Kévin KERICHE & Nathan CHEVALLIER

Info 910 - Cryptologie
Générateurs de nombre pseudo-aléatoire

1. Introduction

Le générateur de nombres pseudo-aléatoires (PRNG) est un algorithme qui comme son nom l'indique génère des nombres qui semble aléatoire, présentant certaines propriétés du hasard. Ces nombres sont donc supposés être suffisamment indépendants les uns des autres, et aucune structure évidente ne doit apparaître.

Mais malgré tout, il est difficile pour un ordinateur de créer des événements réellement aléatoires. On utilise donc le terme pseudo aléatoire car il simule un processus aléatoire.

Un évènement pseudo-aléatoire a l'air aléatoire, mais il est entièrement prédictible ! Nous disons déterministe parce que son contenu peut être connu par une personne qui sait comment cet événement a été programmé. Ce qui a l'air aléatoire pour l'utilisateur est en fait le résultat d'un algorithme mathématique complètement prédictible.

Nous allons voir comment fonctionne un générateur de nombres pseudo-aléatoires à travers différents algorithmes et comment avons nous pu l'utiliser durant plusieurs siècles.

2. Histoire

Les générateurs pseudo-aléatoires (PRNG) sont énormément utilisés en cryptographie à la suite de recherches principalement motivées par l'importance militaire et économique de différentes époques de notre histoire.

Jusqu'au XIX^{e} siècle, les chiffrements utilisés reposaient essentiellement sur le secret autour de la méthode utilisée, et sur l'absence de traitement de masse. De nos jours, ces méthodes sont inutilisables car il existe de nombreuses théories statistiques qui permettent de retrouver l'algorithme de génération à partir de ses résultats.

En 1883, **Auguste Kerckhoffs** exposera une règle fondamentale de la cryptographie moderne car les techniques de chiffrement ne pouvant logiquement plus être gardées secrètes.

La règle appelée aujourd'hui le ***principe de Kerckhoffs*** se résume par cette phrase : *la sécurité d'un système ne doit pas reposer uniquement sur la méconnaissance de la méthode de chiffrement.*

A la suite de cette règle, des algorithmes de chiffrement par clé voient le jour, ce qui marquera le début des générateurs pseudo-aléatoires.

Leur importance est toutefois restée limitée tant que les moyens physiques de calcul n'ont pu supporter les longs et répétitifs calculs qu'ils nécessitent.

En 1946, **John von Neumann** publie son générateur middle-square, l'un des générateur de nombres pseudo-aléatoires qui inspirera le plus.

Par la suite de nombreux algorithmes rapides et populaires virent le jour:

- 1948 D. H. Lehmer introduit les ***générateurs congruentiels linéaires***.
- 1958 par G.J. Mitchell et D.P. Moore améliore les ***générateurs congruentiels linéaires*** et deviendront par la suite extrêmement répandus, la plupart des fonctions de chiffrement basique y ayant recours.

La ***Méthode de Fibonacci*** comporte des lacunes mais est beaucoup utilisée pour générer une suite de nombre pseudo-aléatoire.

En 1963, **Andreï Kolmogorov** publia une théorie solide étudiant la nature mathématique du concept de suite aléatoire s'est développée et s'appelle ***la théorie de la complexité de Kolmogorov***.

De nos jours, aucun algorithme pseudo-aléatoire ne peut vraiment générer de suite à l'abri de toute analyse statistique, car en théorie elle-même doit être aléatoire et que l'algorithme utilisé ne peut s'initialiser lui-même.

Les générateurs cryptographiques actuels sont donc obligés de faire intervenir une part de hasard qui n'est pas engendrée de façon déterministe : on s'oriente donc vers des générateurs hybrides, fondés sur un algorithme robuste de génération de nombres pseudo-aléatoires, et s'initialisant grâce à un moyen physique de production de hasard.

3. Utilisation

- **La méthode de Von Neumann**

La méthode Von Neumann, anciennement appelée la méthode du carré médian, est un moyen simple de produire une séquence de nombres pseudo-aléatoires mais très imparfaite parce qu'elle converge rapidement vers 0 et on retrouve les mêmes nombres.

