

# Rapport projet SECI5

## « Client/Server file system storage »

### Mr. Absil

**Nguyen Khanh-Michel**  
**Section industrielle**  
**2019-2020**

## Table des matières

Description de l'application.....	3
Mode d'emploi .....	4
Explication du (dé)chiffrement asymétrique dans le projet .....	9
Explication du (dé)chiffrement symétrique dans le projet .....	10
Points présents et manquants dans le projet .....	13
Mise en place du projet.....	14

## Description de l'application

### **Description**

L'application a pour but d'implémenter un système sécurisé de stockage de fichier entre un client et un serveur en LAN. Les fichiers ne sont pas censés être récupéré par d'autres personnes. C'est là que la sécurité intervient, tout ce que le client envoie est sécurisé, c'est-à-dire son nom d'utilisateur, son mot de passe, ses fichiers sont donc tous cryptés. Le langage de programmation utilisé pour réaliser le projet est le Java sous NetBeans 11.2.

## Mode d'emploi

Il y a trois projets principaux qui forment le projet global :

- 1) « Security\_Client »
- 2) « Security\_Common »
- 3) « Security\_Server »

Les projets que nous allons lancer sont le projet « Security\_Server » et le projet « Security\_Server ». Le projet « Security\_Common » va regrouper la partie dite sécurité ainsi que la partie des différents types de « Message » du client et du serveur.

Le scenario suivant montre comment lancer le projet. Mais avant cela, nous devons tout d'abord spécifier où nous voulons stocker les différents fichiers des clients. Pour ce faire, nous devons aller dans la classe « ThreadServer » et on modifie la variable destination en donnant le chemin voulu :

```
public class ThreadServer extends Thread {
    private final NetworkServer server;
    private String user;
    private boolean connected;

    private static final String destination= "C:\\Users\\ng-20\\Desktop\\examSECIS"; //it's the path that will store all the users's files/directories.
    private static final String separator=FileSystems.getDefault().getSeparator();
```

Ensuite on lance le serveur (Main.java) dans le projet « Security\_Server » :

```
run:
-----server listenning at port 39000-----
```

On peut désormais lancer le projet du client (MainClient.java) :

```
run:
Generating secret key
Generating private and public key
Sending secret and public key to the server

=====

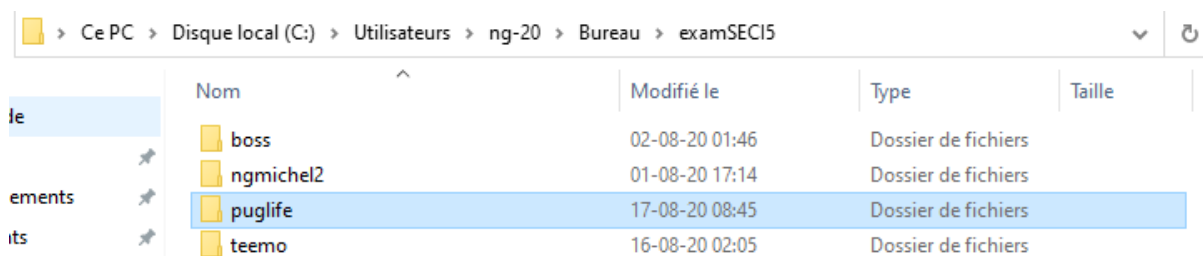
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
|
```

Il faut à présent se connecter ou s'inscrire sur le serveur pour pouvoir upload, delete ou download des fichiers.

