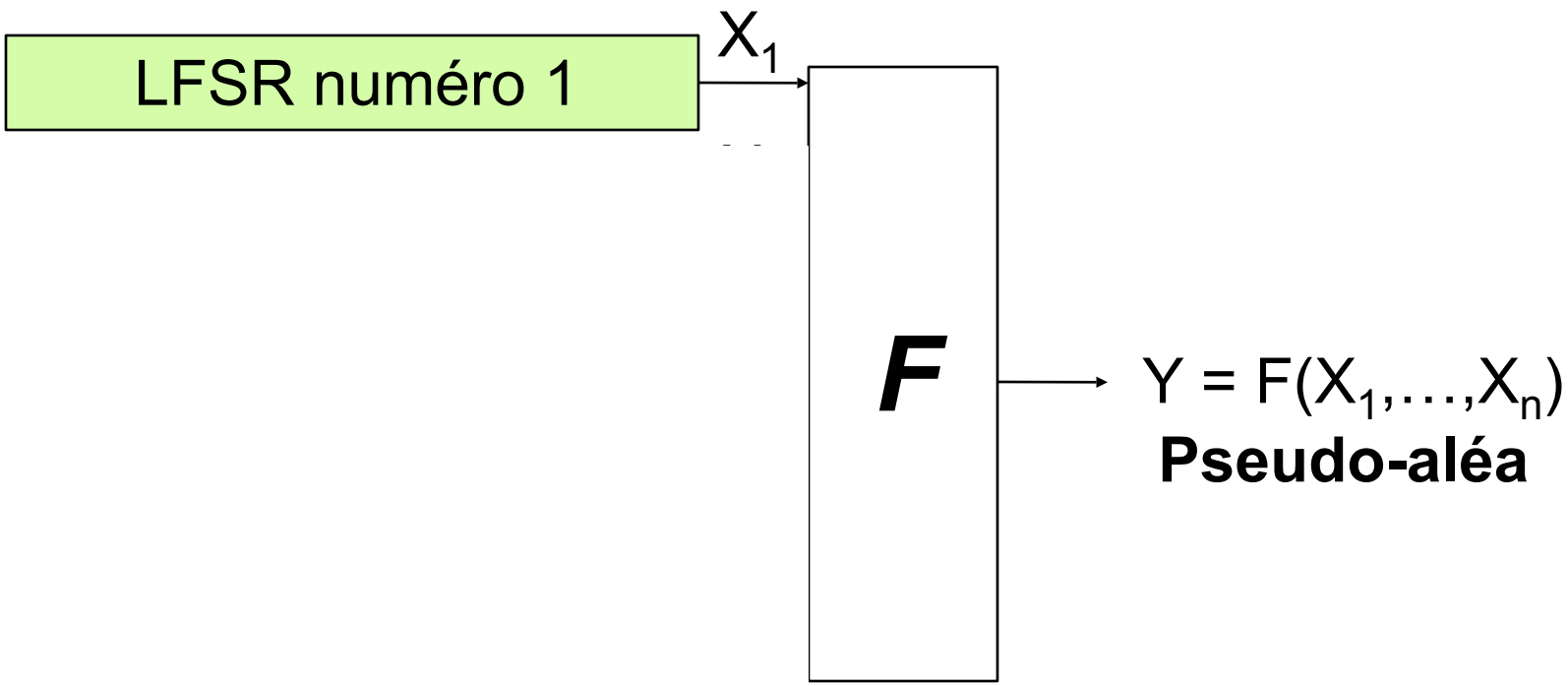
On suppose :

Proba[$Y = X_1$] = 0.5 + ε

Modélisation



Modélisation

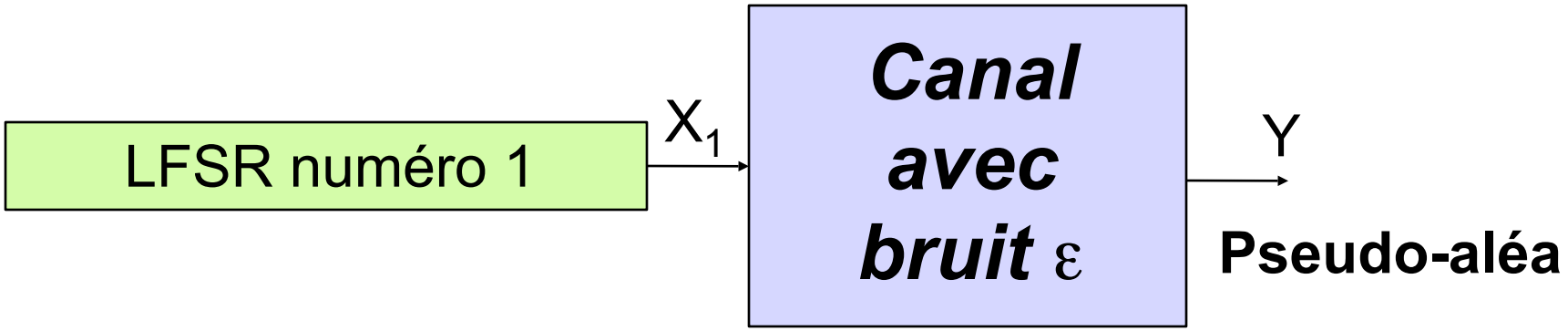


Canal bruité :

$Y = X_1$

avec probabilité $0.5 + \varepsilon$

Idée de l'attaque



Recherche exhaustive sur l'état du LFSR

On teste si les suites produites sont « corrélées » avec le pseudo-aléa observé

Attaque

- Taille de $LFSR_1$ = L bits (état initial E_0)
- On connaît les sorties (s_1, …, s_m) du GPA
- pour E_0 = 1 … 2^L
  - Calculer les sorties (s'_1, …, s'_m) du LFSR initialisé avec E_0
  - Calculer p = $\frac{1}{m}$ #{i tel que $s_i = s'_i$}

Attaque

- Taille de $LFSR_1$ = L bits (état initial E_0)
- On connaît les sorties (s_1, …, s_m) du GPA
- pour $E_0 = 1 \ldots 2^L$
  - Calculer les sorties (s'_1, …, s'_m) du LFSR initialisé avec E_0
  - Calculer $p = \frac{1}{m}$ #{i tel que $s_i = s'_i$}

