Documentation technique — SAE 3.02

Routeur virtuel distribué avec anonymisation

Ce document détaille l'architecture, la structure et le fonctionnement du système de routage en oignon. Si vous cherchez à installer, comprendre ou tester le projet, vous êtes au bon endroit.

1. Ce qu'on livre avec ce projet

Les fichiers

master.py — Le cerveau du système — supervise routeurs et clients, interface PyQt et accès BDD

routeur.py — Chaque routeur virtuel — déchiffre une couche d'oignon et relaye au suivant

client.py — L'interface client — envoie et reçoit des messages via la chaîne de routeurs

README.txt — Guide d'installation et de lancement

sae3.02.pdf — Documentation technique complète

Le cahier des charges : coché ✓

✓ Architecture distribuée (master, routeurs, clients) | ✓ Routage multi-sauts | ✓ Anonymisation | ✓ Base de données | ✓ Documentation complète

2. Fonctionnalités : ce qui marche

- Architecture distribuée TCP entre le master, les routeurs et les clients
- Routage en oignon à plusieurs sauts — chaque routeur voit juste l'étape suivante
- Chiffrement XOR symétrique avec une clé propre à chaque routeur
- Enregistrement dynamique en MariaDB avec suivi du champ 'alive'
- Interfaces PyQt5 pour le master (supervision) et le client (envoi/réception)
- Enregistrement et démarrage automatisés

Limites assumées

Pas de chiffrement industriel (AES, TLS), pas d'authentification forte.

3. Architecture et structure

L'arborescence est simple et intentionnelle :

```
SAE3.02/
├── master.py          (master + GUI + base de données)
├── routeur.py         (routeur virtuel)
├── client.py          (client + GUI)
├── README.txt         (installation et utilisation)
└── sae3.02.pdf        (documentation technique)
```

Dépendances

- Python 3.10+
- PyQt5 pour les interfaces
- mysql-connector-python pour accéder à MariaDB
- Bibliothèques standard : socket, threading, time

Protocole de communication

REGISTER_ROUTER — Les routeurs se déclarent auprès du master

REGISTER_CLIENT — Les clients se déclarent aussi

ASK_ROUTERS — Les clients demandent la liste des routeurs actifs

Messages en oignon — Le message encapsulé dans plusieurs couches de chiffrement

Flux d'un message

1. 1. Le client construit un oignon en chiffrant le message pour chaque routeur de sa route
2. 2. Le premier routeur reçoit l'oignon complet et déchiffre sa couche
3. 3. Il extrait l'adresse du routeur suivant et retransmet
4. 4. Chaque routeur fait pareil : déchiffre, découvre le suivant, retransmet
5. 5. Le dernier routeur livre le message au client destinataire

4. Chiffrement : comment ça marche

On utilise XOR symétrique : simple, léger, sans dépendances externes. La clé est une chaîne fournie par l'administrateur, répétée de manière cyclique sur les données.

5. Installation et lancement

Prérequis :

- Python 3.10+
- PyQt5 et mysql-connector-python
- MariaDB configurée avec les tables nécessaires

Après avoir adapté les paramètres réseau et BDD dans les scripts :

6. Démarrer le master
7. Lancer les routeurs
8. Démarrer les clients

6. Conclusion

Ce projet implémente une architecture complète de routage en oignon distribuée. Malgré les simplifications volontaires sur la sécurité, le système fonctionne, démontre les concepts clairement, et constitue une base extensible pour des explorations futures.