Générateurs de nombres aléatoires

Pseudorandom number generator (PRNG)

Introduction

Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin.

John von Neumann (1903-1957)

Introduction

Attendus concernant les générateurs de nombres aléatoires ?

Référentiel Général de Sécurité (RGS) ANSSI !!!

Introduction

Exemple de certification ANSSI

https://www.ssi.gouv.fr/uploads/IMG/certificat/ANSSI-CC_2014-75fr.pdf

Déterministe

En informatique, une fonction déterministe est une fonction qui, pour les mêmes arguments, renverra toujours le même résultat

WIKIPÉDIA
L'encyclopédie libre

Chiffrement par flot/flux

stream cipher

- Une des deux grandes catégories de chiffrements en **cryptographie symétrique**
- Traiter les données de longueur quelconque, pas besoin de les découper.
- Un générateur de nombres pseudo-aléatoires puis un XOR entre un bit à la sortie du générateur et un bit provenant des données.
- XOR n'est pas la seule opération possible: addition entre deux octets modulo 256,...

Chiffrement par flot/flux

Comparaison avec les chiffrements par blocs

Chiffrement par flot	Chiffrement par bloc en mode CFB ou OFB
1 bit à la fois	blocs de bits
débit élevé	débit moyen
bien adapté à une implémentation “hard”	”soft”
implémentation spécifique	ré-utilisation de l’existant

Utilisation

Essentiellement pour des communications sans fil (GSM, Wifi, Bluetooth, ...) à cause du besoin de débit élevé.

Théorie de l’information

- Théorie probabiliste permettant de quantifier le contenu moyen en information d'un ensemble de messages,
- Ce domaine trouve son origine scientifique avec Claude Shannon avec son article A Mathematical Theory of Communications publié en 1948.

Théorie de l'information

- Parmi les branches importantes de la théorie de l'information de Shannon, on peut citer :
  - le codage de l'information,
  - la mesure quantitative de redondance d'un texte,
  - la compression de données,
  - la cryptographie.

Entropie

- Quantité d'information contenue ou délivrée par une source d'information.
- Du point de vue d'un récepteur, plus la source émet d'informations différentes, plus l'entropie (ou incertitude sur ce que la source émet) est grande.
- Si une source est réputée envoyer toujours le même symbole, par exemple la lettre 'a', alors son entropie est nulle

Entropie

- L'entropie indique alors la quantité d'information nécessaire pour que le récepteur puisse déterminer sans ambiguïté ce que la source a transmis.
- Plus le récepteur reçoit d'information sur le message transmis, plus l'entropie (incertitude) vis-à-vis de ce message décroît.

$$H = -\sum p(i) \log_2 p(i)$$

Tests statistiques

- Il est assez difficile de dire si une suite est ou non aléatoire.
- On ne peut pas dire qu'un générateur est biaisé à la vue d'une suite qui ne paraîtrait absolument pas aléatoire, car cette suite doit bien apparaître avec une certaine fréquence.

Tests statistiques

- Seul le fait que la suite apparaisse avec une mauvaise fréquence (trop forte ou trop faible) permet de dire que le générateur possède un défaut.
- **On ne sait pas parfaitement caractériser le hasard.**

Tests statistiques

- Il faut retenir qu'un générateur peut toujours réussir *n* tests et échouer au *n+1* ieme

Tests statistiques

- Pour vérifier qu'un générateur (dont les nombres obtenus sont des entiers compris entre 1 et 6) se comporte comme un dé à 6 faces, on procède à un test sur la fréquence d'apparition de chaque nombre, celle-ci doit s'approcher de 1/6 grâce au test du χ^2 qui s'applique aux distributions discrètes.
- Pour tester qu'une distribution continue suit bien une loi normale ou exponentielle ou toute autre loi, on utilisera alors le test de Kolmogorov-Smirnov.

Tests statistiques

- NIST :http://csrc.nist.gov/groups/ST/toolkit/rng/batteries_stats_test.html
- http://www.hsc.fr/ressources/articles/nombre/index.html.fr

Générateurs pseudo-aléatoires

- Les générateurs pseudo-aléatoires fonctionnent de la manière suivante :
    - le générateur possède un état interne,
    - chaque appel au générateur modifie l'état interne et renvoie une valeur qui dépend uniquement de l'état interne.

