RUIZ Adrien
L3 Info
Réseaux 3

Institut National Universitaire Champollion

Projet Chat

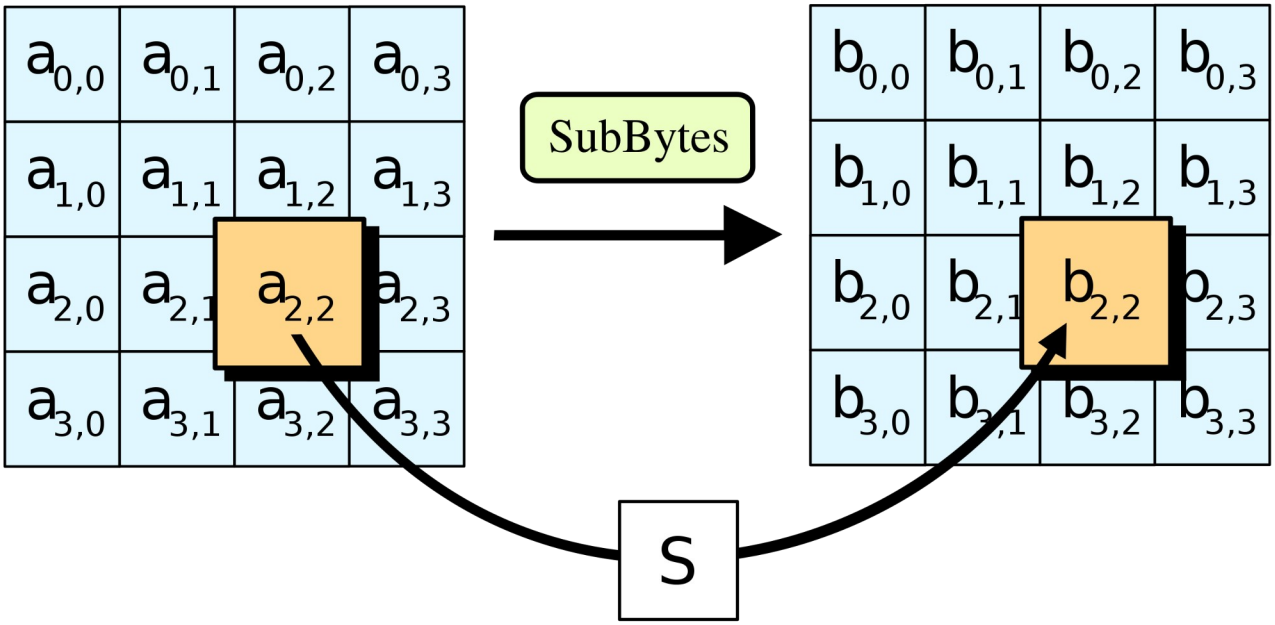


Figure 1: SubBytes operation for AES

Sommaire

1. Introduction

1. État du projet

Voici la liste des fonctionnalités implémentées :

- **N client** peuvent communiquer de manière **simple et sans interruption.**
- **Interface graphique** clair et simple
- **Salons *(Les anciens messages ne sont pas transmis)***
- **Cryptage/décryptage** à l'aide de **AES**
- Transmission de la clef AES de **manière sécurisée**

2. Outils utilisés

Lors de ce Projet, J'ai utilisé différents outils. En voici la liste détaillée :

- **Chat GPT** : Je l'ai utilisé :
  - À des fins d'organisation. Par exemple, c'est lui qui a créé le README.md.
  - De générer certains tests.
- **Maven** :Outils de gestion de projet. Il permet de simplifier grandement la gestion de librairie et m'a permis de générer un .jar exécutable.
- **Scene Builder** :Outils de création d'interface graphique pour javaFX.
- **PlantUML :** Outils de création de diagramme UML/execution.
- **SQLite** :SGBDD local, il me sert à **stocker les messages** dans une BDD local.

3. Librairies utilisées

Voici la liste des librairies utilisées :

- **Junit 5** :Librairie de tests unitaires.
- **Jackson** :Librairie servant à manipuler les fichiers json. Elle m'a servie afin d'exporter les résultats des cryptages.
- **JavaFX** :Librairie d'interface utilisateur java utilisant le modèle MVC. Elle m'a permis de créer une interface graphique via un fichier de balisage FXML (Format utilisé par Scene Builder).
- **Log4j2** :Outils de gestion de logs. J'ai souhaité l'utiliser afin d'apprendre à gérer les logs.

