FICHE INDIVIDUELLE DE PROJET

Nom / Prénom : Rayan SAOUD **Groupe :** RayAnit **Projet :** OnionRouter (R3.09 / SAÉ 3.02)
Rôle : Responsable Cœur Réseau, Cryptographie & Automatisation

1. Introduction : Mon rôle

Dans ce binôme, je me suis chargé de la partie **"Moteur & Sécurité"**. Pendant qu'Arjanit gérait l'interface et la base de données, ma mission était de construire les "tuyaux" du réseau. Je devais m'assurer que les données pouvaient voyager d'une machine à l'autre de façon chiffrée et fluide, sans utiliser de solutions de facilité (librairies toutes faites).

2. Réalisations Techniques

Mon travail s'est découpé en trois gros morceaux :

A. La Cryptographie (Le défi RSA)

C'était la partie la plus dure. Comme `cryptography` était interdit, j'ai recodé RSA.

- J'ai coder la génération de clés (calcul des exposants **e** et **d**) en utilisant des maths modulaires.
- J'ai surtout dû régler le problème de la taille des messages : le RSA de base ne peut pas chiffrer un long texte d'un coup. J'ai donc créé un système de **"Chunking"** (découpage) qui coupe le message en petits blocs, les chiffre un par un, et les recolle. Sans ça, le projet ne marchait pas pour les vrais messages.

B. Le Réseau (Sockets & Threads)

- J'ai codé le serveur des Routeurs avec les `sockets` Python.
- Pour que le routeur ne "gèle" pas quand il calcule une clé, j'ai utilisé des **Threads**. Chaque client est géré dans son propre processus, ce qui permet au routeur de faire plusieurs choses en même temps.
- J'ai créé un protocole maison avec des séparateurs `|||` pour remplacer le JSON (interdit), afin que le routeur sache distinguer l'IP suivante du message chiffré.

C. L'Automatisation

- J'ai écrit le script `start_routers.sh` pour lancer toute l'infrastructure d'un coup sur Linux.
- J'ai géré les signaux systèmes : quand on fait CTRL+C, le routeur intercepte l'ordre et envoie un message d'adieu au Master avant de s'éteindre.

3. Analyse des compétences (AC / CE)

Voici comment mon travail valide les compétences du semestre :

- **AC23.01 (Automatiser l'administration système) :**
  - *Preuve :* J'ai validé cette compétence en créant le script Bash qui déploie automatiquement les nœuds routeurs. J'ai aussi géré l'arrêt propre des processus (gestion du signal `SIGINT`) pour ne pas laisser de ports bloqués sur la machine.
- **AC23.03 (Programmer une application client/serveur) :**
  - *Preuve :* C'est le cœur de mon code `router/main.py`. J'ai mis en place une architecture TCP non-bloquante capable de gérer des connexions simultanées grâce au multi-threading, ce qui est indispensable pour un nœud de réseau.
- **CE3.05 (Intégrer les problématiques de sécurité) :**
  - *Preuve :* J'ai appliqué le principe de sécurité par l'architecture ("Security by Design"). En implémentant le routage en oignon, j'ai garanti qu'aucun nœud ne possède l'information complète (qui parle à qui). Le chiffrement RSA manuel m'a forcé à manipuler concrètement les concepts de clés publiques/privées.
- **CE3.04 (Choisir les outils de développement adaptés) :**
  - *Preuve :* J'ai choisi de séparer mon code en modules réutilisables (`crypto_utils.py`, `protocol.py`) pour qu'Arjanit puisse les utiliser dans son interface sans modifier mon code. L'usage de Git a permis de fusionner nos travaux sans conflit.

4. Conclusion sur mon apprentissage

Ce projet m'a fait passer un cap technique. Avant, le chiffrement RSA était une "boîte noire" magique pour moi. Maintenant que j'ai dû le coder (et galérer avec les tailles de blocs), je comprends vraiment comment ça marche mathématiquement. J'ai aussi appris qu'un système distribué est fragile : il faut gérer les pannes, les déconnexions, et les différences entre Windows et Linux. C'était parfois frustrant, mais voir le message s'afficher à la fin sur le serveur après avoir traversé 3 routeurs virtuels était une vraie satisfaction.