- Un tels générateur n'est pas vraiment aléatoire car les valeurs générées dépendent uniquement de la valeur initiale de l'état interne.

Générateurs pseudo-aléatoires

- Cet valeur initiale de l’état interne est aussi appelé graine (seed en anglais) et ne peut en pratique prendre qu'un nombre fini (mais parfois très grand) de valeurs.

- À cause de ceci, si on génère suffisamment de valeurs pseudo-aléatoires, le générateur se retrouve forcément dans un état déjà visité et les valeurs générées se répètent.

Générateurs pseudo-aléatoires

• Une suite infinie de nombres aléatoires est nécessairement périodique à partir d'un certain rang (« en rho » [1]):

$$\rho$$

[1] : « Simulation à événements discrets », G. Fleury, P. Lacomme, A. Tanguy

Retraitement algorithmique

- Générateurs linéaires congruentiels

$$z_n = (az_{n-1} + b) \bmod m$$

où m est public et a, b, z_0 (la graine) forment la clé. Si a, b et m sont choisis correctement, le générateur sera de période maximale (m).

Avantages : rapides car peu d'opérations, bonne répartition
Inconvénient : cryptographiquement mauvais

Utilisation non cryptographique

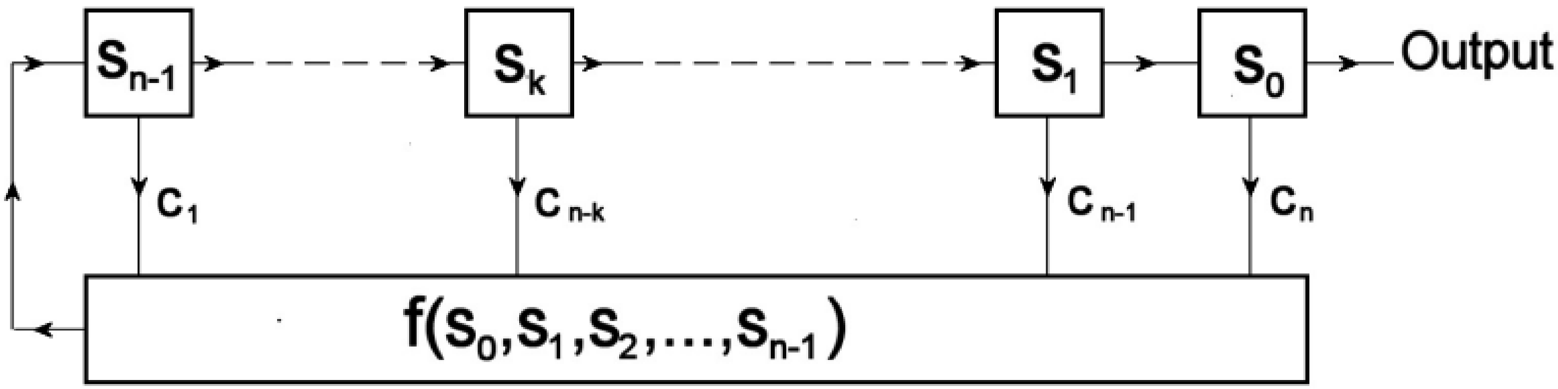
Combinaison de générateurs linéaires (ou polynomiaux) congruentiels

