RAPPORT DE PROJET FINAL

Conception d'une architecture distribuée avec routage en oignon

Membres du groupe **:**

- Rayan SAOUD
- Arjanit Shala

Lien Vidéo : https://youtu.be/nkGbUICb_v0

Groupe : RayAnit **Année :** 2025 **Module :** R3.09 & SAÉ 3.02

Lien GitHub : https://github.com/Ryn-s/Onion_Router

1. Introduction

Contexte et Objectif

Dans le cadre des modules R3.09 (Programmation Évènementielle) et SAÉ 3.02 (Développement d'applications communicantes), nous avons conçu et implémenté un système de communication sécurisé baptisé **"OnionRouter"**.

L'objectif principal était de reproduire le fonctionnement d'un réseau d'anonymisation type *Tor* (The Onion Router). Ce système permet à deux clients d'échanger des messages sans qu'aucun nœud intermédiaire ne puisse connaître à la fois l'identité de l'émetteur et le contenu du message.

Enjeux Techniques

Ce projet nous a confrontés à trois défis majeurs imposés par le cahier des charges :

1. **L'interdiction des librairies de cryptographie :** Nous avons dû recoder l'algorithme RSA (génération de clés, chiffrement modulaire) "à la main".
2. **L'architecture distribuée :** Le système devait fonctionner sur au moins 3 machines distinctes (Windows et Linux).
3. **L'automatisation :** La gestion du déploiement des nœuds devait être automatisée par scripts.

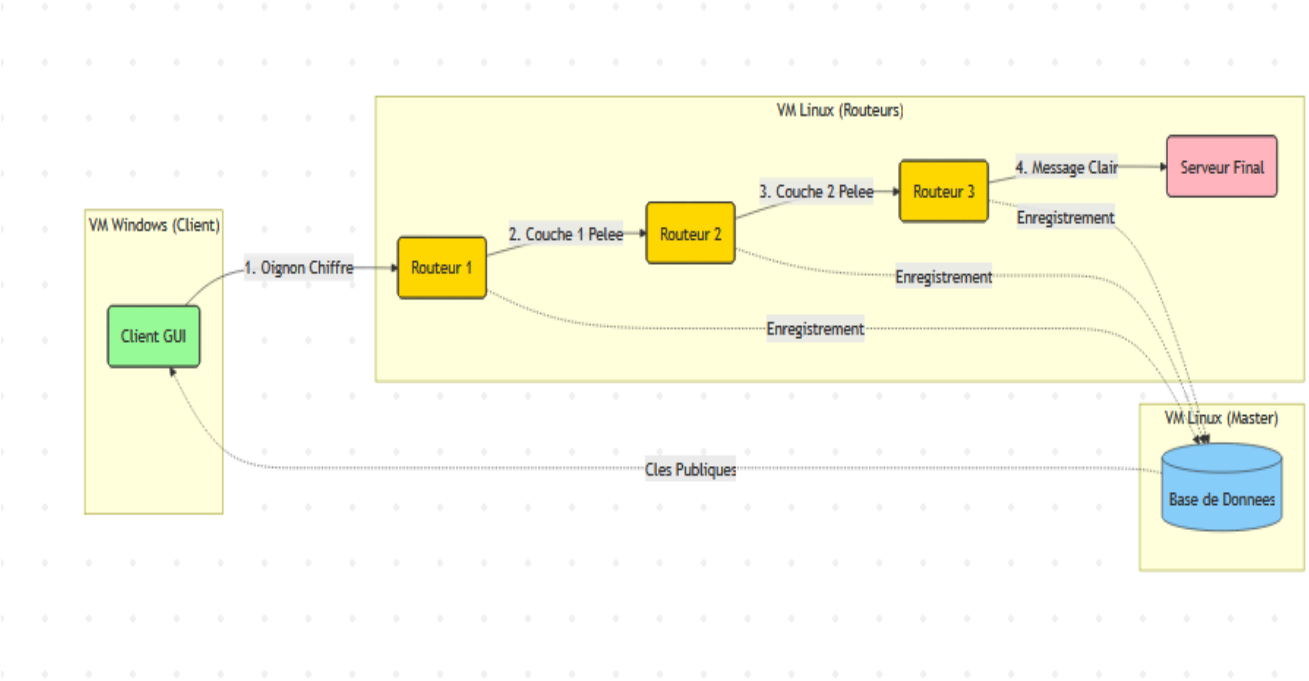
2. Architecture Technique

Notre solution repose sur une architecture distribuée stricte où chaque composant joue un rôle unique.