2. Détails du code

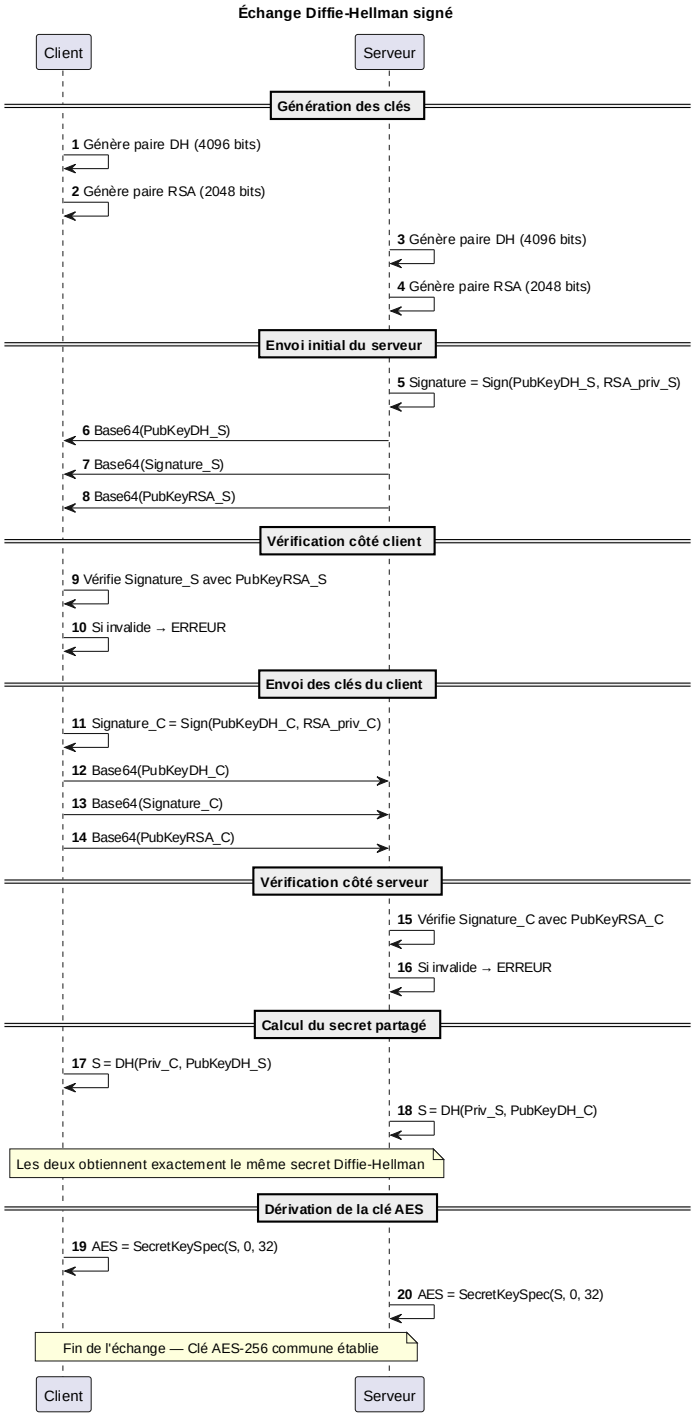
1. Transmission de la clef AES

Afin de sécuriser la transmission de la clef AES, j’ai implémenté l’algorithme **d’échange de clés Diffie-Hellman.** Il permet de partager un secret sans transmettre le dit secret.

Il y donc **une clef par client**.

De plus, lors des échanges de clefs Diffie-Hellman, elles sont **signées par une clef RSA**

En voici le diagramme d’échange :



2. Thread

Lors de la connexion d’un client au serveur, un nouveau thread est crée qui est dédié à lui.

Donc **Pour N client il y aura N thread**.

Côté client, il y 2 threads :

- Thread d’écoute
- Thread d’envois

3. Salons

Côté serveur, les salon sont représenté par un dictionnaire :

- **Clef** : Nom du salon
- **Valeur** : L’ensemble des clients du dis salon

Pour qu’un client puisse rejoindre un salon, il doit envoyer un message de type **COMMAND** et son contenu est le nom du dit salon. Si il n’existe pas, il sera créé.

4. Tests unitaires

Comme dit dans l’introduction, j’ai utilisé ***Junit 5*** pour gérer mes tests unitaires.

Dans ce projet, il n’y a pas beaucoup de tests unitaires. Les seules classes tester sont :

- **DBHandler**
- **Message**
- **CryptoHandler**

5. Diagramme UML

Voici le diagramme UML du projet :