



CHIFFREMENT DE TEXTE

JAVA RMI

Presenté par
Fatima BOUYARMANE

Encadré par
M. Ahmmed AMAMOU



Résumé

Ce projet présente la conception et l'implémentation d'une application de gestion sécurisée du chiffrement et du déchiffrement de texte utilisant Java RMI (Invocation de Méthode à Distance) et l'algorithme AES (Advanced Encryption Standard). L'application vise à fournir une solution robuste pour chiffrer et déchiffrer des données textuelles sensibles, garantissant la confidentialité et la sécurité dans des environnements distribués.

Les fonctionnalités principales de l'application comprennent le chiffrement du texte en clair, le déchiffrement du texte chiffré et la gestion sécurisée des clés. Les utilisateurs peuvent générer, stocker et récupérer des clés de chiffrement, veillant à ce que seules les personnes autorisées puissent y accéder et les utiliser. L'interface utilisateur permet une sélection aisée de l'algorithme AES, la saisie du texte à chiffrer ou à déchiffrer, ainsi que la gestion des clés de chiffrement.

Java RMI est utilisé pour permettre l'invocation de méthode à distance, permettant aux clients d'effectuer des opérations de chiffrement et de déchiffrement sur un serveur distant de manière sécurisée. L'application garantit que toutes les communications et les transferts de données entre le client et le serveur sont protégés contre tout accès non autorisé.

En intégrant AES pour le chiffrement, l'application garantit un niveau élevé de sécurité, la rendant adaptée aux applications nécessitant une forte protection des données. Les détails de mise en œuvre, y compris l'utilisation des bibliothèques cryptographiques de Java et les meilleures pratiques pour la gestion sécurisée des clés, sont discutés en profondeur. Ce projet démontre l'utilisation efficace de Java RMI et d'AES pour créer un système de gestion sécurisé et efficace du chiffrement et du déchiffrement de texte.

Table des matières

I. Introduction.....	
• 1.1 Contexte du projet.....	
• 1.2 Objectifs.....	
II. Présentation des Technologies.....	
• 2.1 Java RMI (Remote Method Invocation).....	
• 2.1.1 Fonctionnement de Java RMI.....	
• 2.1.2 Avantages et inconvénients.....	
• 2.2 Algorithme AES (Advanced Encryption Standard).....	
• 2.2.1 Principes de base.....	
• 2.2.2 Sécurité et performances.....	
• 2.3 Gestion des clés de chiffrement.....	
III. Architecture de l'Application.....	
• 3.1 Diagramme de l'architecture.....	
• 3.2 Description des composants.....	
• 3.2.1 Interface utilisateur.....	
• 3.2.2 Serveur RMI.....	
• 3.2.3 Implémentation AES.....	
• 3.2.4 Gestion des clés.....	
IV. Conception et Implémentation.....	
• 4.1 Diagramme de sequence.....	
• 4.2 Diagramme d'activités.....	
• 4.3 Description détaillée des modules.....	
• 4.3.1 Module de chiffrement.....	
• 4.3.2 Module de déchiffrement.....	
• 4.3.3 Module de gestion des clés.....	
• 4.3.4 Source code.....	
V. Tests et Validation.....	
• 6.1 Plan de test.....	
• 6.2 Résultats des tests.....	
• 6.3 Analyse des performances.....	
VI. Conclusion.....	
• 7.1 Résumé des realizations.....	
• 7.2 Limitations et défis.....	
• 7.3 Perspectives d'amélioration.....	

Introduction

Dans un monde où la sécurité des données est devenue une préoccupation majeure, les méthodes de chiffrement et de déchiffrement jouent un rôle crucial dans la protection des informations sensibles. Le chiffrement permet de convertir des données lisibles en un format crypté qui ne peut être compris sans une clé appropriée, tandis que le déchiffrement permet de récupérer les données originales à partir de leur forme cryptée. Ce projet se concentre sur la conception et l'implémentation d'une application de gestion du chiffrement et du déchiffrement de texte, en utilisant Java RMI (Remote Method Invocation) et l'algorithme de chiffrement AES (Advanced Encryption Standard).

1.1 Contexte du projet

Avec l'augmentation des cyberattaques et des violations de données, il est impératif de sécuriser les informations sensibles contre les accès non autorisés. Les organisations et les individus recherchent des solutions robustes pour protéger leurs données confidentielles, que ce soit pendant leur stockage ou leur transmission. Le chiffrement est l'une des méthodes les plus efficaces pour assurer cette protection.

Java RMI est une technologie qui permet aux objets Java de communiquer entre eux à distance, facilitant ainsi la création d'applications distribuées. En combinant Java RMI avec l'algorithme AES, ce projet vise à développer une application capable de chiffrer et de déchiffrer des textes de manière sécurisée et efficace, tout en permettant une gestion centralisée des clés de chiffrement.

