Cours Turing
Semaine 12

Note : Le paragraphe qui suit est certes un peu mathématique, mais c'est aussi un ingrédient essentiel à ce qui va suivre dans le cours, notamment pour une bonne partie de ce qui concerne la cryptographie à clé publique.

1 Faisons tout d'abord un peu d'arithmétique modulaire...

Définition :
Etant donné deux nombres entiers positifs A, N, $A \pmod N$ (prononcer "A modulo N") est le reste de la division de A par N ; $A \pmod N$ est donc un nombre compris entre 0 et $N-1$.

Exemple : $64 \pmod{11} = 9$, car $64 = 5 \cdot 11 + 9$ (donc le reste de la division vaut 9).

A noter qu'en Python, $A \pmod N$ s'écrit simplement $A\%N$.

Addition modulaire

Pour effectuer une addition modulo N, on effectue d'abord une addition classique, puis on calcule le reste de la division par N du résultat.

Exemple : $(64 + 17) \pmod{11} = 81 \pmod{11} = 4$.

Ceci dit, pour effectuer une addition modulaire, on peut aussi utiliser la propriété suivante, qui consiste à "prendre les modulos" avant d'effectuer l'addition :

$$(A + B) \pmod N = (A \pmod N + B \pmod N) \pmod N$$

ce qui permet de légèrement simplifier le calcul de l'exemple précédent :

$$\begin{aligned}(64 + 17) \pmod{11} &= (64 \pmod{11} + 17 \pmod{11}) \pmod{11} = (9 + 6) \pmod{11} \\ &= 15 \pmod{11} = 4\end{aligned}$$

On voit en effet que seule une addition de deux chiffres (9 et 6) est requise ici (plus besoin d'effectuer l'addition de 64 et 17). Ceci ne semble pas constituer un si grand avantage, car les calculs intermédiaires des modulos sont à effectuer en plus, mais cet avantage prend tout son sens lorsqu'on parle de multiplication...

Multiplication modulaire

De la même manière que pour l'addition, on effectue d'abord une multiplication classique, puis on prend le modulo.

Exemple : $(64 \cdot 17) \pmod{11} = 1088 \pmod{11} = 10$.

Ceci dit, il est aussi possible de prendre les modulos avant d'effectuer la multiplication, grâce à la propriété suivante :

$$(A \cdot B) \pmod{N} = (A \pmod{N} \cdot B \pmod{N}) \pmod{N}$$

ce qui permet de considérablement simplifier le calcul de l'exemple ci-dessus :

$$\begin{aligned}(64 \cdot 17) \pmod{11} &= (64 \pmod{11} \cdot 17 \pmod{11}) \pmod{11} = (9 \cdot 6) \pmod{11} \\ &= 54 \pmod{11} = 10\end{aligned}$$

Au lieu d'effectuer la multiplication $64 \cdot 17 = 1088$, on ne doit calculer que $9 \cdot 6 = 54$, qui fait partie de la table des livrets appris par cœur à l'école primaire.

En résumé, effectuer des opérations en arithmétique modulaire est très pratique, car cela permet de ne jamais faire de calculs impliquant des nombres plus grands que N.

Tables d'addition et de multiplication

Considérons $N = 5$ et dressons les tableaux de l'addition et de la multiplication modulo 5 :

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\cdot$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Il est remarquable que dans le tableau de l'addition, chaque ligne et chaque colonne contient exactement l'ensemble des chiffres de 0 à 4. Il en va de même dans le tableau de la multiplication : si on ignore la ligne et la colonne de zéros, chaque ligne et chaque colonne contient exactement l'ensemble des chiffres de 1 à 4.

Voyons maintenant ce qui se passe pour $N = 4$:

$+$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\cdot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

On voit que si d'une part la propriété précédente reste vraie pour le tableau de l'addition, il n'en va plus de même pour le tableau de la multiplication : la ligne et la colonne correspondant au chiffre 2 ne contiennent ni le chiffre 1, ni le chiffre 3, et contiennent même un 0 en leur intersection. Ceci est dû au fait que 4 n'est pas un nombre premier (tandis que 5 est premier).