    Bon E_0 $\quad p \approx 0.5 + \varepsilon$

    Mauvais E_0 $\quad p \approx 0.5$

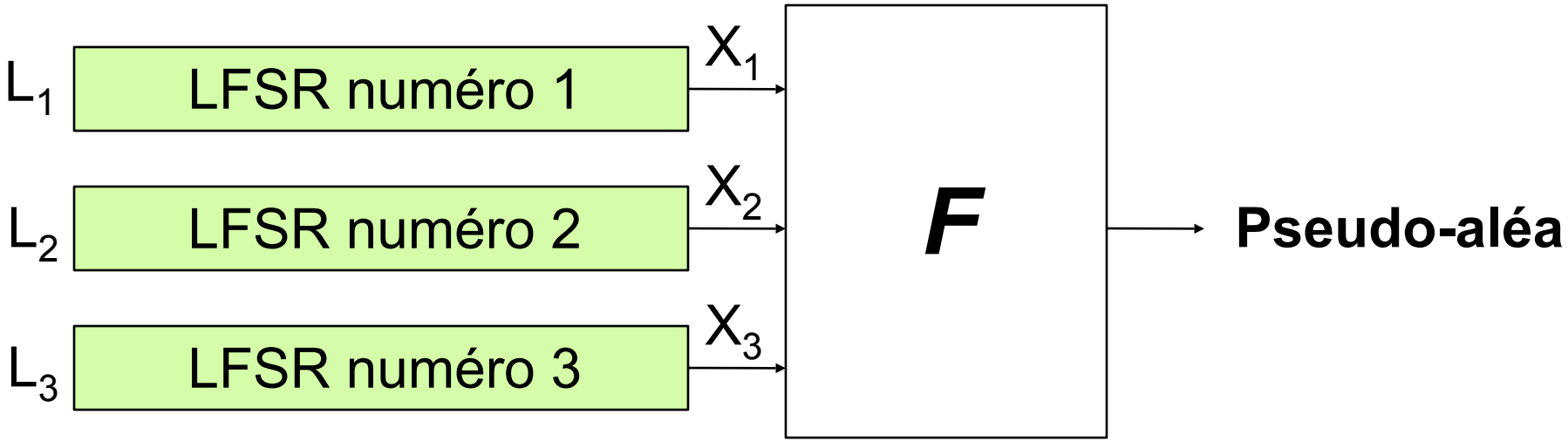
Analyse

- En pratique, il faut généralement $m > \varepsilon^{-2}$
- L'état initial E_0 fournit de l'info sur la clé
- On peut répéter l'attaque sur $LFSR_2$, etc …

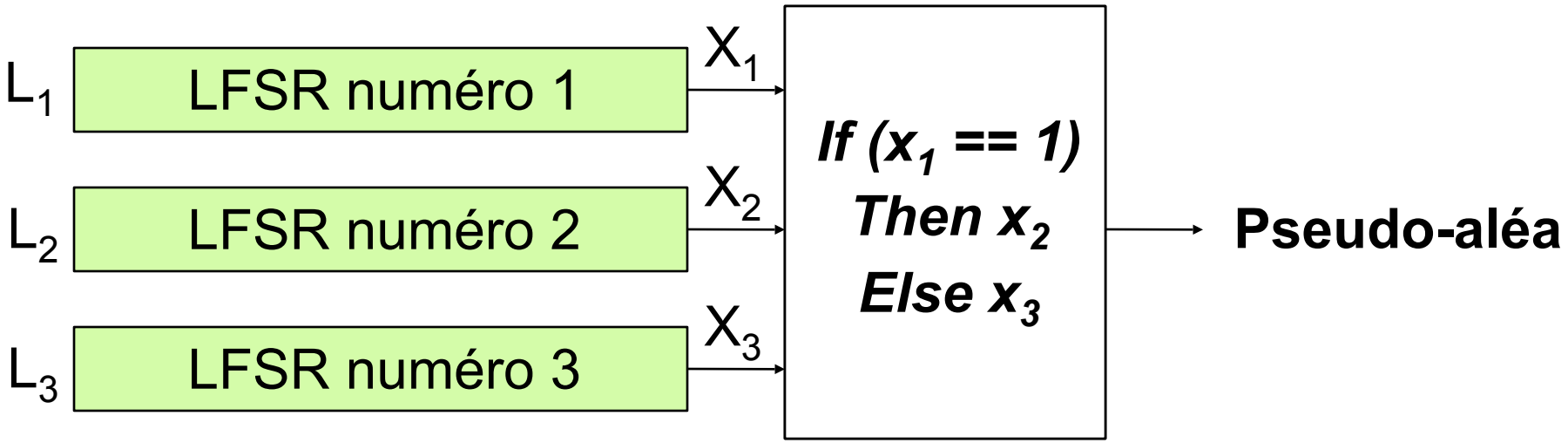
On parle d' ***attaque par corrélation***

$$T = m * 2^L$$

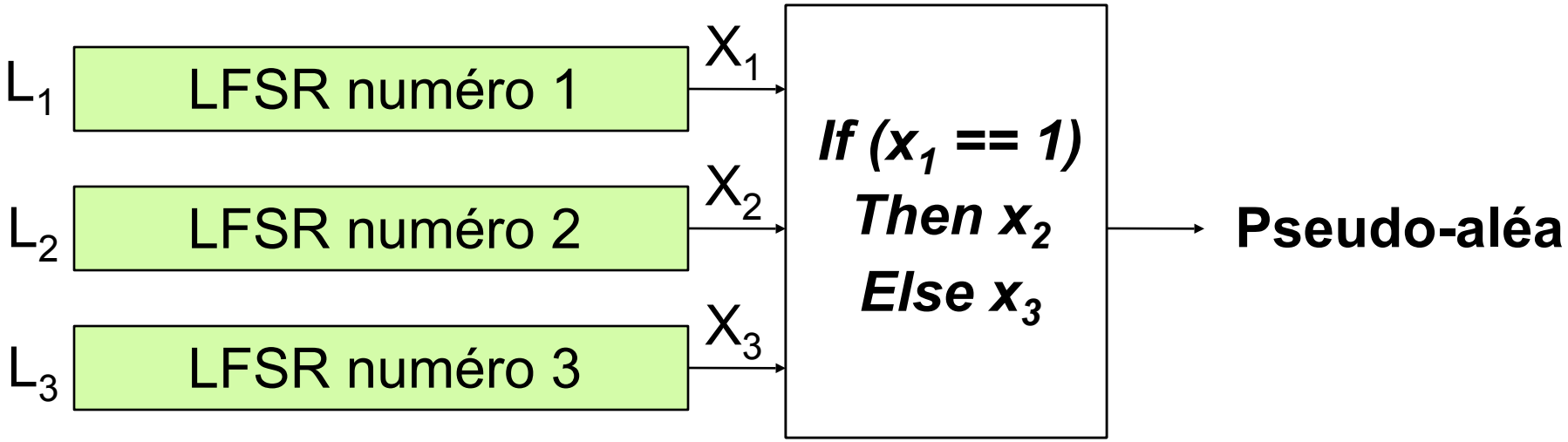
Exemple : Geffe



Exemple : Geffe



Exemple : Geffe



Quel biais a-t-on ?

