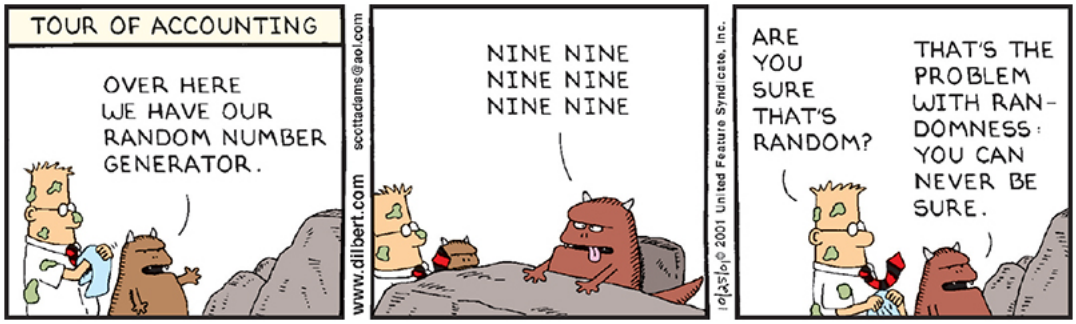
Cybersécurité et qualité des générateurs informatiques de nombres aléatoires

D'une manière un peu paradoxale, il est vital de pouvoir générer des nombres aléatoires en cryptographie et en sécurité informatique, que ce soit entre autres pour générer des clés secrètes de chiffrement, générer des liens de validation (par exemple lorsque l'on crée un compte internet et que l'on reçoit un lien pour valider son mail : celui-ci contient un nombre aléatoire appelé jeton CSRF) ou encore pour simuler des jeux de hasard en ligne (si la génération des nombres aléatoires n'est pas sécurisée, un joueur pourrait casser l'algorithme et prédire les résultats suivants).



La problématique :

Tout programme informatique étant déterministe, pour obtenir un nombre aléatoire, il faut utiliser une source d'entropie externe, c'est-à-dire un phénomène physique aléatoire (température du processeur, bruit électromagnétique d'un disque dur, etc.). Mais c'est un processus couteux en ressources et en temps. C'est pourquoi il faut en général utiliser des **générateurs de nombres pseudo-aléatoires (GNPA)** ; nous étudierons ici différents générateurs et verrons ceux qui peuvent être considérés comme sûrs d'un point de vue cryptographique/sécurité informatique.

Travail à réaliser :

1. Faire un panorama des générateurs aléatoires
   - de manière générale
   - en Python, et discuter de leur qualité pour un usage en cryptographie et en sécurité informatique.
2. Proposer et implémenter des méthodes pour casser des codes utilisant des algorithmes usuels mais utilisés avec des générateurs aléatoires pas assez fiables.
   • En particulier on pourra étudier l'impact de GNPA peu fiables dans le cadre du chiffrement symétrique (particulièrement utile pour chiffrer de grandes quantités de données car très rapide et consommant peu de ressources) et plus précisément pour le **chiffrement de flux** et le **chiffrement par bloc** (pour ce dernier on s'occupera des chiffrements ECB, CBC et CTR)

3. Construire une application permettant de déterminer le degré de fiabilité d'un GNPA pour une utilisation sensible (en cryptographie ou en sécurité informatique).

Outils/Ressources

Il sera intéressant de répondre précisément à la question de savoir pourquoi les fonctions du module « random » en Python ne sont pas suffisamment sécurisés dans le cadre de la cybersécurité et pourquoi, au contraire celles du module « secrets » le sont.
Le **chiffrement par bloc** est basé sur le fameux algorithme AES ; de ses trois variantes, le chiffrement ECB, le chiffrement CBC et le chiffrement CTR, le premier est le moins adapté pour les images, comme on peut le voir clairement sur l'illustration ci-dessous.

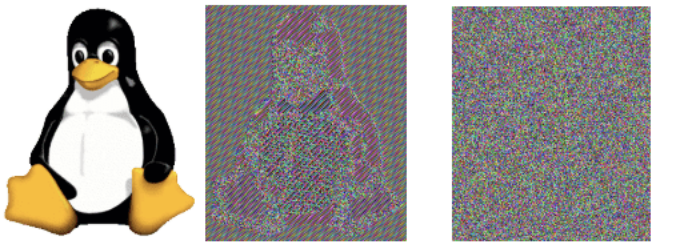


Image non chiffrée Image chiffrée en mode ECB Image chiffrée en mode CBC

https://docs.python.org/fr/dev/library/random.html

https://openclassrooms.com/fr/courses/1757741-securisez-vos-donnees-avec-la-cryptographie/6031865-utilisez-le-chiffrement-symetrique-pour-proteger-vos-informations

https://openclassrooms.com/fr/courses/1757741-securisez-vos-donnees-avec-la-cryptographie/6031865-utilisez-le-chiffrement-symetrique-pour-proteger-vos-informations