Master Logiciel sûr
Cours M1 Programmation réseau

Projet Programmation Réseau

Système de messagerie chiffrée de bout en bout

Réalisé par :

CHERIFI Yidir
DEMOULIN Jean-Gabriel
HADJ AISSA FEKHAR Oussama

A Créteil le 11/04/2025

Encadré par :

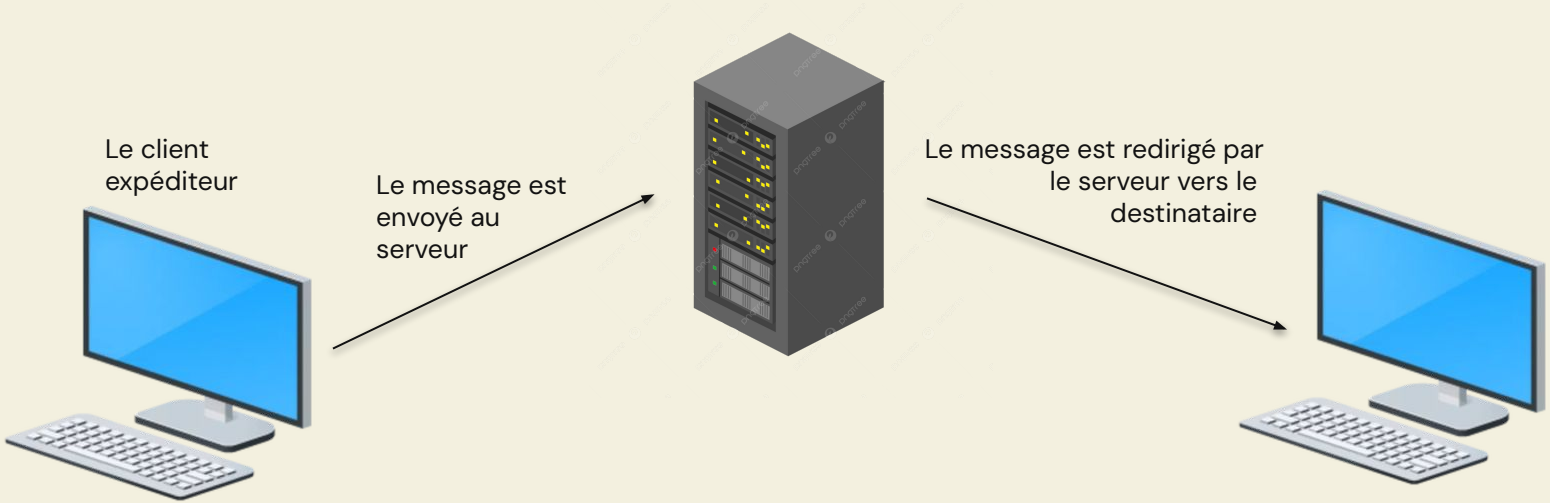
Sovanna Tan

PLAN

1 - Architecture globale
2 - Bibliothèques utilisées
3 - Serveur
4 - Client
5 - Chiffrement et déchiffrement
6 - Démonstration

Architecture globale

Notre code est découpé en 3 fichiers principaux : 1 pour le serveur, 1 pour le client et un pour les fonctions de chiffrement.



Bibliothèques utilisées

2 Bibliothèques principales :

- Socket : Cette bibliothèque permet l'utilisation de l'appel système epoll, qui est à la base de la gestion des demandes au serveur.

- Cryptography : Cette bibliothèque permet le chiffrement (et le déchiffrement) de nos messages de différents manières, dans ce projet nous utilison RSA.

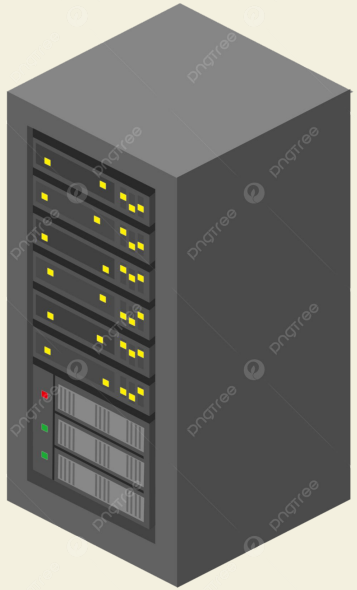
Mais aussi d'autres bibliothèques auxiliaires :