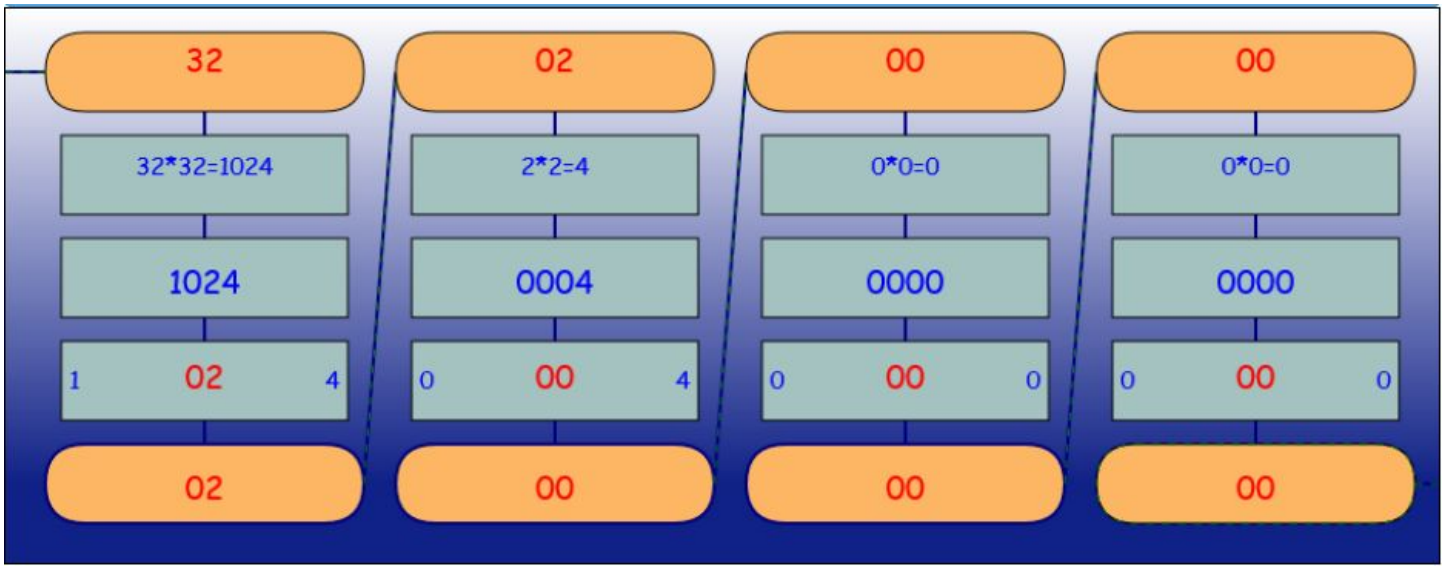
Voici l'algorithme utilisé:

- Commencer par choisir un nombre aléatoire de *n* chiffres.
- L'élever au carré afin d'obtenir un nombre de 2**n chiffres.* Ajouter des zéros à droite si nécessaire
- Extraire les *n* chiffres du milieu. Ce sera le prochain nombre aléatoire.
- Recommencer

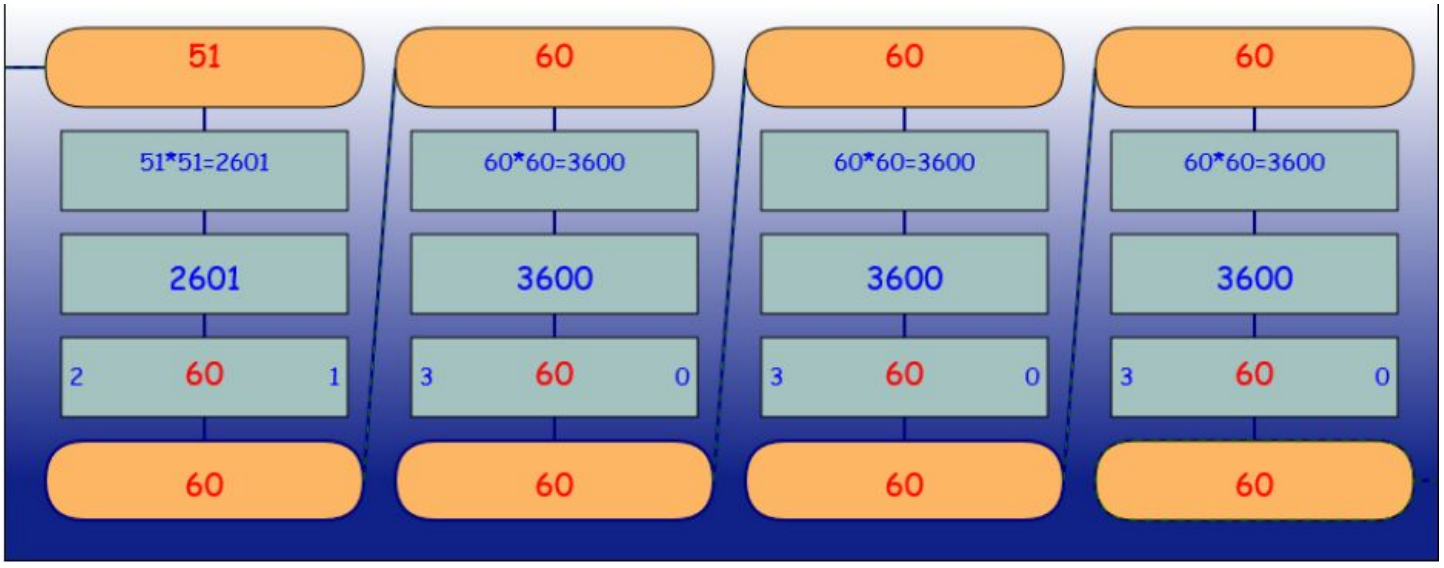
Exemple:

