

Générer la clé privée

1 Descriptif

L'objectif de ce défi est d'implémenter la méthode permettant de générer la clé privée de l'algorithme de RSA. Pour ce défi, on pourra ajouter un setter aux attributs P, Q, phi et e de la classe `GenerateurDeClesRSA`. On rappelle que la clé privée d est l'inverse de e modulo phi et que $phi = (P - 1)(Q - 1)$.

2 Protocole

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

– Debut du defi : Generer cle privée –

Ce message n'attend pas de réponse.

2. Le serveur envoie ensuite une série de triple (P, Q, e) où P, Q et e sont des nombres binaires.
3. Pour chaque triplet (P, Q, e) , le serveur doit recevoir en retour la clé privée d sous forme d'un `MotBinaire` de la taille de la clé.
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 : Generer cle privee --
< 111011110011111000101100011011001111111011100110001000101110001
< 11110001101011001010100010110011000000010001000000010010111100011
< 101000001111011110010101100100101110100101010011100101001001011101001110...
> 100010000011111000010011010001000001010110110110111100001001000100101111...
< OK
< 1010100000110001110000010110000010110011100101000000000100110001
< 1100011011101100000110011000001000101101001010011111011011111011
< 101000011011100010110000110011010111000001011110101111101110101111101011...
> 000001100000010101110001011100100011010110100110100110010011110011001111...
< OK
< 111100110100000101111000010111110110111001110001100000111000001
< 1011010001101010111100111010011100001111000101101110101101011111
< 101011000101011100001111000001101000000101001101101111000001110100011111...
> 010000100110110011001110011011001110100110000101111001100001001110111010...
< OK
```