- Os
- Threading
- Tkinter
- Json

Serveur

1- Fonctionnalités principales :

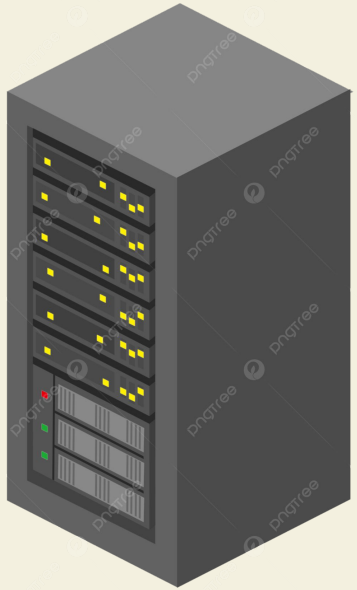
- Gestion des connexions des clients via des sockets.
- Enregistrement des utilisateurs et stockage de leurs clés publiques.
- Diffusion de la liste des utilisateurs connectés à tous les clients.
- Relais des messages chiffrés entre les clients.
- Journalisation des activités (logs) pour le débogage et la surveillance.



Serveur

2- Architecture :

- Utilisation d'une interface graphique (Tkinter) pour afficher les logs en temps réel.
- Boucle principale basée sur epoll pour une gestion efficace des connexions multiples.
- Structure de données pour stocker les informations des clients (clients et clients_info).
- Utilisation des objets JSON pour communiquer avec les clients



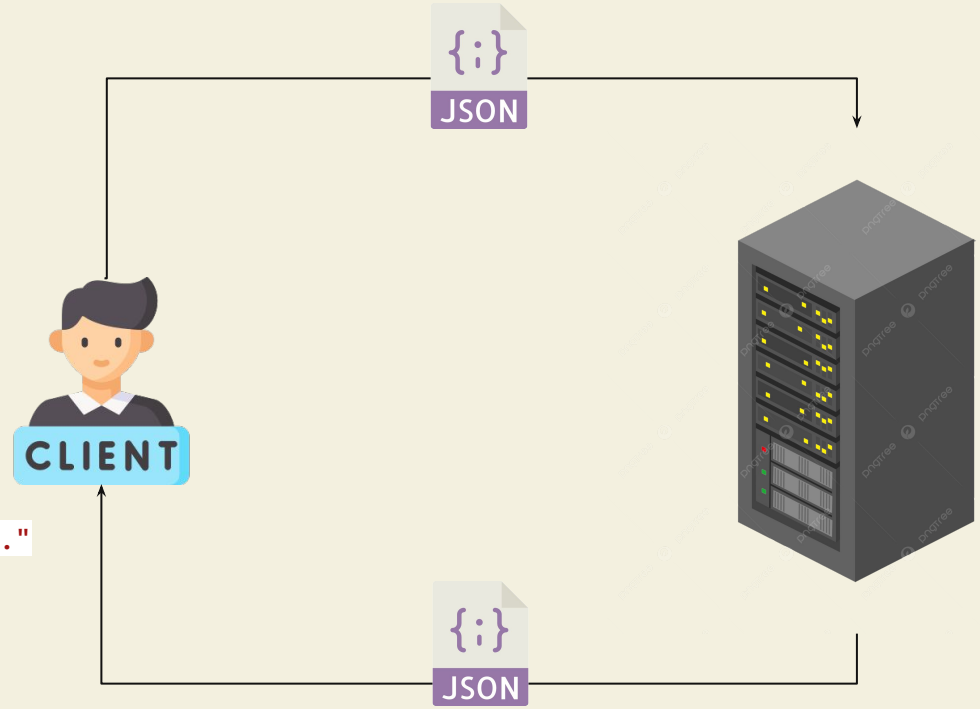
Serveur

Requête Client → Serveur

```
{
  "action": "get_pubkey",
  "target": "Bob",
  "from": "Alice"
}
```