Réponse

- Y = If(X_1) then (X_2) else (X_3)

  X_2 = Y avec probabilité ¾

  X_3 = Y avec probabilité ¾

- Donc on retrouve l'état initial du registre 2 (resp. 3) avec complexité 2^{L_2} (resp. 2^{L_3})
- Au total, attaque en

$$\mathbf{T = 2^{L_1} + 2^{L_2} + 2^{L_3}}$$

Extension

- Extension à

$$\text{Proba}[\text{C.L.}(X_1, \ldots, X_n) = Y] = 0.5 + \varepsilon$$

- Extension en ***attaque à chiffré seul*** quand le clair est biaisé Proba[$p_i = 0$] = $0.5 + \varepsilon'$

  On attaque à chiffré seul en utilisant le biais $\varepsilon.\varepsilon'/4$ (Piling-up lemma)

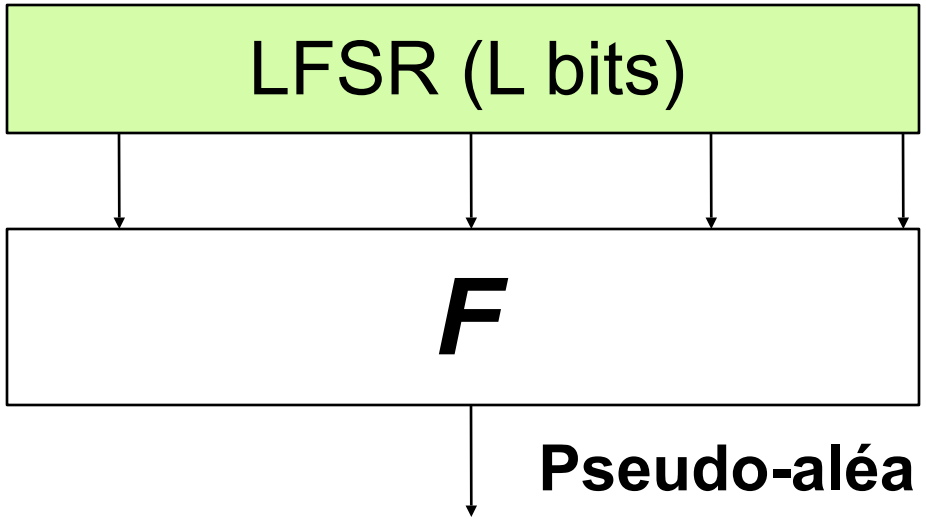
Correlation-immune

- Déf: X_1, X_2, ..., X_n n VA indépendante à valeur 0,1 avec proba. 1/2. $f(x_1,...,x_n)$ est m-order correlation-immune si $\forall$ sous-ensemble

  $X_{i1},..., X_{im}$ de m bits, $Z=f(X_1,...,X_n)$ est statistiquement indépendant de $(X_{i1},...,X_{im})$.
- Par exemple, le XOR est (n-1) immune.
- Si f est m-order immune (m<n), alors l'ordre non-linéaire de f est au plus n-m (degré f).

Attaques par Corrélation Rapides

- Lorsqu'on prend des registres suffisamment grands (de ***taille L ≥ 128 bits***), l'attaque précédente est trop coûteuse !
- Cas du Filtrage



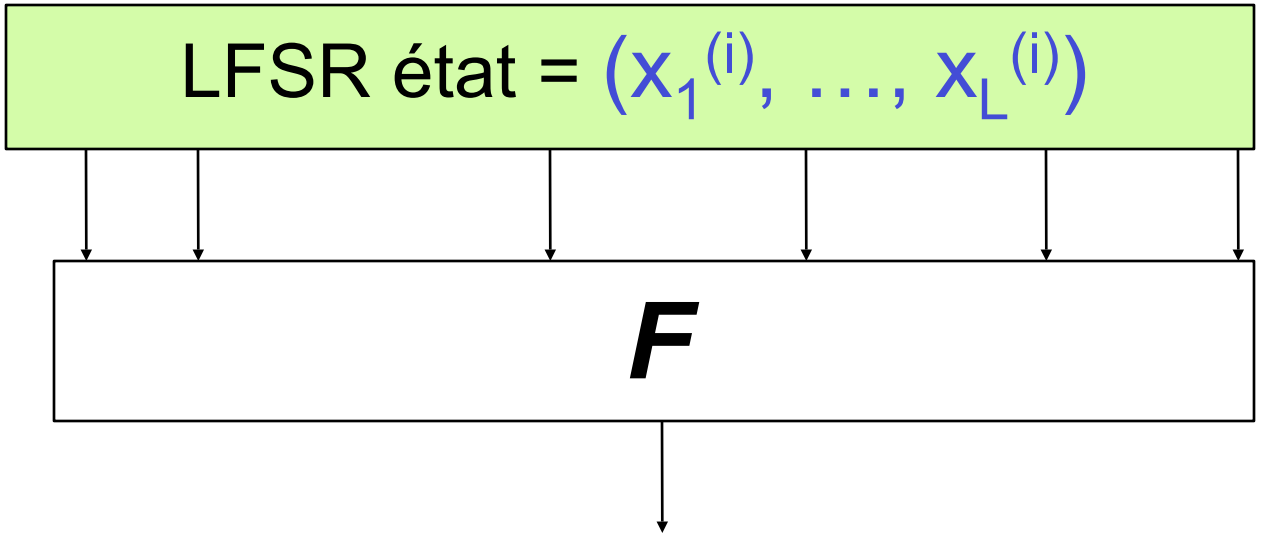
Modèle

- État initial du LFSR = clé = $(k_1, \ldots, k_L)$
- État à l'instant i = $(x_1^{(i)}, \ldots, x_L^{(i)})$
- Sortie du GPA = $z_i = F(x_1^{(i)}, \ldots, x_L^{(i)})$
- <u>Hypothèse</u> :F est « biaisée » :

