MATH EN INFO

Puissance modulaire

FICHE EXPLICATION Nº 12

1 Descriptif

L'objectif de ce défi est d'implémenter une méthode calculant pour un nombre binaire b, le nombre binaire b^a modulo N (où a et N sont deux autres nombres binaires donnés en paramètres.

Compte tenu de la taille des nombres manipulés, il est impératif d'utiliser un algorithme optimisé pour réaliser ce calcul. En effet,

1. Si pour calculer b^a, on se contente de multiplier a fois par b, cela fonctionne pour de petites valeurs de a mais prendra un temps infini (environ 10^{20} siècles) si a est de l'ordre de grandeur de 2^{128} (ce qui sera le cas...).
2. Si on calcule b^a puis que l'on réalise le modulo N, nous devrons donc manipuler pendant un temps b^a, or si a et b sont de l'ordre de grandeur de 2^{128}, b^a aura 2^{135} bits et donc nécessiterait plus de 5.10^{30} gigaoctets de mémoire....

Heureusement, il est possible simplement d'éviter ces deux problèmes :

1. Pour le calcul de b^a, on utilisera un algorithme appelé exponentiation rapide. Cet algorithme utilise la décomposition en binaire de a pour trouver une suite très courte de multiplication à faire pour calculer b^a. Dans l'idée, pour calculer x^{17}, l'algorithme calculera $x^2 = x * x$ puis $x^4 = x^2 * x^2$ puis $x^8 = x^4 * x^4$ puis $x^{16} = x^8 * x^8$ et enfin $x^17 = x^{16} * x$ effectuant ainsi 5 multiplications au lieu de 17. De façon générale, cet algorithme effectuera environ $\log_2(a)$ opérations pour calculer b^a. Ainsi si a est d'un ordre de grandeur proche de 2^{128}, l'algorithme n'effectuera que 128 opérations !
2. Pour éviter de manipuler b^a, on effectuera simplement le modulo à chaque étape de l'algorithme précédent. Ceci assurera que l'on ne manipulera jamais de nombres plus grands que N^2.

Ces deux algorithmes peuvent facilement être trouvés sur internet.

2 Protocole

1. Une fois la connexion établie, le serveur commence par envoyer un premier message annonçant le début du défi :

   -- Debut du defi : Puissance modulaire --

   Ce message n'attend pas de réponse.
2. Le serveur envoie ensuite une série de nombres binaires (de taille aléatoire) trois par trois.
3. Pour chaque triplet (b, a, N) de nombres binaires, le serveur doit recevoir en retour un nombre binaire (sous forme binaire) égal à b^a modulo N.
4. Après chaque réponse, le serveur enverra un message commençant par "OK" ou "NOK" suivant si la réponse est correcte ou non.

5. A la fin du défi, le serveur enverra un message indiquant "Defi valide" ou "Defi echoue !". Aucune réponse n'est attendue.
6. Le serveur terminera la communication par le message "FIN", votre client devra alors fermer la socket. Aucune réponse n'est attendue.

3 Exemple de communication

Voici un exemple (incomplet) d'une communication pour ce défi. Dans cet exemple les "<" et ">" indiquent le sens de transfert de chaque message et ne doivent pas être présents dans la communication.

< -- Debut du defi : Puissance Modulo --
< 10101011011111010111110111000110111110111001
< 11111000001010010101
< 10110111100010100101100010
> 1110111001010101111101
< OK
< 10101010001101010000011110
< 1001101100010101
< 1011
> 0
< OK
< 11001010110
< 11111001010110010000100010000010111011001
< 10001110000101010110010010111111000001001
> 11001011000111101110100100100100110000
< OK