Soustraction et division modulaires
(*Note :* Le présent paragraphe peut être laissé de côté en première lecture.)

Comment effectuer les opérations inverses de l'addition et de la multiplication, à savoir la soustraction et la division ? La réponse est simple, une fois qu'on a dressé les tables ci-dessus.

Commençons par la soustraction : calculer $(A - B) \pmod N$ revient à chercher le nombre C tel que $(B + C) \pmod N = A$. Pour calculer $(1 - 4) \pmod 5$, par exemple, on cherche le nombre C tel que $(4 + C) \pmod 5 = 1$. Ceci se fait en parcourant la première table d'addition établie ci-dessus : dans la ligne 4, on cherche le chiffre 1, et on identifie la colonne de celui-ci, en l'occurence, la colonne 2. Donc $(1 - 4) \pmod 5 = 2$. Une démarche similaire permet d'effectuer une soustraction modulo 4, ou plus généralement modulo N pour toute valeur de N.

Pour la division, le procédé est encore une fois similaire (si on oublie le nombre 0) : pour calculer $(A/B) \pmod N$, on cherche le nombre C tel que $(B \cdot C) \pmod N = A$. Par exemple $(1/4) \pmod 5$ est le nombre C tel que $(4 \cdot C) \pmod 5 = 1$. En parcourant la table de multiplication modulo 5, on cherche donc à la ligne 4 la colonne dans laquelle se trouve le chiffre 1, en l'occurence, la colonne 4. Donc $(1/4) \pmod 5 = 4$. Lorsqu'on voit ça pour la première fois, il faut bien reconnaître que c'est un peu surprenant, car classiquement, on aurait envie de dire que $1/4 = 0,25$. Mais en arithmétique modulaire, le résultat d'une opération doit toujours être un nombre entier compris entre 0 et $N - 1$, même si c'est un peu contre-intuitif...

Et vous aurez aussi sans doute déjà remarqué qu'une division modulo 4 peut poser problème. Par exemple, que peut valoir $(3/2) \pmod 4$? Le nombre 3 n'apparaît pas dans la ligne 2 de la table de multiplication modulo 4. La réponse est qu'il n'est tout simplement pas possible de définir correctement la division modulo N si N n'est pas un nombre premier.

Applications

A ce stade, vous êtes en droit de vous poser une question : mais à quoi peut donc bien servir cette arithmétique modulaire ? Voici plusieurs réponses qui nous intéressent (mais il en existe beaucoup d'autres, bien sûr).

1. Contrairement aux opérations d'addition et de multiplication classiques, les opérations modulaires, lorsque répétées plusieurs fois, ont des comportements peu prévisibles (on parle même de "comportement chaotique") : c'est pour cela qu'elles peuvent se révéler utiles pour générer des nombres aléatoires. Nous verrons cela en détail dans la prochaine section.
2. Curieusement peut-être, l'arithmétique modulaire est aussi très utile pour trouver de (très) grands nombres premiers.
3. Il existe une opération modulaire dite "à sens unique", c'est-à-dire, une opération facile à faire, mais très difficile à inverser.
4. Ces deux derniers points sont des éléments essentiels de la cryptographie dite "à clé publique", que nous verrons ensemble dans les prochaines semaines.

2 Générateurs de nombres aléatoires : une introduction

La semaine dernière, nous avons vu le système de clé à usage unique et comment sa sécurité repose sur la génération d'une clé K aléatoire. Il existe un grand nombre d'applications en informatique, et plus particulièrement en cryptographie, qui reposent sur la génération de nombres aléatoires. Mais comment générer des nombres aléatoires à partir d'un ordinateur, qui est fondamentalement une machine déterministe ?