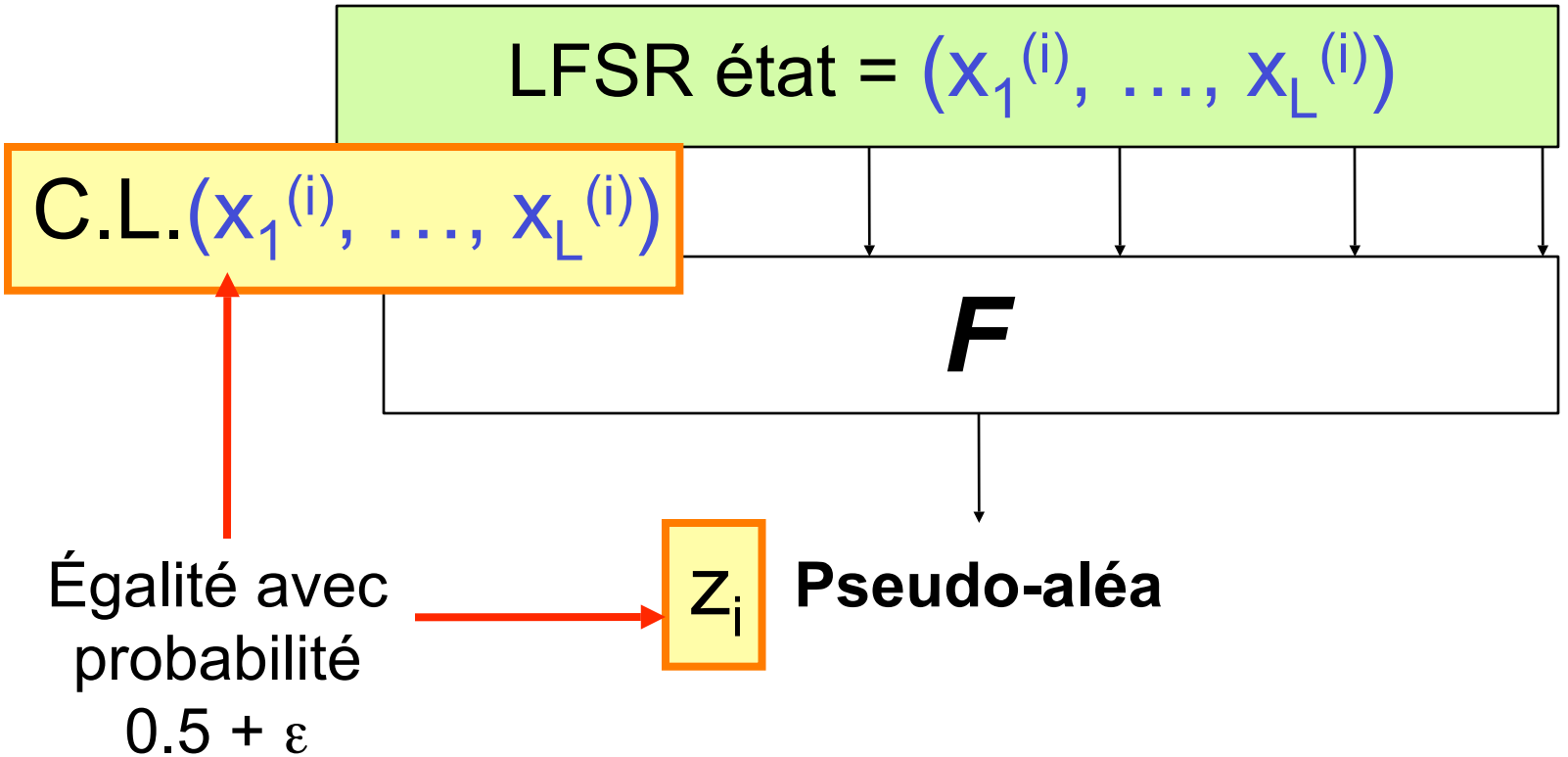
$\exists$ C.L. telle que :

$$\text{Proba}[\text{C.L.}(x_1^{(i)}, \ldots, x_L^{(i)}) = z_i] = 0.5 + \varepsilon$$

Modèle



Modèle


LFSR état = $(x_1^{(i)}, \ldots, x_L^{(i)})$
C.L.$(x_1^{(i)}, \ldots, x_L^{(i)})$
F
Égalité avec probabilité 0.5 + ε
z_i
Pseudo-aléa

Lien avec la clé

- Les $x_j^{(i)}$ dépendent **linéairement** de la clé

$$(x_1^{(i)}, \ldots, x_L^{(i)}) = \varphi^i(k_1, \ldots, k_L)$$

où φ est la fonction d'évolution du LFSR

- Proba[(C.L. $\circ$ φ^i) $(k_1, \ldots, k_L) = z_i$] = $0.5 + \varepsilon$

$\Rightarrow$ Équation linéaire biaisée

Fonction d'évolution

Équation de rebouclage du LFSR

$$x_1^{(t+1)} = \alpha_1 x_1^{(t)} + \ldots + \alpha_L x_L^{(t)}$$

$$x_i^{(t+1)} = x_{i-1}^{(t)} \text{ pour } i>1$$

$$\varphi = \begin{pmatrix} \alpha_1 & \alpha_2 & \ldots & \alpha_{L-1} & \alpha_L \\ 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 1 & 0 \end{pmatrix}$$

Équations

$i=0$ $\quad (C.L. \quad) \quad (k_1, \ldots, k_L) = z_0 \quad$ proba = $0.5+\varepsilon$

$i=1$ $\quad (C.L. \circ \varphi^1) \quad (k_1, \ldots, k_L) = z_1 \quad$ proba = $0.5+\varepsilon$

...

$i=m$ $\quad (C.L. \circ \varphi^m) \quad (k_1, \ldots, k_L) = z_m \quad$ proba = $0.5+\varepsilon$

Comment exploiter ces équations ?

Décodage

- « résoudre » un système d'équations linéaires « biaisées »
- Proche des problèmes de décodage
- Problème difficile dans le cas général

Résolution

- Idée : construire des mots (i.e. des combinaisons d'équations) de poids faible

- Appliquer un algorithme de décodage pour prédire la valeur de chaque bit de clé

Parity Checks

- Combinaison des équations t.q.

$$\Sigma_i \, a_i \, (\text{C.L.} \circ \varphi^i)(k_1, \ldots, k_L) = k_1$$

- Dans ce cas on montre que

$$\Sigma_i \, a_i \, z_i = k_1$$

avec probabilité $0.5(1 + (\varepsilon/2)^h)$

$h = \#\{a_i = 1\}$

« Évaluation de k_1 grâce aux sorties »

Attaque

- Attaque en plusieurs phases
  - Construire M parity checks (phase indépendante du pseudo-aléa observé)
  - Observer le pseudo-aléa $\{z_i\}$ produit par le GPA
  - Évaluer les M parity checks grâce aux z_i
  - La valeur majoritaire est la valeur de k_1

Parity checks

- Recherche de multiples creux (h petit) du polynôme de rebouclage.
- Très coûteux mais plus facile si le polynôme est déjà creux
- Il faut en obtenir M > $(\varepsilon/2)^{-2h}$

Synthèse

- Les attaques par corrélation sont un outil puissant contre les LFSR

- L'étude des propriétés de la fonction F sont essentielles dans ces constructions

Autres Attaques

- Attaques génériques
  - Recherche exhaustive
  - Compromis Temps / Mémoire
- Attaques par Corrélation
- **Attaques dédiées contre certains algorithmes**

