Présentation

D'un générateur de nombres pseudo aléatoires de qualité cryptographique

A l’heure d’aujourd’hui, il n’existe qu’un seul algorithme qui a le privilège d’être incassable d’un point de vue mathématique, cet algorithme nous le connaissons tous.

Il consiste à faire un xor avec un masque jetable

Le masque doit être :

-aussi long que le texte à coder

-il doit être complètement aléatoire

-et ne doit être utilisé qu'une seule fois

Des algothrimes existent et permettent de générer des nombres aléatoires, mais pour qu’elles puissent fonctionner de manière satisfaisante il faut que les générateurs aient à leur disposition une bonne source d’entropie qui d’une manière générale est basée sur des éléments en prise directe avec le monde réel.

Par exemple :

- délai entre 2 frappes de touche du clavier,

- déplacement des têtes de lecture d'un disque dur.

Ces générateurs ont comme inconvénient un débit lent et ne peuvent pas convenir pour une application de cryptographie.

Il n’est pas facile de transmettre la source d’entropie basée sur le monde réel.

Il existe aussi un très bon générateur de nombres aléatoires qui a été inventé par 3 mathématiciens : Lenore Blum, Manuel Blum et Michael Shub qui nous ont donné la fameuse formule BBS réputée sûre et lente ; bien qu'avec la puissance actuelle je ne suis pas vraiment certain que cette réputation soit encore à l'ordre du jour pour ce qui concerne la lenteur.

Leur générateur est basé sur une formule mathématique qui malgré la qualité de l'aléa généré met en évidence une certaine limite liée au diviseur qui lui est constant. L'on peut évidemment contourner cette problématique en le choisissant suffisamment grand.

Dans tous les cas, d'un point de vue purement mathématique la réponse est exploitable, donc nous sommes en présence d'un algorithme qui à partir de quelques éléments peut générer une certaine quantité d'aléas ce qui permet de créer le fameux masque jetable des 2 côtés du canal de la transmission.

Maintenant, j’attire votre attention que malgré le fait que nous ne nous connaissons pas, nous partageons un savoir commun ; ce savoir en commun, je décide de le hiérarchiser, de le classer, de l’organiser puis je vais le parcourir et se sont les éléments du parcours ou le parcours qui va me permettre de créer des nombres aléatoires.

Entrons maintenant dans le vif du sujet.

Il nous faut un savoir commun infini que je puisse organiser, agencer avec une organisation induite par mes choix de départ la seule réponse que j’ai trouvée consiste à se servir de l’ensemble des nombres premiers ; vous noterez l’originalité de la démarche.

C'est relativement nouveau l’exploitation des nombres premiers

et la cryptographie :-)

Donc maintenant nous avons un savoir commun mathématique infini à notre disposition et donc une bonne bande passante.

Il ne nous reste donc plus qu’à parcourir ce savoir en commun puis de se servir de ce parcours pour générer nos nombres aléatoires.

Pour la génération des nombres aléatoires, je me suis fortement inspiré de la formule créée par les 3 mathématiciens Lenore Blum, Manuel Blum et Michael Shubqui ont publié la formule appelée BBS.

$$x_{n+1} = (x_n)^2 \bmod M$$

Avec $M = pq$, le produit de deux grands nombres premiers p et q, et la sortie de l'algorithme l'on prend le bit le moins significatif.

Je me contente donc d’un simple modulo entre 2 nombres premiers avec le bit de poids le plus faible.

Avant de passer au parcours et donc au choix des nombres premiers, je vais détailler une approche qui permet de générer des nombres premiers de bonne taille.

Dans un premier temps, je me contente d’un test statistique pour définir si le nombre est premier, ensuite de me concentrer uniquement sur les nombres premiers de type Dirichlet A + n x B

Cet ensemble de nombres est infini .

Je prends un nombre premier de petite taille et je le considère sous son écriture en base 2, je vais ensuite lui rajouter par additions successives des puissances de 2 donc nous avons un nombre premier en base 2 qui par la suite pourra très bien s'écrire 2^n,.....111 en base 2 ce qui permet de créer des nombres premiers de bonne taille de manière très rapide.

J'obtiens très rapidement des nombres premiers d'une centaine de bits.

Pour simplifier la compréhension, imaginons un nombre premier de 1OO bits ; dans ce nombre de 100 bits, je vais rechercher les nombres premiers qui existent en enlevant successivement le bit de poids le plus fort, puis je teste le reste, si ce n'est pas un nombre premier je ré-enlève le bit de poids le plus fort puis je re-teste le tout jusqu'à ce qu'il ne reste plus de bits.

Pour la création de grands nombres premiers, je fais exactement le contraire,

par exemple :

11
111
10111
10010111
1110010111
1111110010111
10101111110010111
1001010101111110010111
1100101010111111O010111
110111001010101111110010111
10111011100101010111110010111

Donc nous disposons maintenant d'un savoir commun infini, d'un générateur de nombres premiers et d'un calcul le reste d'une division qui ne donne pas beaucoup d'informations sur le diviseur ni sur le nombre à diviser pour un attaquant.

Maintenant, je me propose de mettre tout cela en ordre de marche pour faire un générateur que je considère de bonne qualité.

La graine :

deux nombres premiers et une valeur qui indique la taille des nombres.

Le premier des nombres, je vais le faire croître avec l'algo précédemment décrit à une bonne centaine de bits.

Cette taille est purement subjective, elle représente pour moi un bon compromis entre sécurité et vitesse de calcul bien pour la vitesse cette algorithme ne soit pas vraiment très bon.

Donc j'ai un nombre premier de 100 bits, je prends le deuxième nombre premier de la graine que je fais pousser à 50 bits ce qui me donne 2 nombres premiers : un de 100 et un autre de 50 bits.

Puis j'effectue le calcul qui consiste à calculer le reste du premier par le deuxième, puis à prendre les bits de poids les plus faibles du résultat, je change les deux nombres premiers et je recommence les différents tests que j'ai fait, cela va me permette de mettre en avant la qualité de l'aléa généré.

Les résultats pour 10 Mo ainsi que le code source sont sur mon site que Google n'a pas encore référencé, une histoire de hasard sûrement :

http://remy.aumeunier.generateur.adsl

Vous trouverez les sources et les résultats des tests pour faire une comparaison avec de l'aléa généré avec la formule BBS , de l'aléa hard bruit blanc et avec de l'aléa radioactif inoffensif.

Je rajoute que cette approche est cassable puisque le calcul est prédictible si il y a connaissance de tous les éléments donc la sécurité de cryptage ne réside pas uniquement par la répartition aléatoire, des éléments (0/1) mais aussi dans son non côté non prédictible.

Remy Aumeunier

réf: le très bon portail sur la cryptographie

http://fr.wikipedia.org/wiki/Cryptographie