- meilleur comportement statistique
- plus grandes périodes
- pas plus sûrs

Registre à décalage à rétroaction

Composé de deux éléments

- un registre à décalage
- une fonction de rétroaction f



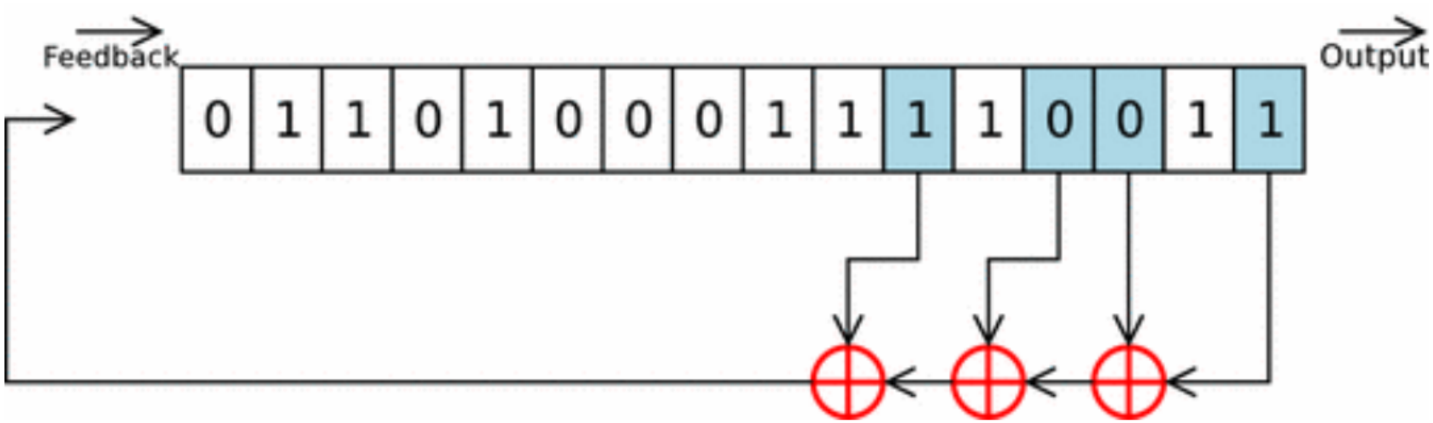
A chaque étape (top d'horloge), s_0 est retourné, tous les autres bits sont décalés vers la droite et s_n est calculé par la fonction de rétroaction f.

Retraitement algorithmique

- Registre à décalage à rétroaction linéaire (LFSR)

$$s_n = f(s_0, \ldots, s_{n-1}) = \sum_{i=1}^{n} c_i s_{n-i}.$$

Par exemple



Il y a 2^n possibilités d'initialisation (la clé)

Retraitement algorithmique

$\rightarrow$ période maximale $T = 2^n - 1$

Dans le cas de la période maximale, on parle de m-suite.