2.1 Schéma de l'infrastructure

L'infrastructure de production utilisée pour la validation est composée de trois machines physiques/virtuelles interconnectées en réseau pont (Bridge) :

1. **Machine VM Windows (Client) :** Interface utilisateur pour l'envoi de messages.
2. **VM Linux (Réseau de Transport) :** Héberge les 3 routeurs virtuels (R1, R2, R3) et le serveur de réception.
3. **VM Linux (Master) :** Héberge l'annuaire central (Base de données MariaDB) et le service d'enregistrement.



2.2 Rôle des composants

- **Le Master (Annuaire) :** Il est le point de vérité du réseau. Il stocke dans une base MariaDB la liste des routeurs actifs, leurs IP, ports et surtout leurs Clés Publiques RSA. Il dispose d'une interface graphique de supervision (`monitor.py`).
- **Les Routeurs (Nœuds) :** Ce sont des passeurs "stateless" (sans mémoire). Ils reçoivent un paquet chiffré, déchiffrent *une seule* couche avec leur clé privée, découvrent l'adresse du suivant et transmettent le reste. Ils ne connaissent jamais l'origine du message (Zero Knowledge).
- **Le Client :** C'est lui qui construit "l'oignon". Il récupère les clés publiques auprès du Master et chiffre le message en couches successives (pour R3, puis R2, puis R1).

3. Choix d'Implémentation (Points Forts)

Pour répondre aux contraintes pédagogiques, nous avons fait des choix techniques forts.

3.1 Cryptographie "Maison" & Chunking

L'interdiction d'utiliser la librairie `cryptography` nous a posé un problème de taille : l'algorithme RSA standard (scolaire) ne peut pas chiffrer un message dont la valeur numérique dépasse le module n (environ 128 caractères pour une clé 1024 bits).

Notre solution : Le "Chunking" **(Découpage par blocs)** Nous avons développé un algorithme qui :

1. Découpe le message clair (ou l'oignon) en blocs de taille fixe (TailleClé / 8 - marge).
2. Chiffre chaque bloc individuellement (C = M^e (mod n)).
3. Concatène les blocs chiffrés avec un séparateur (`/`). Cela permet de chiffrer des messages de taille arbitraire, condition indispensable pour le routage en oignon où la taille du paquet augmente à chaque couche.

3.2 Protocole de Transport Personnalisé

L'utilisation de JSON étant restreinte, nous avons conçu un protocole applicatif léger basé sur des délimiteurs (`|||`). Format d'une trame : `NEXT_HOP_IP:PORT |||` `PAYLOAD_CHIFFRÉ` Ce choix optimise le traitement par les sockets TCP (pas de parsing complexe) et réduit la latence.

3.3 Robustesse & Nettoyage Automatique

Un problème récurrent des systèmes distribués est la gestion des "nœuds fantômes" (routeurs crashés restant dans la BDD). Nous avons implémenté deux sécurités :

1. **Nettoyage au démarrage :** Le Master vide la table de routage à chaque lancement.
2. **Graceful Shutdown :** Lorsqu'un routeur est arrêté via `SIGINT` (Ctrl+C), il intercepte le signal et envoie une requête `UNREGISTER` au Master avant de s'éteindre.

4. Réponse au Cahier des Charges

Le tableau ci-dessous résume la conformité de notre livrable vis-à-vis de la grille d'évaluation.

Critère	État	Justification Technique
Routage Multi-sauts	✅ **Total**	Routage validé sur 3 sauts (Client -> R1 -> R2 -> R3 -> Serveur).
Chiffrement Asymétrique	✅ **Total**	RSA 1024 bits manuel avec gestion des blocs (Chunking).
Anonymisation	✅ **Total**	Principe du "Zero Knowledge" respecté. Preuve visuelle dans les logs.
Base de Données	✅ **Total**	Utilisation de MariaDB pour stocker clés et topologie.
Interface Master	✅ **Total**	Interface Qt (`monitor.py`) pour visualisation temps réel.
Interface Client	✅ **Total**	Interface Qt avec choix du mode (Aléatoire ou Manuel).
Automatisation	✅ **Total**	Script Bash (`start_routers.sh`) pour déploiement rapide.
Multi-threading	✅ **Total**	Routeurs capables de gérer plusieurs clients simultanément.