Commençons donc par le scénario de l'inscription :

```
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
registration
Enter your user name :
puglife
Enter your password :
ilovepug
registred
you are registered
=====
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
|
```

Nous pouvons voir qu'un dossier pour l'utilisateur « pugLife » a été créé :



	Nom	Modifié le	Type	Taille
	boss	02-08-20 01:46	Dossier de fichiers	
	ngmichel2	01-08-20 17:14	Dossier de fichiers	
	puglife	17-08-20 08:45	Dossier de fichiers	
	teemo	16-08-20 02:05	Dossier de fichiers	

Une fois l'inscription faite, nous devons d'abord nous connecter afin de pouvoir download ou upload des fichiers :

```
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
connect
Enter your user name :
puglife
Enter your password :
ilovepug
connected
connected
=====
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
```

Essayons d'upload un fichier se trouvant sur notre Bureau :

```

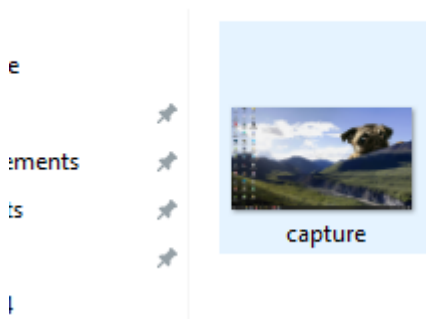
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
upload
Enter the file name with his absolute path :
C:\Users\ng-20\Desktop\capture.png
=====

Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
|

```

Comme nous pouvons le voir, l'image a bien été transféré vers le dossier de l'utilisateur :

📁 > Ce PC > Disque local (C:) > Utilisateurs > ng-20 > Bureau > examSEC15 > puglife



Ensuite, essayons de download un fichier de l'utilisateur se trouvant dans le serveur vers notre PC domicile :

```

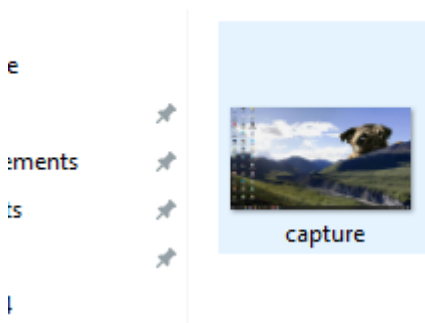
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
download
Enter the fileName :
capture.png
Enter the destination directory :
C:\Users\ng-20\Desktop\MyComputerAtHome
=====

Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
|

```

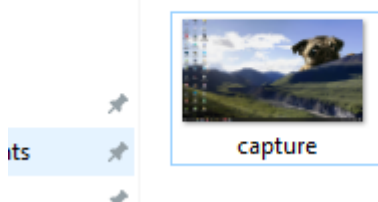
Le fichier que l'utilisateur veut download se trouvant dans le serveur :

Ce PC > Disque local (C:) > Utilisateurs > ng-20 > Bureau > examSECI5 > puglife



Le download a bien été fait et le fichier se trouve bien dans le PC de l'utilisateur :

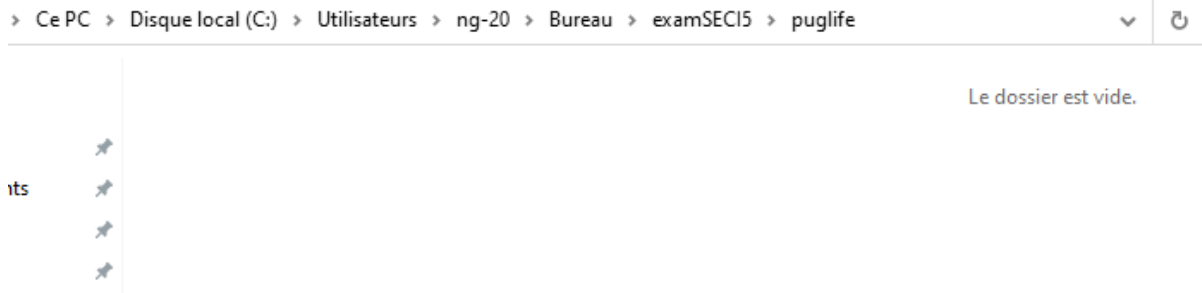
Ce PC > Disque local (C:) > Utilisateurs > ng-20 > Bureau > MyComputerAtHome



Maintenant, essayons de supprimer un des fichiers de l'utilisateur se trouvant dans le serveur :

```
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
delete
Enter the file name that you want to delete :
capture.png
deleted
=====
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
|
```

On voit que le fichier a bien été supprimé :



