MATH EN INFO

Chiffrer

FICHE EXPLICATION Nº 21

1 Descriptif

L'objectif de ce défi est d'implémenter une méthode permettant de chiffrer un message entier en utilisant la clé publique. Pour chiffrer un long message, il suffit de découper ce message en morceaux (de la taille getTailleMorceau), de chiffrer chacun des morceaux indépendamment puis de concaténer les résultats. Une méthode scinder et une méthode concatener son disponible dans la classe NombreBinaire.

Dans le test, les messages à chiffrer feront exactement 252 bits.

2 Protocole

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

   – Début du défi : Chiffrer –

   Ce message n'attend pas de réponse.
2. Le serveur envoie ensuite une série de triplets (M,N,e) où M est un mot binaire de 252 bits et N et e des nombres binaires.
3. Pour chaque triplet (M,N,e), le serveur doit recevoir en retour M chiffré avec la clé (N,e) (sous forme binaire).
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 "Défi validé" ou "Défi échoué !". 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.

< -- Début du défi : Chiffrer --
< 00101001001011111111100110000000100000011010000101101100001101 1...
< 10000001010100101001001011011110110100000010101110000111001001...
< 00010010101000011011010000000010110110000011110000010111101000...
> 01100000110011001101111100000000101011000000011011010011001101...
< OK
< 01101101101011111111110010011010101011000111001001100000111 00...
< 11010011010101110100010011010001000100010100111000110010111 01...
< 10001100111100100100000001000110110001011110110011000110100 10...
> 01011011011010010010000100000010111110100010100101100111001 01...
< OK
< 01010101001100010110101000111001111011011100000011000001011100...
< 11100011101000010100100111111111100010011101000011000010110 1...
< 10010010010010111111001000000001011101000001111100111011101 0...
> 10011000010111011101100111001101001011001110010111100000010 1...
< OK