MATH EN INFO

Déchiffrer

FICHE EXPLICATION Nº 22

1 Descriptif

L'objectif de ce défi est d'implémenter une méthode permettant de déchiffrer un message entier en utilisant la clé publique et la clé privée et l'algorithme de RSA. On rappelle que pour déchiffrer un long message, il suffit de découper ce message en morceaux de la taille de la clé et de déchiffrer chacun des morceaux indépendamment.

Dans le test, les messages à déchiffrer feront exactement 256 bits.

2 Protocole

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

   -- Debut du defi : Dechiffrer --

   Ce message n'attend pas de réponse.
2. Le serveur envoie ensuite une série de triplet (M,N,d) où M est un mot binaire de 256 bits et N et d des nombres binaires.
3. Pour chaque triplet (M,N,d), le serveur doit recevoir en retour M déchiffré avec la clé (N,d) (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 "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 : Dechiffrer -- <
00000101010111011110100001111110001000000111011000001111011111...
< 10100100000010111001100101000000010011001000010001011011110001...
< 10011001101110110100001001011101111111010101010010011010010111...
> 01101001100100000100011001011111101111001101100100000101100111...
< OK
< 00001001111000111101110000010101000111101000001011100111110000...
< 01101010101011101100010000111111010101011011110011110111111101...
< 00001001100010110111111011000000000100010101000010110010010100...
> 00001110111000001010000111111000010111110111010111111011010110...
< OK
< 10101001111011100101101010000111100000110010010010101000010000...
< 10000001011100101010111111011101001110111010001100001110111100...
< 01010000010110110000110110011010000000001011000000101000010111...
> 00111111111000011001101001011110111010101110001101000110101111...
< OK