Pour simuler le hasard, une première idée est d'utiliser, par exemple, le nombre de millisecondes écoulées depuis le début d'une journée au moment où on veut tirer un nombre au hasard ; ainsi, si le temps indiqué par l'horloge de l'ordinateur indique 9 heures, 13 minutes, 45 secondes et 174 millisecondes, alors ce nombre vaut 33225174. Ceci fournit un grand nombre qu'on peut considérer comme plus ou moins aléatoire (en considérant au besoin seulement les 5 derniers chiffres de ce nombre, ici 25174, pour laisser de côté les premiers chiffres qui seront toujours identiques si on tire plusieurs nombres de suite). Mais comment choisir ensuite d'autres nombres entiers aléatoires ? Si on consulte l'horloge de l'ordinateur à intervalles réguliers pour trouver d'autres nombres, forcément qu'un motif répétitif fera irruption dans la séquence et endommagera la côté aléatoire de celle-ci.

On peut cependant retenir le premier nombre ainsi trouvé comme *graine* (ou *seed* en anglais) à fournir à un algorithme qui, à partir de là, génèrera une séquence de nombres qu'on espère "les plus aléatoires possible". Encore une fois, vu que les ordinateurs sont des machines déterministes, l'algorithme, quel qu'il soit, ne pourra pas fournir de vrais nombres aléatoires : on parle donc de générateurs *pseudo*-aléatoires. Dans les prochaines sections, nous considérons deux exemples.

2.1 La méthode des carrés tronqués (Von Neumann, vers 1950)

Une première idée pour générer des séquences de (grands) nombres aléatoires est la suivante. A partir d'un nombre à 8 chiffres, par exemple $X = 30472901$, calculons son carré :

$$X^2 = (0)928597695355801$$

et ne retenons que les 8 chiffres du milieu de celui-ci (en rouge) pour le prochain nombre, et ainsi de suite... Cette idée, pour ingénieuse qu'elle soit, génère cependant une séquence de nombres qui est loin d'être aléatoire : au bout d'un moment, la séquence ainsi produite tombe sur un nombre qui se répète (comme par exemple $X = 60 \to X^2 = 3600$) ou effectue une boucle à travers quelques valeurs seulement (comme par exemple $X = 57 \to X^2 = 3249 \to (0)5776$).

Note : Ceci n'enlève rien pour autant au génie de John Von Neumann, brillant mathématicien et physicien, qui fut un non seulement un des pères de l'informatique moderne avec Alan Turing, mais aussi une des premières personnes à envisager d'utiliser le hasard pour résoudre des problèmes complexes, d'où sa contribution à la recherche de générateurs de nombres aléatoires.

2.2 Les générateurs "à congruence linéaire" de Lehmer (vers 1950)

Cette deuxième idée fait appel à l'arithmétique modulaire : soit N un nombre entier positif et A, X_0 deux nombres compris chacun entre 2 et $N-1$. A partir de ces nombres, on génère la suite de nombres suivante :

$$X_1 = (A \cdot X_0)\,(\text{mod N}), \quad X_2 = (A \cdot X_1)\,(\text{mod N}), \quad X_3 = (A \cdot X_2)\,(\text{mod N}), \quad \ldots$$

définition qu'on peut réécrire de manière plus compacte :

$$X_{n+1} = (A \cdot X_n)\,(\text{mod N}), \quad \text{pour } n \geq 0$$

La question est maintenant de savoir si la suite de nombres $X_0, X_1, X_2, X_3, \ldots$ ainsi générée imite bien les propriétés d'une suite aléatoire, ou pas. Voyons tout d'abord quelques exemples.

Note : Vu que tous les nombres générés seront des nombres compris entre 0 et $N-1$, il est clair qu'en pratique, on aura envie de choisir N (très) grand. Mais pour comprendre un peu mieux ce qui se passe, on choisira ici des nombres N plutôt petits.