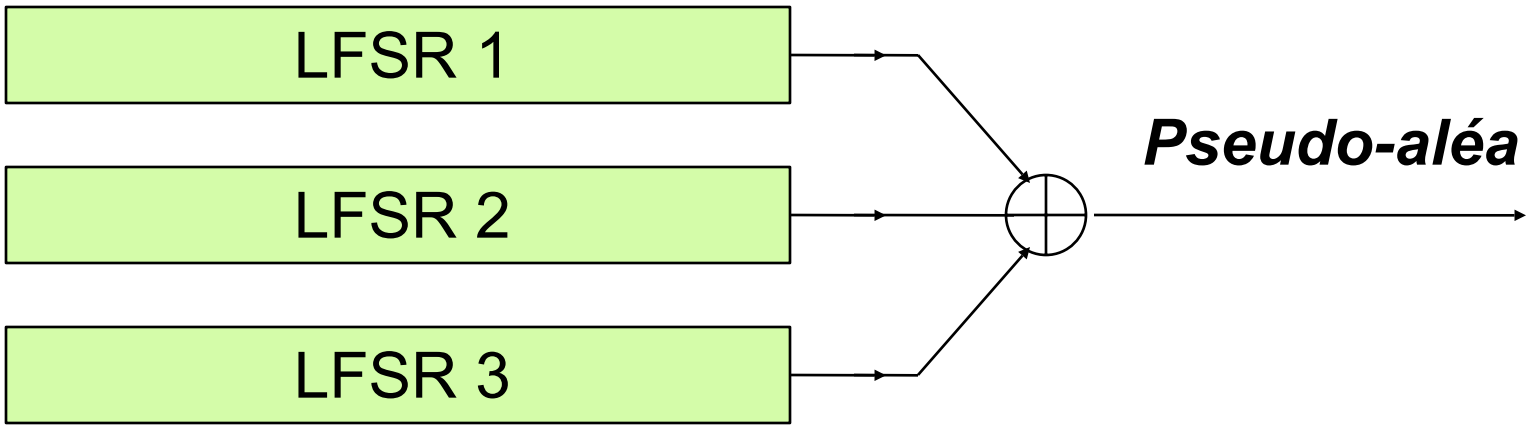
A5/1 et A5/2

- Norme de téléphonie GSM
- Algorithmes utilisés pour le chiffrement
  - A5/1 : stream cipher
  - A5/2 : stream cipher
  - A5/3 : block cipher KASUMI en mode OFB
- A5/1 et A5/2 longtemps restés confidentiels (jusqu'en 1999)

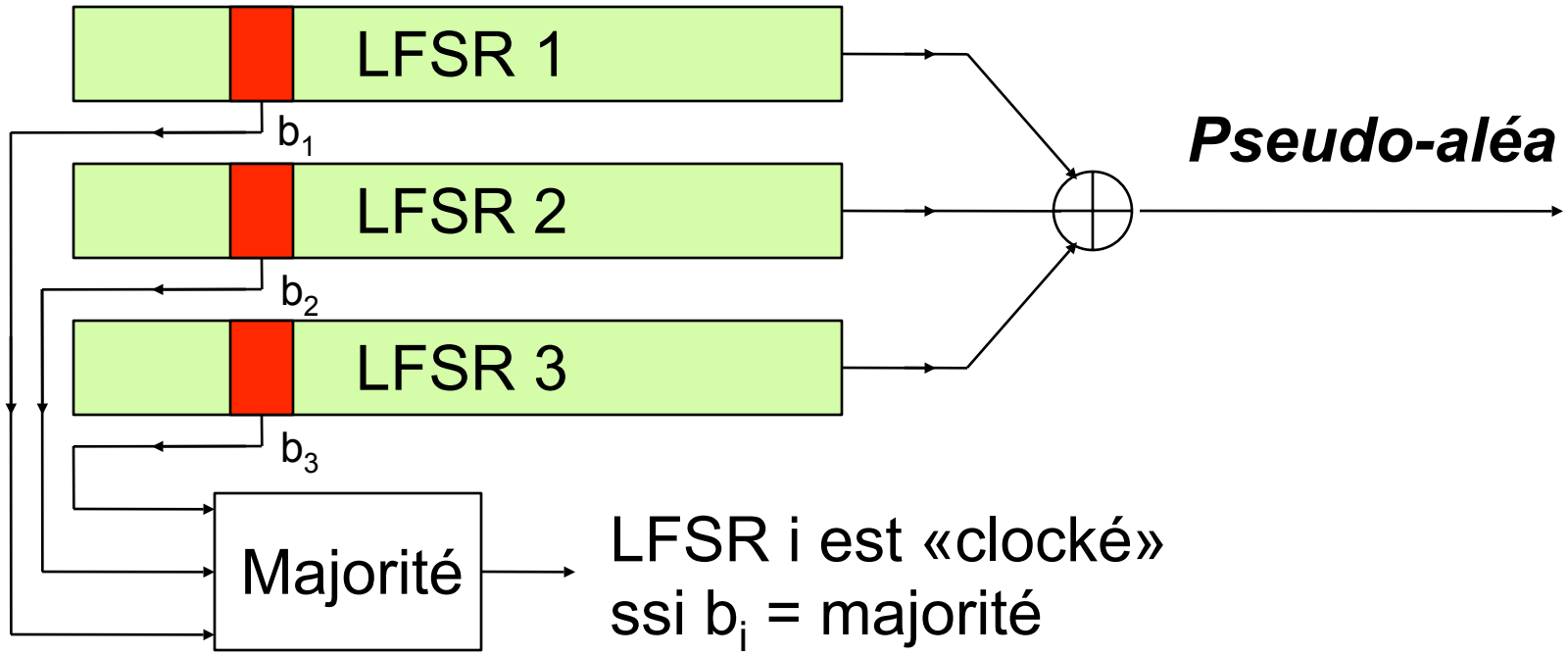
A5/1

- Algorithme minimaliste (contrainte de surface matérielle)
- Taille de clé = 64 bits
- État interne = 64 bits
- Combinaison de LFSR avec contrôle de l'horloge