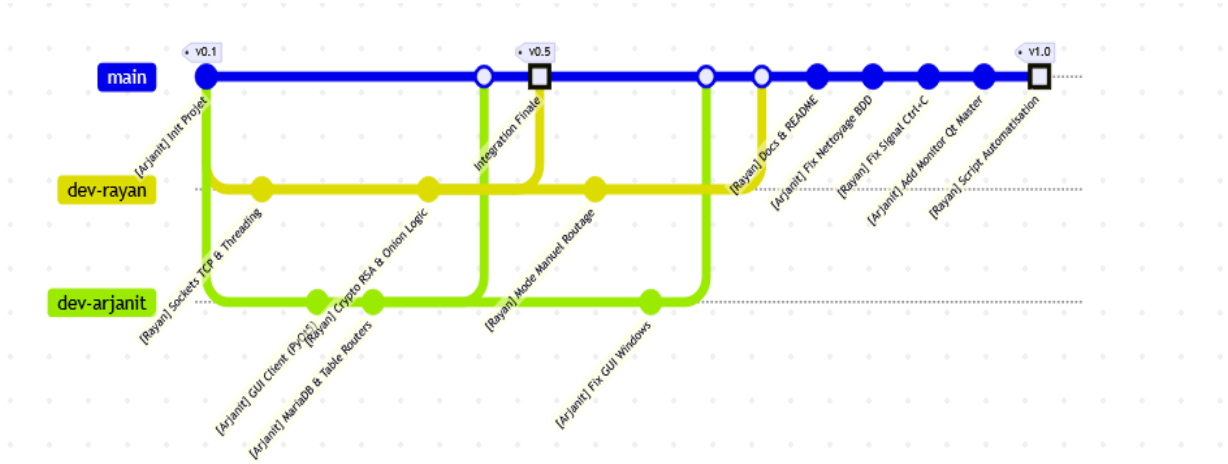
5. Gestion de Projet

Nous avons adopté une méthodologie agile simplifiée, séparant clairement les responsabilités pour avancer en parallèle sans conflit de code.

- **Rayan** : Responsable de la couche Transport, de la Cryptographie et de l'Automatisation.
- **Arjanit** : Responsable de l'Infrastructure centrale (Master/BDD) et des Interfaces Graphiques.

Preuve de collaboration

Nous avons utilisé Git pour versionner notre code. Le graphe ci-dessous montre l'évolution parallèle des fonctionnalités.



ANNEXE 3 : Journal de Bord - Projet OnionRouter

Ce document retrace l'historique des contributions majeures sur le dépôt Git.

Date	Auteur	Description du Commit
2025-12-30	Rayan	**Feat:** Script d'automatisation de déploiement (AC23.01) et logs de preuve d'anonymisation
2025-12-30	Arjanit	**Add:** Interface graphique de monitoring Master (Qt) et liste des dépendances
2025-12-30	Rayan	**Fix:** Envoi du signal UNREGISTER au Master lors de l'arrêt (Ctrl+C)
2025-12-30	Arjanit	**Fix:** Nettoyage auto de la BDD au démarrage et gestion désinscription routeurs
2025-12-28	Rayan	**Docs:** Mise à jour du README final et ajout du journal de bord
2025-12-28	Arjanit	**Update:** Correction des bugs d'affichage dans le GUI Client sur Windows
2025-12-24	Rayan	**Feat:** Ajout du mode 'Manuel' pour choisir le chemin des routeurs
2025-12-20	Rayan	**Intégration finale :** Connexion GUI Client avec le Core et correction bugs Master
2025-12-15	Rayan	**Implémentation :** Crypto RSA manuelle et Logique de Routage Oignon
2025-12-12	Arjanit	**Feat:** Création de la base de données MariaDB et table 'routers'
2025-12-10	Arjanit	**Feat:** Interface graphique de base pour le Client (PyQt5)
2025-12-05	Rayan	**Feat:** Sockets TCP de base et gestion du multi-threading pour les routeurs

2025-11-28	Arjanit	**Initialisation :** Structure projet, script BDD et serveur Master de base