Commençons donc avec $N = 5$, $A = 4$ et $X_0 = 2$. Ceci donne lieu à la séquence suivante :

$$\begin{aligned} X_0 = 2, \quad X_1 &= (4 \cdot 2)\,(\text{mod } 5) = 3, \quad X_2 = (4 \cdot 3)\,(\text{mod } 5) = 2, \\ X_3 &= (4 \cdot 2)\,(\text{mod } 5) = 3, \quad X_4 = (4 \cdot 3)\,(\text{mod } 5) = 2, \quad \ldots \end{aligned}$$

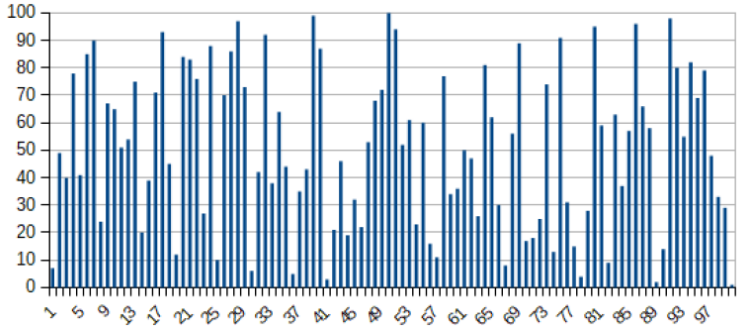
donc une alternance de 2 et de 3 : pas vraiment une séquence aléatoire !

En choisissant $A = 3$, on trouve par contre :

$$\begin{aligned} X_0 = 2, \quad X_1 &= (3 \cdot 2)\,(\text{mod } 5) = 1, \quad X_2 = (3 \cdot 1)\,(\text{mod } 5) = 3, \\ X_3 &= (3 \cdot 3)\,(\text{mod } 5) = 4, \quad X_4 = (3 \cdot 4)\,(\text{mod } 5) = 2, \quad \ldots \end{aligned}$$

Bien sûr, comme N est petit ici, on n'obtient pas vraiment une séquence aléatoire non plus, mais au moins une séquence qui passe *par toutes les valeurs comprises entre 1 et 4 avant de revenir au point de départ* : c'est déjà beaucoup mieux !

Essayons encore avec des nombres plus grands : $N = 101$, $A = 7$ et $X_0 = 7$. Dans ce cas, la séquence des nombres ainsi générée explore aussi toutes les valeurs possibles comprises entre 1 et 100 avant de revenir au point de départ, comme le montre le graphique suivant :



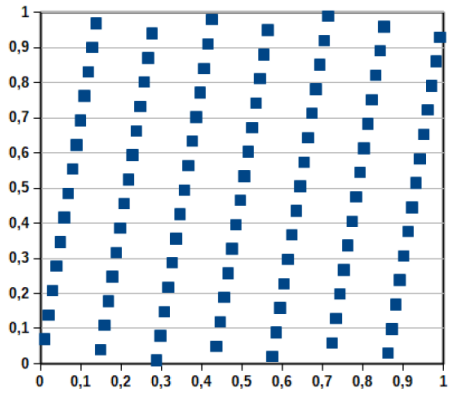
et les nombres semblent en effet explorés dans un ordre relativement aléatoire.

Le défaut principal de cette méthode

En reprenant l'exemple de la page précécente avec $N = 101$, $A = 7$ et $X_0 = 7$, représentons maintenant la succession de ces mêmes nombres sur un graphe en deux dimensions, où chaque point du graphe a pour coordonnées (x, y), avec x, y des nombres compris entre 0 et 1 tels que

$$x = \frac{X_n}{N} \quad \text{et} \quad y = \frac{X_{n+1}}{N} = \frac{(A \cdot X_n) \,(\text{mod } N)}{N}$$

pour une certaine valeur de n. Voici à quoi ressemble ce graphe :



On voit apparaître ici une régularité qui n'est pas vraiment désirable lorsqu'on veut générer des séquences de nombres aléatoires.

Ceci dit, si maintenant $N = 2^{31} - 1 = 2147483647$, $A = 7$ et $X_0 = 7$, cette méthode de génération de nombres aléatoires reste assez efficace !