A5/1



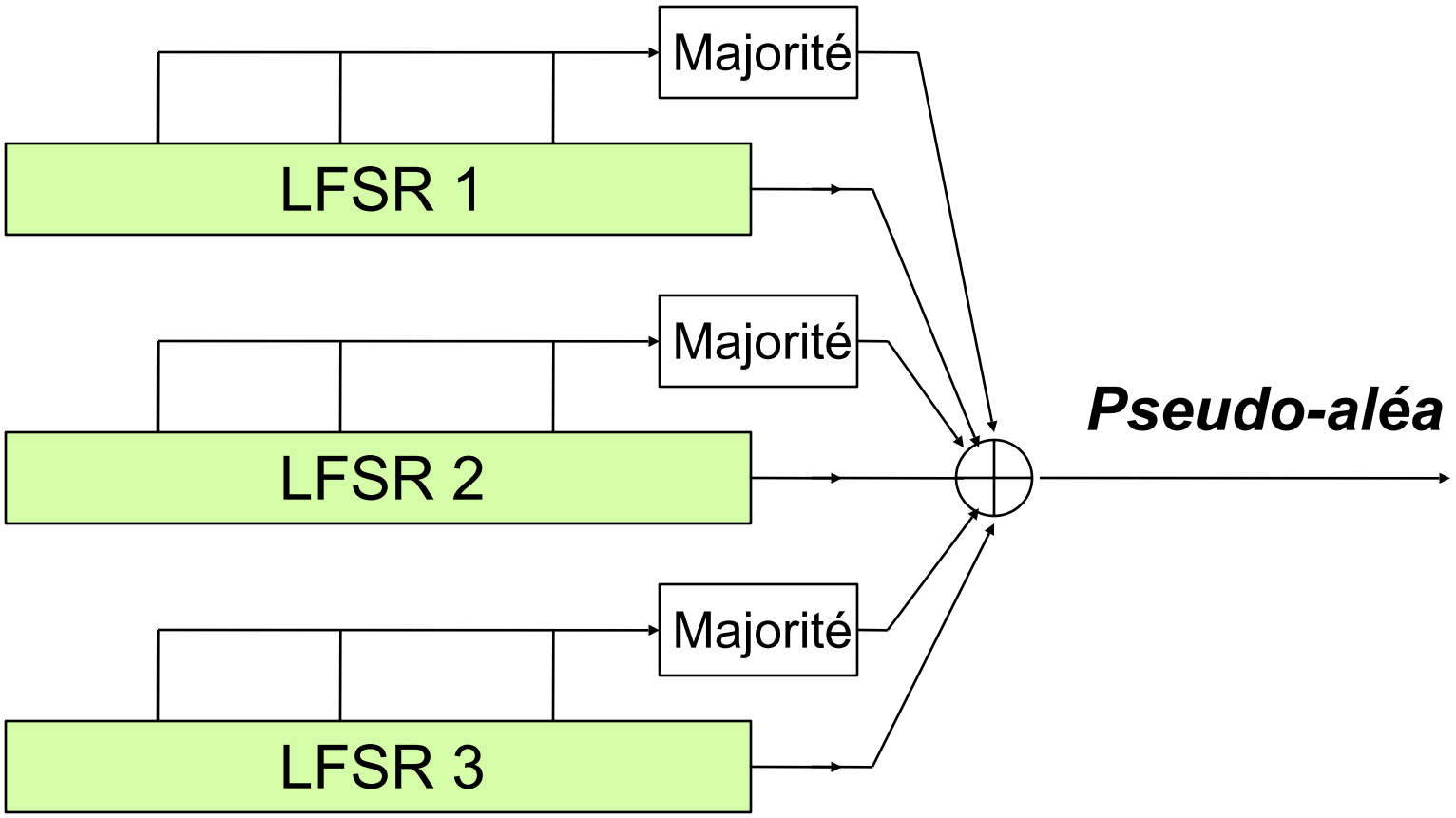
A5/1


LFSR 1
LFSR 2
LFSR 3
b1
b2
b3
Majorité
Pseudo-aléa
LFSR i est «clocké»
ssi bi = majorité

