Réponse au cahier des charges

Table des matières

Éléments implémentés et non implémentés

Le projet met en œuvre un système fonctionnel de routage en oignon permettant l'échange de messages entre clients via une chaîne de routeurs. Les fonctionnalités attendues dans le cahier des charges ont été implémentées et validées par des tests en conditions réelles (exécution locale sur plusieurs systèmes d'exploitations).

Le cœur du système repose sur un réseau de routeurs capables de recevoir, déchiffrer et relayer des paquets chiffrés selon le principe du routage en oignon. Chaque routeur ne déchiffre qu'une seule couche du message avant de le transmettre au suivant, garantissant ainsi la confidentialité du contenu et de la destination finale. Les clients disposent d'une interface graphique permettant de s'enregistrer dynamiquement, de choisir une route explicite ou aléatoire, puis d'envoyer des messages protégés par plusieurs couches de chiffrement.

La gestion des clés cryptographiques et des informations réseau est centralisée via une base de données SQL. Celle-ci stocke les clés publiques des routeurs, les adresses IP, les ports, ainsi que les clients enregistrés. Ce choix permet une architecture plus dynamique que des fichiers de configuration statiques, notamment pour l'ajout ou la suppression de routeurs sans modification du code client.

Certaines fonctionnalités prévues de manière théorique n'ont toutefois pas été implémentées. Le système ne gère pas l'anonymisation du trafic au niveau temporel (padding, délais aléatoires ou messages factices). De plus, l'authentification entités (certificats, signatures numériques) n'est pas présente : ainsi si quelqu'un parvient à ajouter ou remplacer une clé dans la base de données, le système lui fera confiance sans vérification supplémentaire, se qui n'est pas parfait. Enfin, le projet ne vise pas une résistance cryptographique élevé et est pour le moment pour un projet et non pas pour des entreprises (nous allons en parler plus tard).

Structure du code, modules, protocole et API interne

Le projet est structuré de manière modulaire afin de séparer clairement les responsabilités fonctionnelles. Le module central « rsa_outils.py » regroupe toute la logique liée à la cryptographie et à l'accès à la base de données. Il fournit une API interne unifiée pour la génération des clés, le chiffrement et le déchiffrement hybride, ainsi que pour l'enregistrement et la récupération des informations réseau (routeurs et clients). Cette centralisation permet d'éviter la duplication de logique cryptographique dans les autres composants.

Les routeurs, représentés par le script R1.py, implémentent un serveur TCP multi-thread. Chaque routeur est capable de gérer simultanément plusieurs connexions entrantes, de distinguer les messages de contrôle (enregistrement client ou encore arrêt d'urgence) des paquets de routage, puis d'appliquer le traitement cryptographique correspondant. Le protocole applicatif utilisé est simple et textuel afin de faciliter lea résolution de problèmes. Les paquets suivent un format strict composé de trois champs : la route restante, le destinataire final et la charge utile chiffrée.

Le client est implémenté sous la forme d'une interface graphique (ClientGUI.py) développée avec PyQt5. Cette interface permet à l'utilisateur de configurer dynamiquement son environnement réseau, de visualiser les logs techniques et les messages reçus, et de construire automatiquement un oignon de chiffrement avant l'envoi. La communication réseau côté client repose également sur TCP, avec une gestion asynchrone des messages entrants afin de ne pas bloquer l'interface graphique.

L'API interne n'est pas exposée sous forme de service externe, mais se matérialise par des fonctions et méthodes clairement définies dans la classe « GestionCryptoSQL ». Cette approche facilite la lisibilité du projet et permet une évolution future vers d'autres interfaces (client en ligne de commande, etc.).

Description de l’algorithme de chiffrage : forces et faiblesses

L’algorithme de chiffrement utilisé dans le projet repose sur un schéma hybride combinant cryptographie asymétrique et symétrique. Chaque routeur possède une paire de clés RSA générée dynamiquement lors de son initialisation. La clé publique est enregistrée en base de données afin d’être accessible aux clients et aux autres composants du système.

Lors de la construction de l’oignon côté client, le message est chiffré successivement pour chaque routeur de la route, en partant du dernier vers le premier. À chaque étape, une clé de session aléatoire est générée (seed). Cette clé est chiffrée avec la clé publique RSA du routeur concerné, puis utilisée pour chiffrer le contenu à l’aide d’un masque pseudo-aléatoire basé sur une opération XOR (en octets). Le résultat est encodé en Base64 afin de garantir une transmission fiable sous forme textuelle.