On appelle polynôme de rétroaction le polynôme de $\mathbb{F}_2[X]$

$$F(X) = 1 + c_1X + c_2X^2 + \cdots + c_nX^n$$

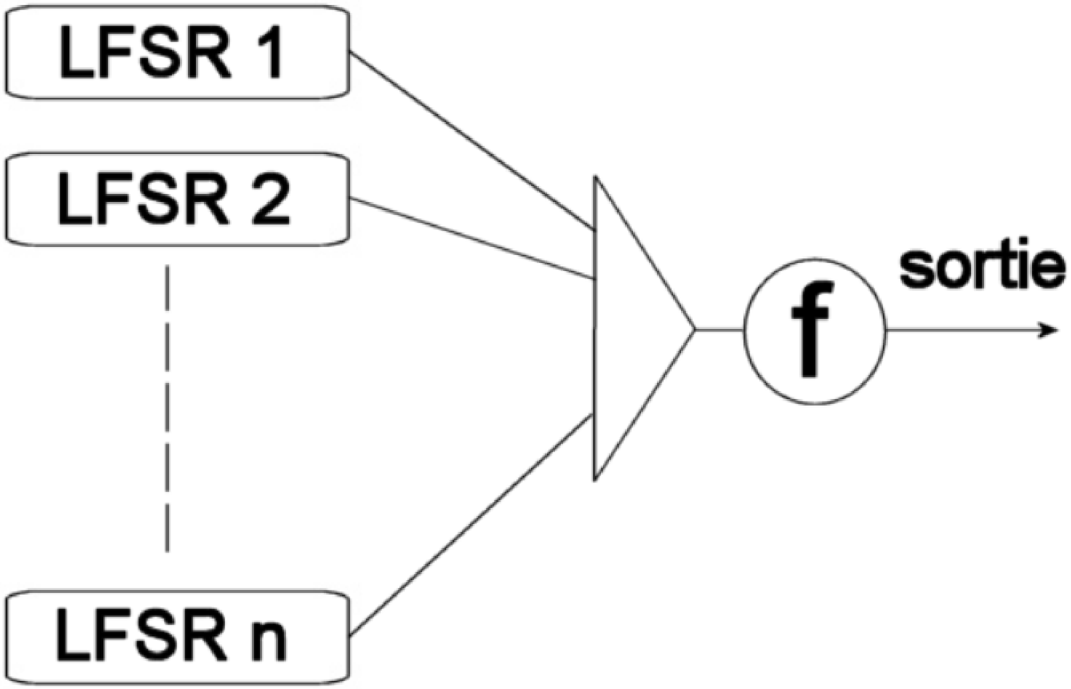
Theorem

La suite est une m-suite si son polynôme de rétroaction est le polynôme minimal d'un générateur du groupe cyclique $\mathbb{F}_{2^n}^*$ (c.a.d. le polynôme est primitif), autrement dit si

- F *est irréductible*
- F *divise* $X^{2^n-1} + 1$
- F *ne divise pas* $X^d + 1$ *si* $d | 2^n - 1$

Générer un polynôme primitif est un problème difficile.

Combinaison de LFSR



avec f non linéaire. f et les polynômes de rétroaction sont publics. La clé est l'état initial des LFSR.

Combinaison de LFSR

Exemple : générateur de Geffe

- 3 LFSR de longueurs premières entre elles 2 à 2.
- $f(x_1, x_2, x_3) = x_1x_2 \oplus x_1x_3 \oplus x_2$
- période $(2^{L_1} - 1)(2^{L_2} - 1)(2^{L_3} - 1)$
- complexité linéaire : $L_1L_2 + L_1L_3 + L_2$

Combinaison de LFSR

On exploite une éventuelle corrélation entre la sortie d'un des LFSR et la sortie générale.

Exemple : Générateur de Geffe

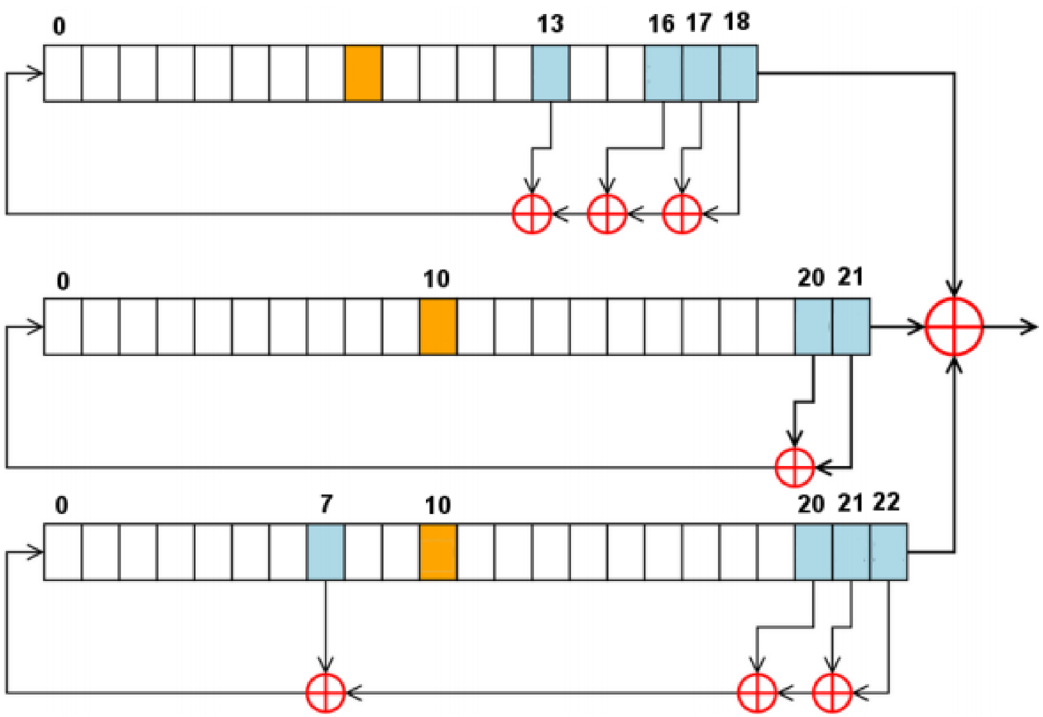
- La sortie du générateur est égale à celle de LFSR2 dans 75% des cas.
- On essaye de deviner l'état initial du LFSR2 (i.e. on essaye tous les états initiaux possibles).
- On compare les résultats obtenus avec le Geffe attaqué d'une part et avec le LFSR2 deviné d'autre part.
- Si on a deviné le bon LFSR2, les résultats sont les mêmes dans 75% des cas, sinon seulement dans 50% des cas.