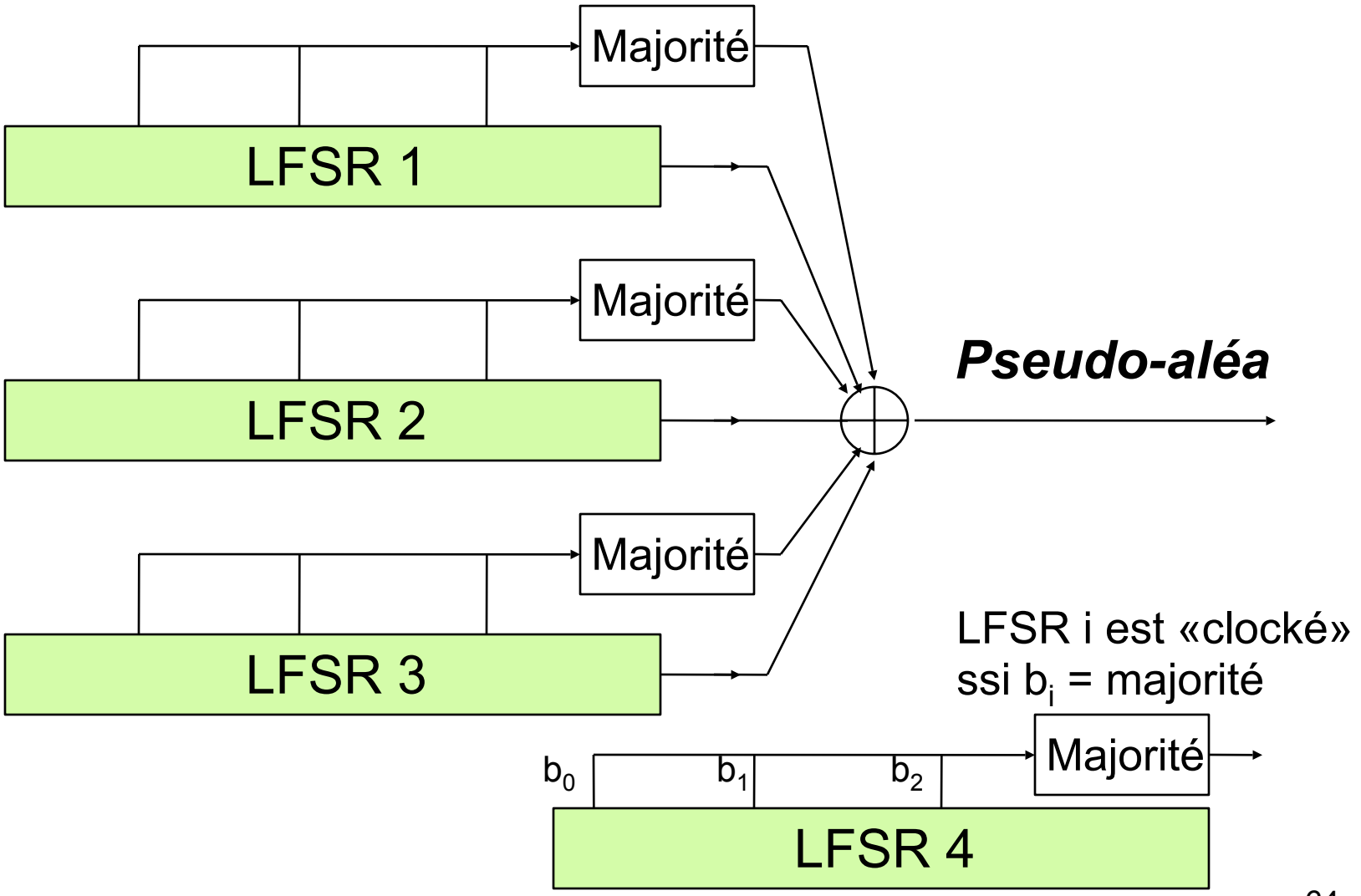
Attaques contre A5/1

- Compromis Temps/Mémoire
  - $T \approx 2^{42}$
  - $M \approx 2^{21}$
- Reconstruction de l'état interne
  - Recherche exhaustive sur 10 bits/registre
  - Prédiction des sorties pour quelques tours
  - Algèbre linéaire pour retrouver la clé
- Attaques réalisables en temps réel ?

A5/2



A5/2


Majorité
LFSR 1
Majorité
LFSR 2
Majorité
LFSR 3
Pseudo-aléa
LFSR i est «clocké»
ssi b_i = majorité
b_0
b_1
b_2
Majorité
LFSR 4

A5/2

- État interne plus grand que A5/1 (81 bits)
- Calcul non-linéaire des sorties (F de degré 2)
- Registre auxiliaire de contrôle
  - Faiblesse potentielle ?
  - Structure plus simple

Cryptanalyse de A5/2

- Recherche exhaustive sur l'état initial de LFSR 4
  - Complexité 2^{16}
  - Toutes les avances sont connues

- Les sorties du GPA sont des fonctions quadratiques de la clé (Majorité)

Attaque algébrique

- Clé = $k_1, \ldots, k_{64}$
- Sorties du GPA = polynôme en les k_i :
  - Monômes de degré 0 : 1
  - Monômes de degré 1 : $k_1, \ldots, k_{64}$
  - Monômes de degré 2 : $k_i * k_j$
- Nombre de monômes N

$$N = 1 + 64 + (64*63)/2 = 2081$$

Attaque algébrique

- En fait, on a N = 656 monômes
- Linéarisation :
  - Chaque monôme = nouvelle variable
- 656 bits de sortie du GPA connus
- Système linéaire
  - 656 équations
  - 656 inconnues

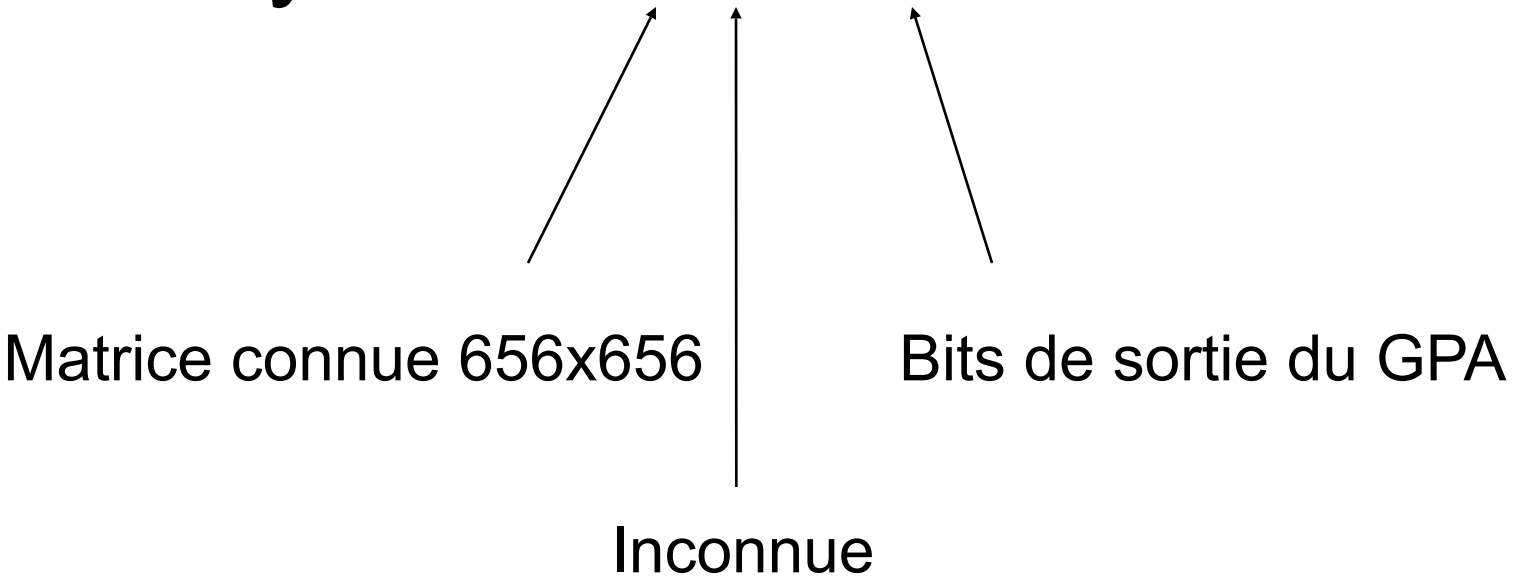
Synthèse

- La résolution coûte environ $656^3 \approx 2^{27}$ opérations (algorithme de Gauss)
- Au total