Du côté des routeurs, le déchiffrement s’effectue couche par couche. Chaque routeur utilise sa clé privée RSA pour récupérer la clé de session, puis déchiffre uniquement sa couche avant de transmettre le reste du message, toujours chiffré, au routeur suivant. Ce mécanisme garantit qu’aucun routeur intermédiaire ne peut accéder au message final ni connaître l’intégralité de la route.

Cette approche présente plusieurs forces. Elle respecte les principes du routage en oignon, limite la taille des messages malgré l’empilement des couches, et sépare clairement les rôles du chiffrement asymétrique et symétrique. L’algorithme est assez simple, tout en restant fonctionnel pour le projet.

Cependant, certaines faiblesses doivent être soulignées. Les tailles de clés RSA sont limitées afin de réduire la charge de calcul, ce qui affaiblit la sécurité face à des attaques plus modernes. Le chiffrement symétrique basé sur un XOR pseudo-aléatoire ne constitue pas une solution très robuste comparée à d’autres standards. Enfin, l’absence de mécanismes de vérification des routeurs ou d’authenticité expose le système à des attaques de modification de message, donc si une personne modifie le message depuis un routeur, le destinataire ne s’en rendra pas compte.

Malgré ces limites, l’algorithme de chiffrement remplit son objectif dans le cadre du projet : démontrer concrètement le fonctionnement du routage en oignon et les principes de confidentialité associés, tout en conservant une implémentation accessible et maîtrisée.

Gestion du projet

Le projet s'est déroulé sur la période du 17 novembre au 31 décembre 2025. La planification a été organisée de manière progressive afin de garantir le développement des composants essentiels dans un ordre cohérent. Les premières étapes ont consisté en l'organisation du projet, la création du dépôt Git et l'établissement d'un diagramme de Gantt pour suivre l'avancement. Les composants réseau ont été développés au début : la gestion des sockets TCP/UDP a permis d'assurer la communication entre clients et routeurs, tandis que l'implémentation des threads a rendu possible la gestion de plusieurs connexions.

Le module cryptographique a été mis en place afin de prendre en charge le chiffrement hybride des messages, cependant j'ai pris beaucoup de temps a comprendre et j'ai du modifier à plusieurs reprise le chiffrement pour qu'il soit enfin fonctionnel. Ensuite, la logique de routage a été développée, permettant aux routeurs de relayer les paquets en appliquant le principe du routage en oignon. La base MariaDB a été configurée pour stocker les clés, les informations des clients et des routeurs ainsi que les logs de transmission. Une interface graphique en Qt a été ajoutée pour les clients ainsi que pour le master, facilitant la configuration, l'envoi de messages et la visualisation des logs techniques.

L'approche adoptée a été itérative : chaque fonctionnalité a été testée indépendamment puis tester à nouveau dans le système complet. Certaines difficultés ont été rencontrées, notamment pour la gestion du multi-thread, la construction correcte des oignons de chiffrement et la synchronisation des informations réseau entre la configuration locale et la base de données. Chacunes de ces difficultés étaient dû aux connaissances relativement absentes des librairies utilisés. Ces difficultés ont été progressivement résolues par des tests sur des machines en locales en testant toutes les combinaisons entre les routeurs et les clients.

Le suivi du projet via le diagramme de Gantt a permis de respecter les échéances et d'anticiper les éventuels retards. La planification n'a pas été totalement respectée, car j'ai dû me former et comprendre le fonctionnement de certaines librairies comme Qt avant des les utiliser, se qui m'a décalé de mon planning initial. Ce travail m'a permis d'identifier les améliorations possibles, telles que l'ajout d'une authentification forte pour routeurs, l'intégrité des messages ou encore l'amélioration du chiffrement symétrique et de l'anonymisation du trafic.

Conclusion

Le projet a permis de mettre en place un système fonctionnel de routage en oignon, démontrant les principes de confidentialité et de relai sécurisé des messages. L'architecture développée est modulaire et maintenable, combinant les aspects réseau, cryptographie et interface utilisateur. Le respect d'une planification structurée a facilité l'avancement et le suivi du projet, avec des tests réguliers pour valider le fonctionnement des composants.

Malgré certaines limites, telles que l'utilisation d'un chiffrement asymétrique/symétrique simple et l'absence d'authentification forte, le projet atteint ses objectifs. Il constitue une base solide pour des évolutions futures, permettant d'envisager un renforcement de la sécurité, une meilleure anonymisation du trafic ou l'ajout d'une interface externe ou un client en ligne de commande.