La commande « help » va permettre de lister toutes les commandes disponibles avec ou sans les arguments attendus :

```
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
help
disconnect: command without param
help: syntax is: help without param
download: syntax is: download filename and then destinationDirectory
upload: syntax is: upload filename(absolute path)
registration: syntax is: registration name and then password
delete: syntax is: delete filename(file name)
connect: syntax is: connect name and then password
=====
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
```

Et pour finir, si le client veut se déconnecter du serveur, il entre la commande « disconnect » :

```
Enter a command : (registration,connect,disconnect,download,upload,delete,deleteUser,help)
disconnect
BUILD SUCCESSFUL (total time: 12 minutes 22 seconds)
|
```



## Explication du (dé)chiffrement asymétrique dans le projet

Le (dé)chiffrement asymétrique est utilisé lors de l'envoi de la clé secrète d'un client au serveur. Voici comment le scénario se produit :

- Lorsque le serveur reçoit un client (dans son port), ce dernier va générer les différentes clés nécessaires. Notamment, sa clé publique/privée ainsi qu'une clé secrète commune aux deux sujets.
- Le client va alors chiffrer la clé secrète avec la clé publique du serveur et va l'envoyer à ce dernier.
- Le serveur va ensuite recevoir cette clé secrète qu'il aura déchiffrer avec sa clé privée.
- Le client ainsi que le serveur pourront enfin, chiffrer leur message avec la clé secrète.

Ce mécanisme est appelé le « Key Wrapping » et ses avantages sont qu'il est rapide lorsque le chiffrement asymétrique est uniquement utilisé sur la clé secrète. L'autre avantage est qu'il est sécurisé mais seulement si le serveur possède déjà la clé publique du client.

## Explication du (dé)chiffrement symétrique dans le projet

Pour toutes les fonctionnalités disponibles pour le client, le (dé)chiffrement symétrique sera utilisé. En effet, lorsque le client fera sa demande/son envoi de données au serveur, ils seront d'abord signés avec la clé privée du client et ensuite chiffrés avec la clé secrète.

De l'autre côté, lorsque le serveur recevra le message du client, il le déchiffrera avec la clé secrète et ensuite, il le vérifiera si celui-ci a bien été signé par le bon client en question (avec la clé publique du client). En fonction du message contenu envoyé par le client, le serveur va lancer soit :

- L'inscription ;
- La connexion ;
- Le download d'un fichier ;
- L'upload d'un fichier ;
- Le delete d'un fichier ;

Par la suite, le serveur va reproduire le même scénario initial fait par le client : il signe son message avec sa clé privée et ensuite le chiffrera avec la clé secrète. Côté client, il reçoit le message du serveur, le déchiffrera avec la clé secrète et vérifie s'il a bien été signé avec la clé publique du serveur.

### **Inscription**

Lorsque le client désire s'inscrire, le serveur demande un nom d'utilisateur et ensuite un mot de passe est demandé. Si le client a bien respecté ce que le serveur lui a demandé, le scénario suivant se met en jeu :

- Il hash et sale le mot de passe décrypte et le stocke dans sa base de données.
- Le sel y est également stocké.
- Il stocke dans sa base de données le nom du client.

## **Connexion**

Lorsque le client désire se connecter à son compte, le serveur lui demande son nom d'utilisateur et son mot de passe. Une fois les deux informations entrées, le scénario suivant se met en jeu :

- Il cherche le sel du client dans sa base de données grâce à son username.
- Une fois qu'il la trouve, il hash et sale le mot de passe décrypté.
- Il compare le résultat avec ce qu'il y a comme mot de passe hashé et sale dans la base de données. Si les deux sont pareils, l'authentification est réussie.

## **Upload de fichiers**

Lorsque l'inscription ou l'authentification a été réussie, le client peut désirer de upload un fichier. Quand il choisit cette fonctionnalité, le scénario suivant se met en jeu :

- Il rentre le chemin absolu du fichier (depuis son propre PC) qu'il souhaite envoyer au serveur.
- Une fois que c'est fait, le dossier est parcouru récursivement et les fichiers sont envoyés un par un au serveur.

