DM - Réseau

Adam Chareyre

Table des matières

Sujet

L'objectif du sujet est d'écrire un programme en Python qui se comporte comme un serveur. Il devra être en écoute sur le port TCP 4567.
Il devra suivre le comportement suivant :

- Envoie la chaîne de caractères "Quelle est votre année de naissance ?" au client
- Le client répond avec une année au serveur
- Le serveur qui connaît maintenant l'année de naissance du client lui renvoie son âge.

Mise en place

En réalité, le sujet est relativement simple à traiter. En effet, le calcul de l'âge est une simple soustraction et les interactions client-serveur ont déjà été présentées lors des TP 3 et 4. Beaucoup d'attention a été donnée à des détails plus ou moins important qui seront vu plus loin.
Tout d'abord, il est nécessaire de développer un serveur qui proposera une interface sur laquelle se connecter, c'est le rôle du fichier **server.py**. Pour que l'application soit portable, l'adresse IPv4 choisie pour le serveur est celle du localhost, autrement dit 127.0.0.1.
Pour vérifier l'exactitude du serveur, un client est disponible au travers du fichier **client.py**.
Ces deux fichiers sont suffisants pour répondre au sujet.
Vous trouverez, en fonction du système d'exploitation que vous utilisez, des fichiers qui vous permettront de lancer directement le serveur et le client (dans cet ordre), ce sont respectivement les fichiers **server.(sh/cmd)** et **client.(sh/cmd)**.

Multi-threading

Chaque client occupe un thread dans le serveur ce qui permet d'éviter une congestion par un unique utilisateur en attente.

RSA

Un chiffrement RSA a été développé pour que le client et le serveur communiquent de façon sécurisée. Bien qu'il puisse sembler un peu dérisoire de vouloir chiffrer ce genre d'information, j'ai trouvé que le principe était bon et que cela constituerait un bon exercice et je l'ai donc implémenté. Cependant, je crains le cruel manque d'efficacité de ma méthode pour convertir mes messages en nombres qui seront chiffrés (voir les méthodes **charSequenceToNumber** et **numberToCharSequence** du fichier **utils.py**).
La méthode de chiffrement utilise l'algorithme de Rabin-Miller pour déterminer si un grand nombre est premier (ce qui est obligatoire pour obtenir des clefs).
Au lancement du fichier **server.py**, une clef publique et une clef privée seront générées. Il en est de même pour le fichier **client.py** : à son lancement, une clef publique et une clef privée seront générées. Lorsque le client se connectera au serveur, il récupérera la clef publique du serveur afin d'envoyer plus tard, des messages chiffrés. Juste après, le client enverra sa clef publique afin que le serveur lui envoie des messages chiffrés.

Le problème du calcul de l'âge

Malheureusement, le calcul de l'âge ne peut pas toujours être exact. Et pour cause le manque de renseignement. En effet, si vous êtes né en fin d'année, le programme risque de vous attribuer un âge trop élevé (de 1 exactement). Une estimation en fonction du mois actuel permet de limiter ce genre d'erreur. C'est la raison pour laquelle vous allez être

amené à voir un pourcentage à côté de votre âge qui correspond à la probabilité qu'a le programme d'être juste le jour où vous l'utilisez.

Tests unitaires

Vous trouverez dans les fichiers **utils.py** et **rsa.py** des tests unitaires simples qui permettent de confirmer la justesse de certains algorithmes contenus dans ces même fichiers.