Réponse Serveur → Client

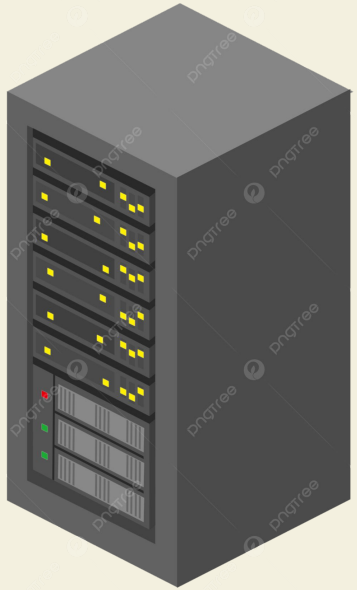
```
{
  "action": "pubkey",
  "from": "Bob",
  "pubkey": "-----BEGIN PUBLIC KEY-----..."
}
```



Serveur

3- Sécurité:

- Le serveur ne déchiffre pas les messages, il sert uniquement de relais.
- Les clés publiques des clients sont stockées et partagées uniquement sur demande pour établir des communications sécurisées.



Client

1- Fonctionnalités principales :

- Interface graphique (Tkinter) pour interagir avec l'utilisateur.
- Connexion au serveur et enregistrement avec un nom d'utilisateur et une clé publique.
- Génération de paires de clés RSA et échange sécurisé de clés AES avec d'autres clients.
- Chiffrement et déchiffrement des messages à l'aide des clés partagées.

Client

2- Architecture :

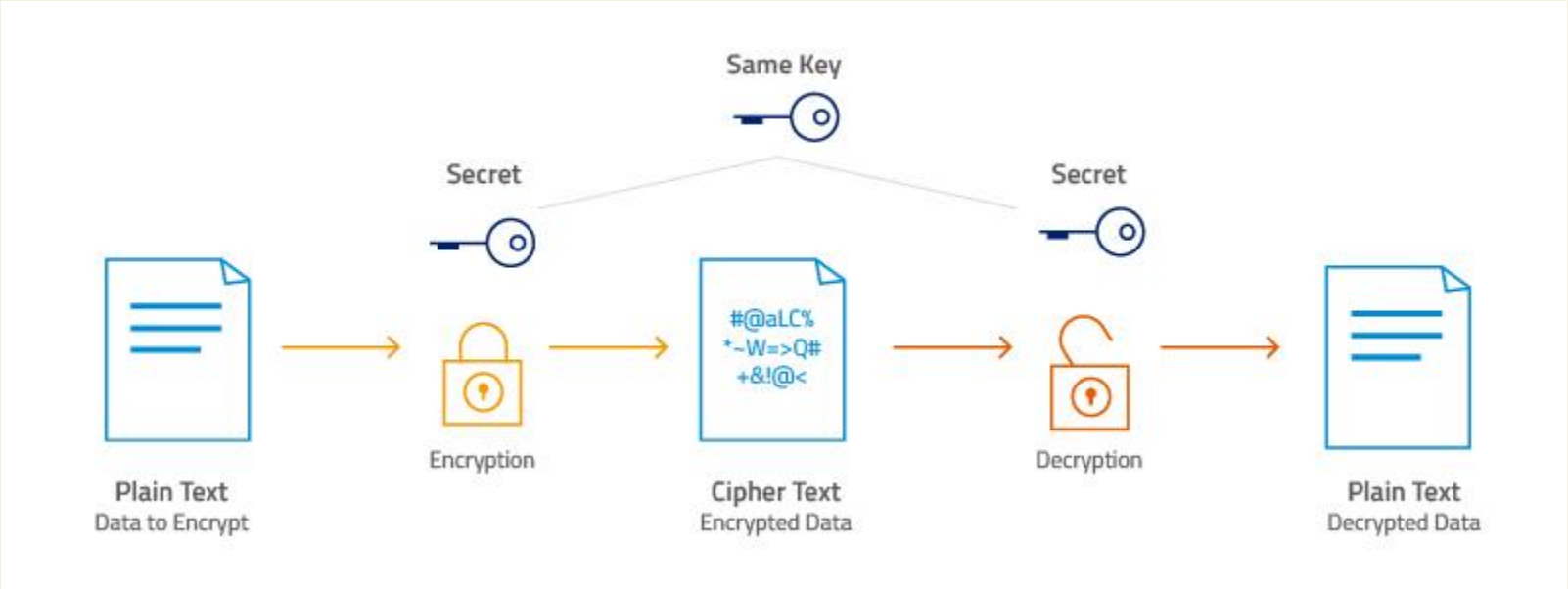
- Interface utilisateur divisée en zones : liste des utilisateurs, historique des messages, champ de saisie.
- Thread séparé pour recevoir les messages du serveur sans bloquer l'interface.