## **Download de fichiers**

Lorsque l'inscription ou l'authentification a été réussie, le client peut désirer de download un fichier. Quand il choisit cette fonctionnalité, le scénario suivant se met en jeu :

- Il donne le nom du fichier qu'il désire avoir.
- Il va ensuite donner le chemin vers où il souhaite déposer son fichier (vers son PC de domicile par exemple).
- Une fois que c'est fait, le fichier désiré est envoyé au client.

(Le client ne voit jamais de fichiers cryptés, que ce soit pour l'upload ou du download)

## **Delete de fichiers**

Lorsque l'inscription ou l'authentification a été réussie, le client peut désirer de delete un fichier. Quand il choisit cette fonctionnalité, le scenario suivant se met en jeu :

- Le client donne le nom du fichier qu'il désire supprimer dans le serveur.
- Le fichier sera ensuite supprimé du serveur ainsi que dans la base de données.

## Points présents et manquants dans le projet

### **Points manquants :**

Mon projet n'a pas été développé à 100% comme demandé dans les énoncés de l'examen. Les raisons à cela sont surtout par manque de temps. Mais malgré cela, je pense que l'essentiel a pu être mis en place (au niveau de la sécurité). Voici les points qu'il manque dans le projet :

- La hiérarchie et donc la présence entre un admin et un « simple » utilisateur.
- Les différents droits des dossiers des utilisateurs dans le serveur.
- La fonctionnalité de pouvoir supprimer un utilisateur via une commande (fonctionnalité utilisée par l'admin).

### **Points présents :**

- Les différentes fonctionnalités qui sont disponibles (s'enregistrer, se connecter, se déconnecter, download/upload un fichier, supprimer un fichier, commande d'aide).
- La mise en place d'un système de sécurité dans le projet par l'utilisation de ce que j'ai appris dans le cours (mot de passe, clés cryptographiques, etc...).
- La mise en place d'une base de données.

## Mise en place du projet

### Client/serveur

En troisième année, avec Monsieur Lechien pour le cours de Java, nous avons codé et implémenté des bases permettant de pouvoir communiquer en LAN entre un serveur et un client. La partie client/serveur de ce projet a donc été grandement inspiré des différents labos faits en classe, ce qui m'a beaucoup aidé durant la création du projet en sécurité.

### Base de données

En ce qui concerne de la base de données, celle-ci a également été vu dans le cours d'Ateliers logiciels V (Java) avec Monsieur Lechien. Nous avons dû créer une base de données dans les labos mais aussi dans les projets faits à domicile et aux interros. La partie de base des données de ce projet a donc également été grandement inspiré des différents travaux de cette année en Atelier logiciels V.

Dans le projet, il existe 2 bases de données : ClientDB et Data\_ClientDB.

### Sécurité

La sécurité étant un nouveau concept pour moi, celle-ci m'a demandé beaucoup d'efforts. Cette partie était la plus intéressante étant donné qu'il s'agit du sujet principal du cours. Cela m'a donc permis de découvrir les différentes techniques de sécurité, choses qui sont très importantes dans le monde informatique. J'ai pu réaliser ce projet grâce aux slides du cours de Monsieur Absil et à ses explications, des différentes sources sur internet m'ont comme par exemple : <https://stackoverflow.com/>, <https://www.mkyong.com/>, <https://www.geeksforgeeks.org/>, <https://fr.wikipedia.org/>, <https://docs.oracle.com/javase/7/docs/api/overview-summary.html>. Il y a aussi des vidéos explicatives sur les sujets du cours sur YouTube qui m'ont permis de m'éclaircir sur plusieurs points que je ne comprenais pas.

### **Choix du langage de programmation**

J'ai décidé de choisir le Java car il s'agit de tout simplement du langage dont j'ai le plus d'acquis et de connaissances. J'ai également pu trouver qu'il était totalement possible d'implémenter la partie sécurité dans ce langage et qu'il existe des bibliothèques spécifiques à cela dans Java et qui étaient en rapport avec le cours de Monsieur Absil.