$$T = 2^{16} * 2^{27} \approx 2^{43}$$

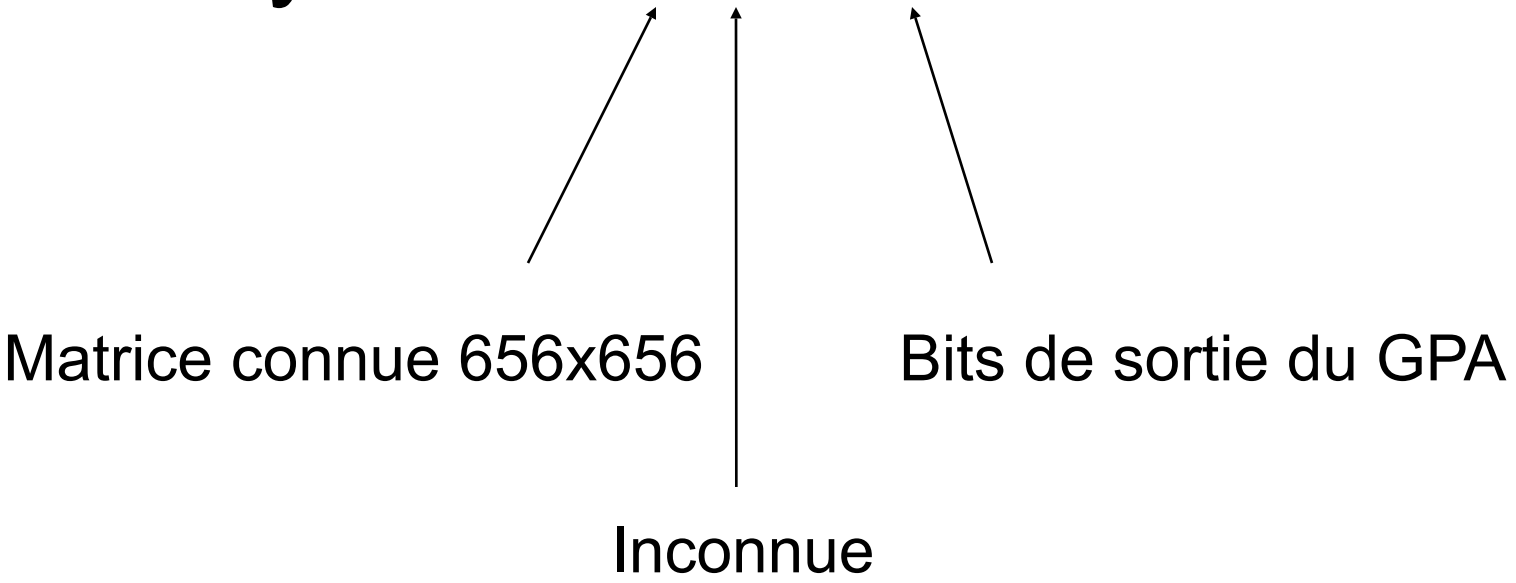
Amélioration

Système Ax = b



Amélioration

Système Ax = b



⇒ ***Précalculer la matrice inverse A^{-1}***

En pratique

- Attaque quasi instantanée
  - < seconde
  - Précalcul de 4 heures sur un PC
- Possibilité de changer d'algorithme
  - mauvais protocole de négociation
  - Clé A5/2 = Clé A5/1 = Clé A5/3
- Attaque à chiffré seul
  - clair connu grâce au code correcteur CRC

Autres Cryptanalyses

- Attaques algébriques
  - Linéarisation
  - Bases de Gröbner
- Attaques pour les LFSR avec contrôle de l'horloge
- Attaques contre RC4

RC4

- S: état interne de 256 octets et i, j: 2 indices=octet
- KSA:
  - initialiser S[i]=i
  - for i=0 à 255 do
    - j=j+S[i]+K[i mod k] mod 256
    - swap(S[i],S[j])
  - j=0
- PRGA:
  - i=i+1 mod 256, j=j+S[i] mod 256
  - swap(S[i],S[j]) et return S[S[i]+S[j] mod 256]

Finney states

- S grand et PRGA a de grands cycles
- Existe petite classe d'états internes avec période 256*255: j=i+1 et S[j]=1
- Jamais atteint par le KSA

0	1	2	3
	1 ^{j}	X	Y
	X ^{i}	1 ^{j}	Y
	X	Y ^{i}	1 ^{j}

Output: S[X+1]

Output: S[Y+1]

Biais statistique

- Le second octet de sortie est biaisé: 0 avec proba ≈ 2^{-7} au lieu de 2^{-8}
- Si l'état après KSA: S[2]=0 et S[1]≠2
- Pr[z_2=0]=1/256+1/256*(1-1/256)≈$1/2^7$

0	1	2	X
i j	X j	0	Y
	Y i	0	X j
	Y	X i	0 j

Output: S[X+Y]

Output: S[X+0]=0

IV mode et WEP

- RC4 avec IV
- KSA: i croît de façon régulier et j aléatoir
- Une fois que i a avancé, avec forte proba. les premiers éléments de S ne sont pas changés
- Premier octet est S[S[1]+S[S[1]]] et dépend de S[1] et S[S[1]]
- Comment prédire ces 2 valeurs ?

WEP Attack

- IV de 3 mots ajouté avant la clé secrète
- On connaît les A premiers octets K[3],...,K[A+2] et IV forme (A+3,-1,X) pour X aléatoire

0	1	2	A+3
0 i j	1	2	A+3
A+3 i	1	2	0 j
	0 i	2	1 j

S[1]=0; S[S[1]]=A+3

- On peut calculer les A+3 premières étapes et on jette si S[1] ou S[0] est modifié

WEP attack (suite)

- L'étape suivante i=A+3 et S[A+3] dépend seulement de K[A+3]
- Si ces 3 éléments ne sont pas modifiés par KSA, le premier octet de sortie donne suffisamment d'info pour calculer K[A+3]
- Pour protéger de cette attaque, équipement WIFI vérifie les IVs de la forme (A+3,-1,X)
- Ceci non suffisant car tout IV tq après 3 étapes S[1]+S[S[1]]=A+3 permet d'attaquer

Toyocrypt

- LFSR de 128 bits et fonction de filtrage:
  $f(s_0,\ldots,s_{127})=s_{127}+\sum_{i=0}^{62} s_i s_{\alpha i}+s_{10}s_{23}s_{32}s_{42}$
  $+s_1s_2s_9s_{12}s_{18}s_{20}s_{23}s_{25}s_{26}s_{28}s_{33}s_{41}s_{42}s_{51}s_{53}s_{59}$
  $+\prod_{i=0}^{62} s_i$ avec $\alpha i \in \{63,\ldots,125\}$.
- Trouver g tq fg de petit ordre: $g(s)=(s_{23}-1)$ ou $g(s)=(s_{42}-1)$ et les équations $f(s)=b_t$ deviennent $f(s)s_{23}-f(s)=b_t(s_{23}-1)$. ($s_{23}s_{42}$ divisent les monômes de degré 4, 17 et 63.)

Toyocrypt (suite)

- $f(s)s_{23}-f(s)=s_{127}s_{23}+s_{23}\sum_{i=0,\neq 23}^{62} s_i s_{\alpha i}=b_t(s_{23}-1)$
- Equations de degré 3 vraie avec proba 1.
- Même chose pour 42.
- Pour chaque bit de sortie, on a 2 équations de degré 3 en les bits de la clé k_i (état du LFSR).
- Relinéarisation: C_{128}^3 monômes de degré 3
- $\approx 2^{18.4}$ et si on a cette quantité de bits, syst. linéaire à résoudre en temps 2^{49}. car on a 2 eq.

Synthèse

- De nombreuses techniques de cryptanalyse pour les stream ciphers
  - Attaque générique
  - Attaque dédiées contre certains algorithmes

- Peu de primitives sûres (projet européens NESSIE, ECRYPT)