Cette attaque peut être généralisée à tous les générateurs pseudo-aléatoires qui ont plusieurs composantes

Exemple : A5 (GSM)

3 LFSR de tailles 19,22 et 23, utilisant les polynômes clairsemés

$$X^{19} + X^{18} + X^{17} + X^{14} + 1$$
$$X^{22} + X^{21} + 1$$
$$X^{23} + X^{22} + X^{21} + X^{8} + 1$$



Les registres sont remplis au départ avec une clé de 64 bits (en fait 54 seulement).

Non-linéarité : les registres sont décalés à la majorité des bits du milieu (si on a deux 0 et un 1, on décale les registres où il y a les 0).

- Fait pour être cassable.
- Fondamentalement bon (bonne répartition de la sortie).

Exemple : E0 (Bluetooth)

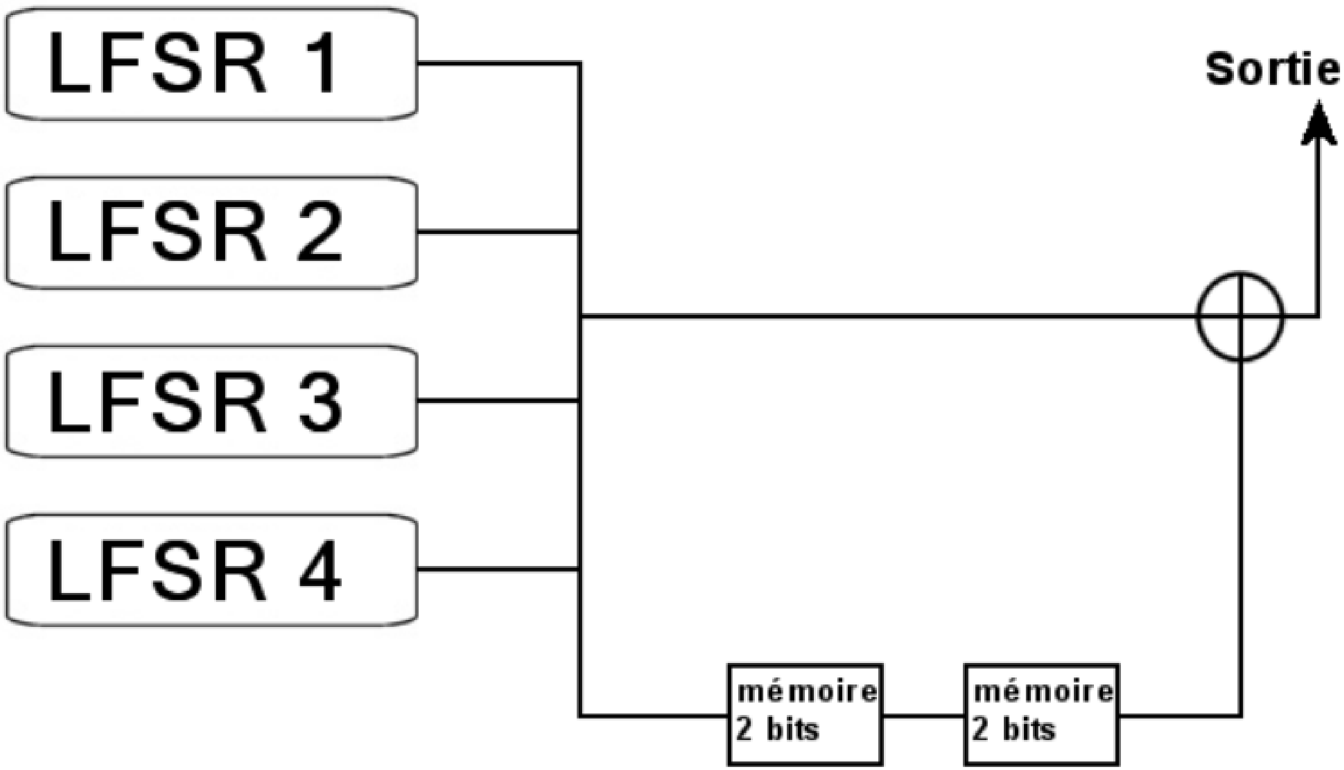
4 LFSR de 25, 31, 33 et 39 bits

$$X^{25} + X^{20} + X^{12} + X^{8} + 1$$
$$X^{31} + X^{24} + X^{16} + X^{12} + 1$$
$$X^{33} + X^{28} + X^{24} + X^{4} + 1$$
$$X^{39} + X^{36} + X^{28} + X^{4} + 1$$

Clé de 128 bits.
Non-linéarité assurée par 2 états internes de 2 bits chacun (mémoire tampon) mis à jour en fonction de

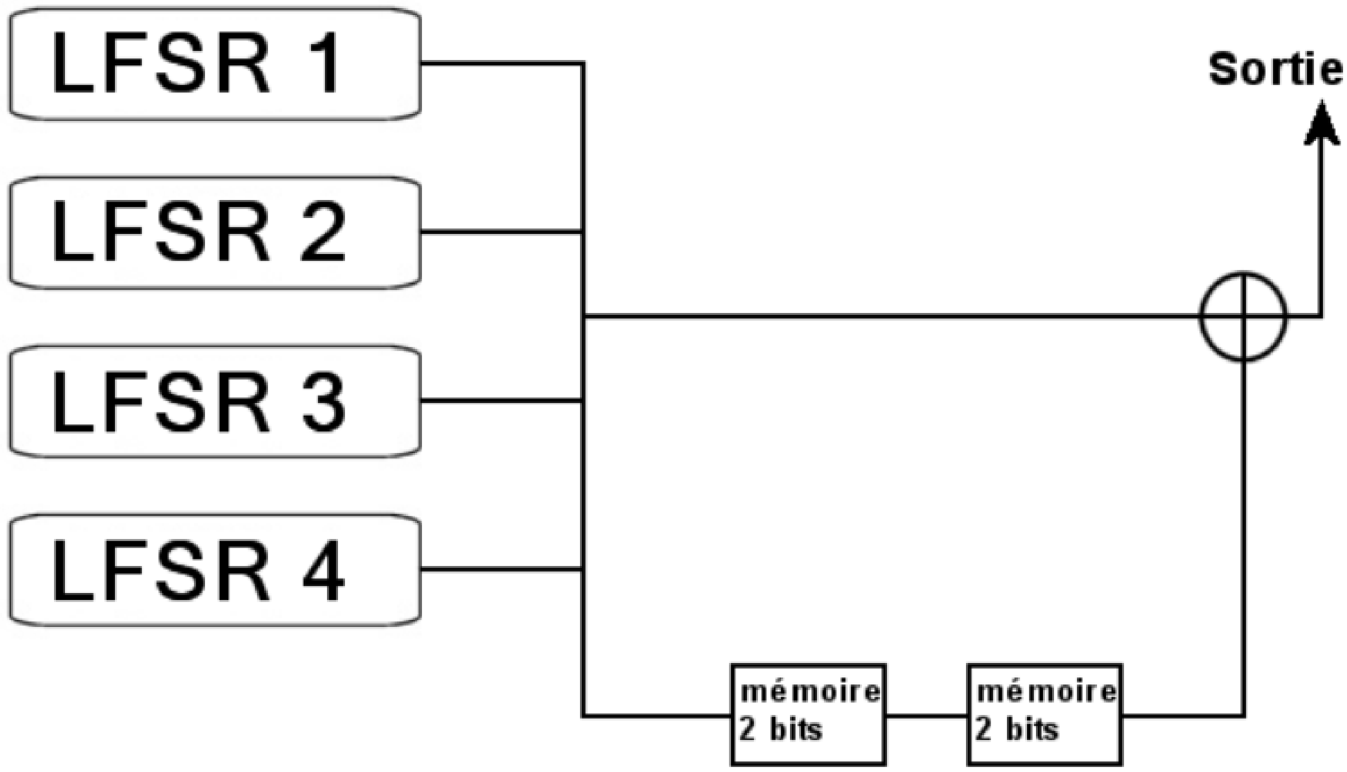
- l'état courant
- l'état précédent
- les sorties des LFSR

