Réponses au cahier des charges
Groupe : Les Pingouins

HARTMANN Quentin & KOUAME Akaza

Table des matières

1. Introduction

Ce document constitue la réponse formelle au cahier des charges de la SAE 3.02. Il détaille la mise en œuvre technique, les choix architecturaux, et la gestion du projet visant à créer un système de communication anonyme basé sur le principe du routage en oignon.

2. Réponses au cahier des charges

Le système développé répond point par point aux exigences du sujet :

Architecture Distribuée

Utilisation de 3 types de nœuds (Annuaire, Routeur, Client) sur des VMs distinctes.

Routage en Oignon

Encapsulation de messages avec le chiffrement RSA.

Anonymat

Chaque routeur ne connait pas l'arborescence complète.

Cryptographie

Implémentation du RSA avec la génération, le chiffrement et le déchiffrement.

Base de Données

Utilisation de MariaDB pour l'annuaire centralisé des routeurs.

Interface Graphique

Développement d'interfaces PyQt5 pour le suivi visuel des paquets.

Protocole Personnalisé

Communication par sockets avec un format de message textuel structuré.

3. Eléments implémentés et non implémentés

Éléments Implémentés

- Génération de clefs RSA avec la création de paires (publique/privée) de 1024 bits.
- Annuaire Dynamique, on a bien l'inscription en temps réel des routeurs dans une base MariaDB.
- Visualisation avec les interfaces graphique montrant les différentes étapes

Éléments Non Implémentés

- L'annuaire ne vérifie pas si un routeur est toujours actif après son inscription.
- Notre annuaire reste un point central de danger.

4. Structure du code et protocole

Modules et structure

Le projet est structuré en plusieurs modules Python :

- directory.py qui gère l'annuaire, la base de données et la distribution de la liste des routeurs.
- router.py qui implémente la logique de relais et de déchiffrement d'une couche de l'oignon.
- client.py qui fait l'interface utilisateur, la construction de l'oignon et l'envoi des messages.
- crypto.py qui est le cœur mathématique avec le chiffrement.
- db.py qui permet l'abstraction de la couche de données mariadb.

Protocole et API

La communication entre les nœuds utilise un protocole textuel simple :

- Inscription d'un routeur sous la forme : *INSCRIPTION|IP|PORT|E|N*
- Récupération de la liste : LISTE qui est sous forme : *IP1;PORT1;E1;N1|IP2;PORT2;E2;N2|...*
- Transfert de paquet : IP_SUIVANTE|PORT_SUIVANT|PAYLOAD_CHIFFRE

5. Algorithme de chiffrement

Principe générale

Pour chiffrer les données efficacement, nous utilisons un chiffrement hybride, c'est-à-dire une combinaison de deux méthodes :

RSA

- Sert uniquement à protéger une clé secrète temporaire, appelée clé de session.
- Cette clé est un nombre aléatoire utilisé une seule fois.
- Formule mathématique :

$$c = m^e \,(\mathrm{mod} n)$$

XOR

- Sert à chiffrer le message réel à l'aide de la clé de session.
- Chaque bit du message est combiné avec la clé via l'opération XOR.
- Formule :

$$message\ chiffré = message\ clair + \text{clé de session}$$

RSA va protéger la clé, et XOR va lui chiffrer le message. Cela nous permet d'être rapide et sécurisé.

Avantages et limites

Avantages

Rapidité
Le chiffrement XOR est très rapide surtout pour des messages qui vont être volumineux contrairement à RSA qui lui est lent sur de grandes quantités de données.

Séparation des rôles
RSA va être utilisé uniquement pour sécuriser la clé et non le message entier, ce qui va améliorer les performances.

Sécurité par couches
Même si un attaquant arriverait à casser le XOR, il lui faudrait encore récupérer la clé de session protégée par RSA.

Limites

XOR est simple
Utilisé seul, le XOR est moins sûr qu'un algorithme moderne comme AES, car il peut être vulnérable s'il fait face à des analyses statistiques.

RSA 1024 bits
Ce niveau de clé est suffisant pour notre projet mais il est aujourd'hui considéré comme insuffisant contre des attaquants expérimentés.

6. Gestion du projet

Méthodologie

Nous avons utilisé une approche assez simple :

- Phase 1 : Recherche et prototypage du chiffrage (RSA).
- Phase 2 : Développement de l'architecture réseau de base.
- Phase 3 : Intégration de la base de données et de l'interface graphique.
- Phase 4 : Tests en environnement multi-VM et documentation.

Outils et rôles

Versionnage : GitHub pour la mise en commun des travaux.

Coordination : Discord pour communiquer sur le projet.

Rôles :

- Quentin : Architecture système, Réseau, BDD.
- Akaza : Interface Graphique (PyQt5).
- Les deux : Algorithmes de chiffrage et développement du code.

7. Conclusion

Ce projet remplit les objectifs pédagogiques et techniques. Il prouve qu'un réseau anonyme et sécurisé peut être réalisé en utilisant des solutions simples et un chiffrement asymétrique fiable.