Quelques failles:

- La convergence vers 0:



- Suite de même nombres:

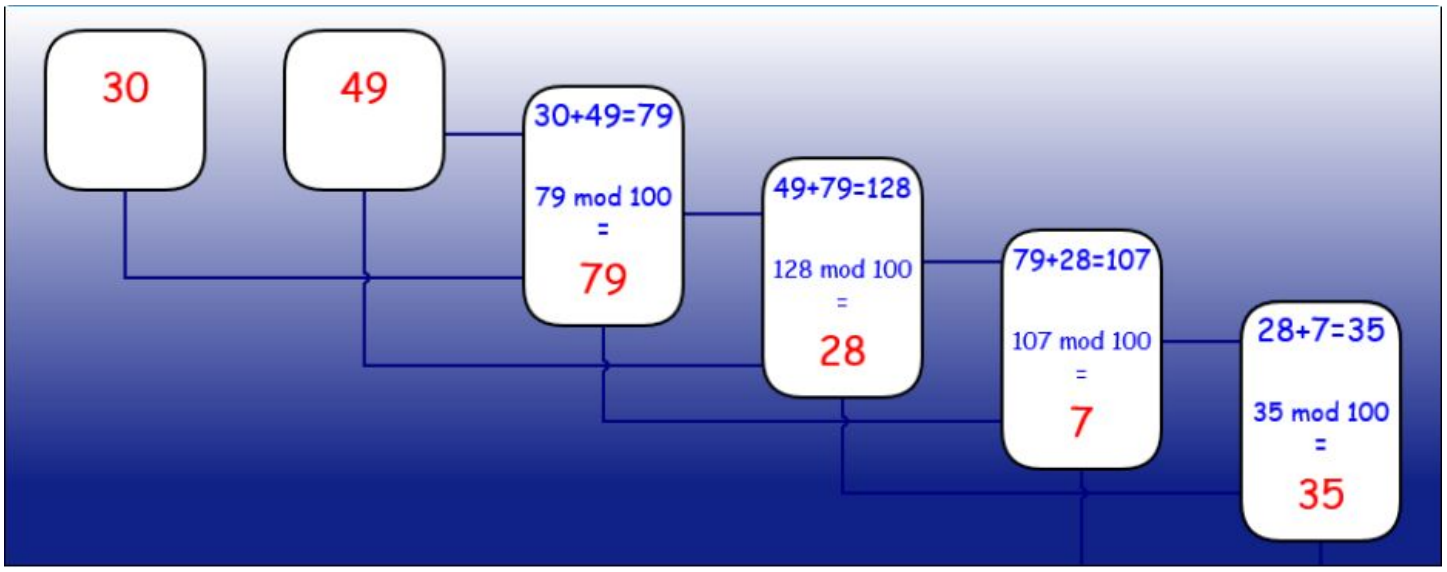


- **Méthode de Fibonacci**

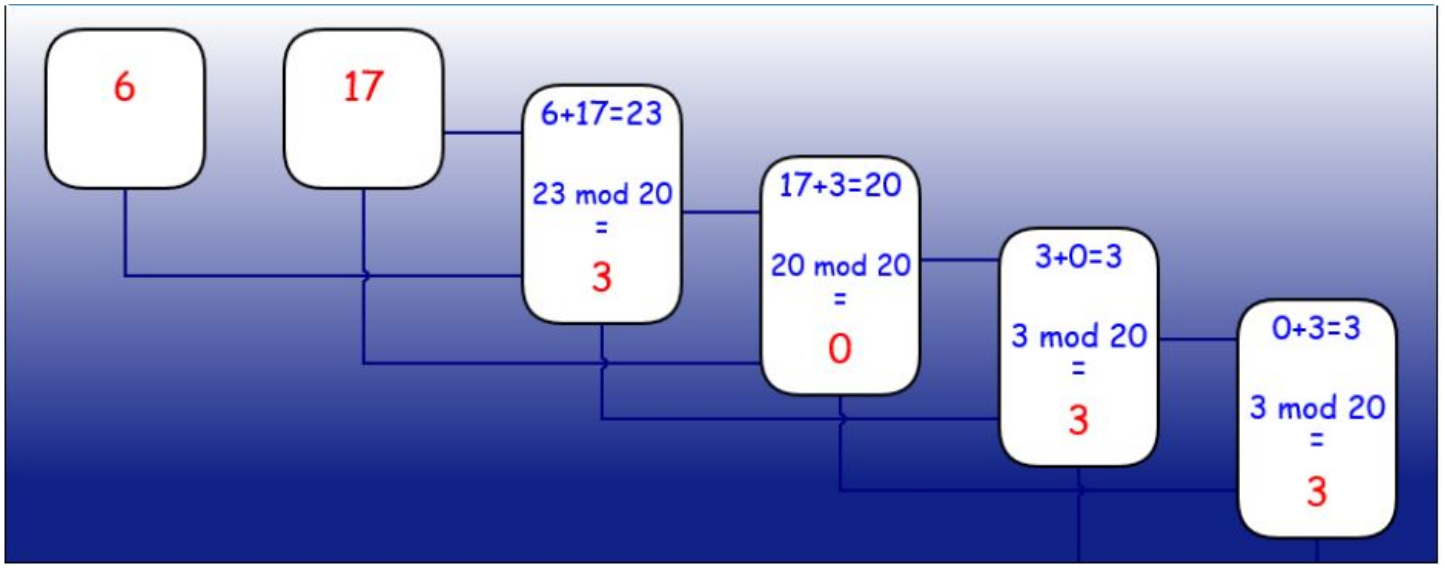
La méthode de Fibonacci, comme son nom l'indique, est basée sur la suite de fibonacci. Cette méthode est donc simple à implémenter et demande peu de ressources. Toutefois, elle dépend fortement du choix des nombres pour l'initialisation.

Voici la formule mathématique: $X_n = (X_{n-1} + X_{n-2}) \bmod M$ (on applique un modulo M à la suite de fibonacci). Autrement dit, chaque terme de la suite est la somme des deux termes précédents modulo M.

Exemple:



Plus le modulo est petit, plus on retrouve la convergence vers 0 et l'apparition de mêmes nombres.