Exemple : E0 (Bluetooth)



Attaques en 2^{40} mais gourmandes en mémoire.

Exemple : E0 (Bluetooth)



Attaques en 2^{40} mais gourmandes en mémoire.

Autres exemples : WEP/RC4

Dû à Rivest (1987), commercial.
Détails connus depuis 1994 (équivalent libre : Arcfour).

Fonctionnement

Pas basé sur les LFSR mais sur une table S a 256 cases $(S_0, \ldots, S_{255})$ formant une permutation des nombres entre 0 et 255. Cette permutation initiale dépend de la clé (de 40 à 256 bits).
2 compteur i et j sont initialisés à 0. Pour générer un octet aléatoire :

$$i = i + 1 \bmod 256$$
$$j = j + S_i$$
$$\text{Échanger } S_i \text{ et } S_j$$
$$t = S_i + S_j \bmod 256$$
$$\text{Renvoyer } S_t$$

- Très facile à implémenter.
- Rien n'empêche des tables plus grandes.

Autres exemples : WEP/RC4

RC4-64 (ou 128) avec 24 bits de vecteur initialisation (**IV**), envoyé en clair.

$\Rightarrow$ reste 40 bits de clé (ou 104)

Rôle de **IV** : éviter d'utiliser 2 fois la même graine.
En pratique le **IV** change pour chaque paquet.

Problèmes

- **IV** très court
- Aucune recommandation sur la sélection des **IV** (incrémentation, tirage aléatoire ?)

$\Rightarrow$ ensemble des **IV** très vite parcouru (11h à 54Mbps, voire 10 secondes si c'est aléatoire)

Autres exemples : WEP/RC4

La faiblesse ne vient pas du RC4.

Alternatives

- WPA (**IV** de 48 bits, clés changées régulièrement) compatible avec le WEP.
- WPA2 utilisant AES, pas compatible, plus lent, plus sûr.

Autres exemples : BBS

Générateur de Blum-Blum-Shub

- p, q 2 premiers congrus à 3 mod 4 et $n = pq$
- x_0 un résidu quadratique (graine)
- $x_{i+1} = x_i^2 \bmod n$
- le bit de parité de x_i donne la suite aléatoire

Propriétés

- Prouvé sûr si la factorisation de n est difficile
- Indiscernable d'une suite aléatoire
- Pour avoir une grande période, il faut que $gcd(\varphi(p-1), \varphi(q-1))$ soit petit.
- Affreusement lent

Sources

- Master Crypto de Rennes 1:
https://etudes.univ-rennes1.fr/master-crypto/
Et plus précisément les cours sur le chiffrement par flot
- Cours de Maxime Hauay de l'Université de Aix-Marseille :
https://old.i2m.univ-amu.fr/~mhauray/enseignement/L2-InfoProb/index-L2_InfoProb.html
- « Simulation à événements discrets », G. Fleury, P. Lacomme, A. Tanguy
- « Cryptographie appliquée » B. Schneier
- ANSSI : RGS Version 2.0 Annexe B1

That's all folks