Client

3- Sécurité:

- Chiffrement des messages avec AES et échange sécurisé des clés via RSA.
- Les clés privées ne quittent jamais le client.

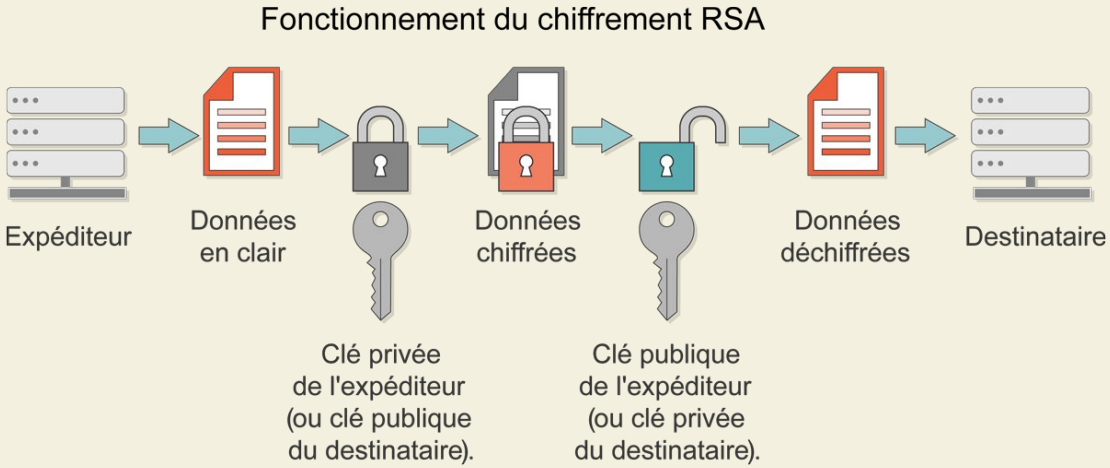
AES


Same Key
Secret
Secret
#@aLC%
*~W=>Q#
+&!@<
Encryption
Decryption
Plain Text
Data to Encrypt
Cipher Text
Encrypted Data
Plain Text
Decrypted Data

Chiffrement et déchiffrement

Le chiffrement se fait côté client. Lors de la première connexion, une paire de clés (publique/privée) est générée.

Pour sécuriser l'envoi de la clé AES, le client chiffre les données avec la **clé publique** du destinataire.
Le destinataire les déchiffre ensuite avec sa **clé privée**.



Extrait Wireshark

547	852.098019	192.168.1.101	192.168.1.100	TCP	183 57636 → 12345 [PSH, ACK] Seq=1529 Ack=560 Win=64768 Len=129
548	852.153494	192.168.1.100	192.168.1.101	TCP	60 12345 → 57636 [ACK] Seq=560 Ack=1658 Win=1049600 Len=0

```
7c 8a e1 9c 4e 1d 7c c2  c6 42 f4 a0 08 00 45 00   |···N·|· ·B····E·
00 a9 11 32 40 00 80 06  00 00 c0 a8 01 65 c0 a8   ···2@··· ·····e··
01 64 e1 24 30 39 d6 66  e7 5a 42 f5 b5 1e 50 18   ·d·$09·f ·ZB···P·
00 fd 84 b5 00 00 7b 22  61 63 74 69 6f 6e 22 3a   ·····{" action":
20 22 6d 65 73 73 61 67  65 22 2c 20 22 74 6f 22    "messag e", "to"
3a 20 22 79 69 64 69 72  22 2c 20 22 66 72 6f 6d   : "yidir ", "from
22 3a 20 22 4a 47 22 2c  20 22 63 6f 6e 74 65 6e   ": "JG",  "conten
74 22 3a 20 22 32 65 38  38 62 36 34 64 31 39 62   t": "2e8 8b64d19b
33 63 35 37 62 64 38 31  65 33 36 62 30 33 65 39   3c57bd81 e36b03e9
62 37 32 35 37 39 65 33  61 62 35 63 32 33 64 32   b72579e3 ab5c23d2
31 35 32 32 38 34 66 32  38 31 35 65 32 36 32 38   152284f2 815e2628
65 65 62 37 37 22 7d                               eeb77"}
```

Démonstration