- **Générateurs pseudo-aléatoires cryptographiques**

    - *Yarrow*

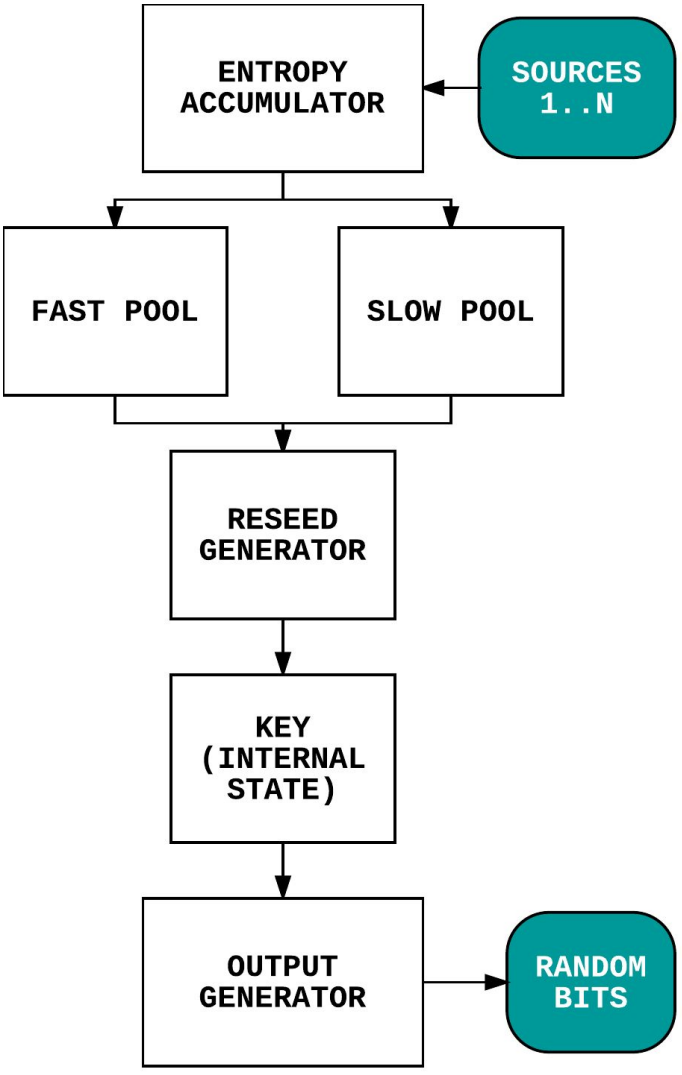
Yarrow dispose de deux *groupes* de 160 bits qui collectent aléatoirement de l'entropie à partir de différentes sources.

Il existe un groupe est dit "rapide" et l'autre est dit "lent". Les données provenant du groupe "lent" sont plus sûres mais utilisées moins souvent. En alternance avec le groupe "rapide", ces résultats sont ensuite hachés par SHA-1 pour devenir ensuite des “*graines”*.

Ces données récoltées sont évaluées par un estimateur d’entropie qui déterminera leur “qualité”, autrement dit, ce qui permettra de savoir si le contenu des groupes est vulnérable aux attaques.

Une fois que les données sont dites “validées”, elles passent dans une fonction de hachage (SHA-1) et vont ensuite servir de clé pour un algorithme de chiffrement asymétrique (3DES).

Enfin, on obtient un flux de données pseudo-aléatoires avec une forte puissance de calcul, ce qui rend impossible la distinction entre une suite généré par Yarrow et une suite parfaitement aléatoire.



- ***Fortuna***

Fortuna est une *famille* de générateurs cryptographiques de nombres pseudo-aléatoires, qui a succédé à Yarrow.

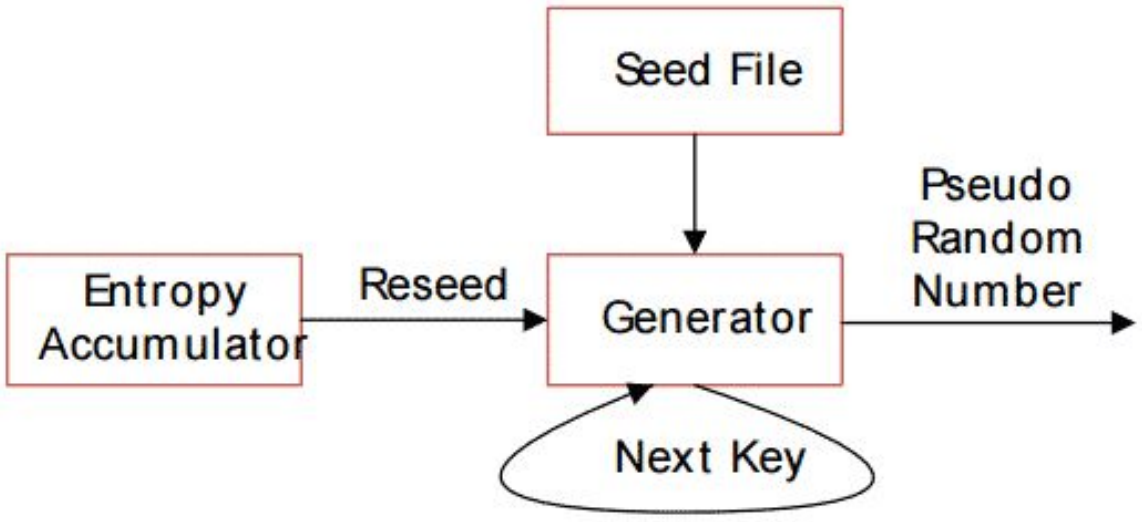
Le générateur est divisé en plusieurs parties :

- le générateur lui-même qui une fois qu'il a reçu une *graine* produira une suite infinie
- l'accumulateur d'entropie qui récupère des données aléatoires à partir de diverses sources, il envoie une nouvelle graine au générateur lorsque l'entropie est suffisante.
- le *fichier des graines* qui stocke suffisamment d'informations pour pouvoir commencer une génération de nombres dès que l'ordinateur a démarré.

Le générateur est basé sur un algorithme de chiffrement par bloc. L'idée est d'employer l'algorithme de chiffrement en mode *compteur*, c'est-à-dire qu'il chiffre successivement les valeurs d'un compteur qui s'incrémente.

Cette procédure n'est toutefois pas suffisante car le compteur a un nombre d'états fini et donc la séquence générée est également finie. C'est pourquoi la clé de chiffrement est périodiquement modifiée. Si plus de 1 mégaoctet de données est généré, alors la clé est automatiquement changée. La clé est également modifiée après chaque requête principale.

L'accumulateur d'entropie est conçu pour être résistant à une attaque par injection de données, sans toutefois utiliser des estimateurs d'entropie trop complexes et donc peu sûrs. On considère plusieurs *groupes* d'entropie, chaque source d'entropie va répartir ses informations dans les différents groupes de manière uniforme. L'idée principale derrière ce système est de produire une nouvelle *graine* en changeant à chaque fois de groupe. Plus formellement, lors de la n-ième création de la graine, le groupe k est utilisé seulement si 2^k divise n. Autrement dit, le groupe k n'est utilisé que sur une durée de $1/2^k$ par rapport au temps total. En augmentant k, les nouvelles graines sont envoyées moins fréquemment mais par contre, on accumule plus d'entropie.



4. Domaines d'application

De nos jours, les générateur nombre pseudo-aléatoire sont utilisé dans divers domaines d’application:

- **Système de chiffrement**

  Les méthodes de chiffrement utilisent régulièrement des suites binaire pseudo-aléatoire.

- **Confidentialité des échanges sur les réseaux sans fil**

  L’échange par réseaux sans fil pose de nombreux problèmes concernant la confidentialité des éléments échangés. Cela dit, il faut alors un mécanisme qui peut conserver le mieux possible cette confidentialité.

  Le nombre pseudo-aléatoire ainsi généré sert à chiffrer la transmission, et alors à assurer la confidentialité de cette dernière.

- **Sécurisation des applications web**

  De nos jours, les applications web ont un système qui sert à protéger leurs utilisateurs par la création de sessions. Pour toutes sortes d’attaques, il est devenu courant d’utiliser un gestionnaire d’applications web capable de générer des identifiants qu’on ne peut pas prévoir.

- **Domaine biomédical**

  Pour les applications biomédicales leurs utilisation des générateur nombre pseudo-aléatoire est la pour améliorer la modélisation et pour accélérer le calcul de traitement. En effet, sur les grilles de calcul, il est impératif que chaque processus de calcul puisse disposer d’une sous-séquence aléatoire issue d’une séquence globale de nombres aléatoires à générer.

- **Une application originale : le GPS**

  Les codes pseudo-aléatoires connaissent une application originale dans le cadre du système de localisation par satellites GPS.

5. Conclusion

Comme nous avons pu le voir, les générateur de nombres pseudo-aléatoire sont utilisé dans différents domaines
Ils doivent être capables de produire une sortie suffisamment peu discernable d'un flux de données aléatoires et doivent également être capable de résister à des attaques telle que l'injection de données utilisée de manière à produire des imperfections dans l'algorithme, ou encore des analyses statistiques qui permettraient de prédire la suite. Ces différentes propriétés font qu'ils puissent être utilisés en cryptographie.

6. Sitographie

https://fr.wikipedia.org/wiki/G%C3%A9n%C3%A9rateur_de_nombres_pseudo-al%C3%A9atoires

http://lwh.free.fr/pages/algo/crypto/prng.html

https://www.apprendre-en-ligne.net/random/pseudo.html

https://fr.wikipedia.org/wiki/Fortuna_(cryptographie)

https://www.wikizero.com/en/Yarrow_algorithm