1.2 Objectifs

Les principaux objectifs de ce projet sont les suivants :

1. **Développer une application de chiffrement et de déchiffrement de texte** : Utiliser l'algorithme AES pour garantir la sécurité des données textuelles, en fournissant des fonctionnalités pour chiffrer et déchiffrer des textes.
2. **Intégrer Java RMI pour la communication à distance** : Mettre en œuvre Java RMI pour permettre l'invocation de méthodes à distance, facilitant ainsi les opérations de chiffrement et de déchiffrement sur un serveur centralisé.
3. **Assurer une gestion sécurisée des clés de chiffrement** : Créer un module pour la génération, le stockage et la récupération des clés de chiffrement, en s'assurant que ces clés sont protégées contre tout accès non autorisé.
4. **Fournir une interface utilisateur conviviale** : Concevoir une interface graphique intuitive permettant aux utilisateurs de sélectionner l'algorithme de chiffrement, de saisir du texte à chiffrer ou à déchiffrer, et de gérer les clés de chiffrement.
5. **Garantir la sécurité des communications** : Implémenter des mesures de sécurité pour protéger les communications entre le client et le serveur, incluant l'authentification et le contrôle d'accès.

Ce projet vise à démontrer comment les technologies Java RMI et AES peuvent être utilisées ensemble pour créer une application de chiffrement et de déchiffrement de texte sécurisée et efficace, répondant aux besoins de protection des données dans un environnement distribué.

II. Présentation des Technologies

2.1 Java RMI (Remote Method Invocation)

Java RMI (Remote Method Invocation) est une technologie de Java qui permet l'invocation de méthodes sur des objets situés à distance. Elle facilite la communication entre des applications distribuées, permettant ainsi à des objets Java de s'appeler et de s'exécuter à distance.

2.1.1 Fonctionnement de Java RMI

Java RMI fonctionne en permettant à un objet sur une machine virtuelle Java (JVM) d'invoquer des méthodes sur un objet situé sur une autre JVM. Voici les étapes clés du fonctionnement de Java RMI :

- **Définition de l'interface distante** : Une interface qui déclare les méthodes pouvant être appelées à distance.
- **Implémentation de l'interface distante** : Une classe qui implémente l'interface distante et fournit les définitions des méthodes.
- **Enregistrement de l'objet distant** : L'objet distant est enregistré avec le registre RMI, permettant aux clients de le localiser.
- **Appel des méthodes distantes** : Les clients utilisent un stub, une représentation locale de l'objet distant, pour appeler des méthodes comme s'il s'agissait d'un objet local.

2.1.2 Avantages et inconvénients

Avantages:

- **Simplicité** : Java RMI simplifie le développement d'applications distribuées en masquant les détails de la communication réseau.
- **Transparence** : Les appels de méthodes distantes sont similaires aux appels de méthodes locales, ce qui simplifie le code client.
- **Sécurité** : Intégration avec les mécanismes de sécurité de Java pour authentifier et autoriser les appels.

Inconvénients:

- **Performance** : Les appels distants peuvent être plus lents en raison de la latence du réseau et de la surcharge de la sérialisation.
- **Compatibilité** : Java RMI est limité à l'environnement Java, ce qui peut poser des problèmes d'interopérabilité avec d'autres langages.
- **Complexité de déploiement** : La configuration et la gestion des registres RMI et des objets distants peuvent être complexes.

2.2 Algorithme AES (Advanced Encryption Standard)

AES (Advanced Encryption Standard) est un algorithme de chiffrement symétrique largement utilisé pour sécuriser les données sensibles. Il est rapide, sécurisé et utilisé dans de nombreuses applications pour protéger les données.

2.2.1 Principes de base

AES fonctionne sur des blocs de 128 bits et utilise des clés de 128, 192 ou 256 bits. Les principales étapes de l'algorithme AES sont:

- **SubBytes** : Substitution des octets en utilisant une table de substitution (S-Box).
- **ShiftRows** : Déplacement des lignes de la matrice d'état.
- **MixColumns** : Mélange des colonnes de la matrice d'état pour diffuser les bits.
- **AddRoundKey** : Application de la clé de ronde à la matrice d'état par une opération XOR.

2.2.2 Sécurité et performances

Sécurité:

- **Robustesse** : AES est résistant à toutes les attaques connues à ce jour, y compris les attaques par force brute.
- **Clés de longueur variable** : Les différentes tailles de clés offrent des niveaux de sécurité supplémentaires.

Performances:

- **Efficacité** : AES est conçu pour être rapide et efficace, aussi bien en logiciel qu'en matériel.
- **Adaptabilité** : Utilisé dans de nombreuses applications, des réseaux sans fil aux transactions bancaires.

2.3 Gestion des clés de chiffrement

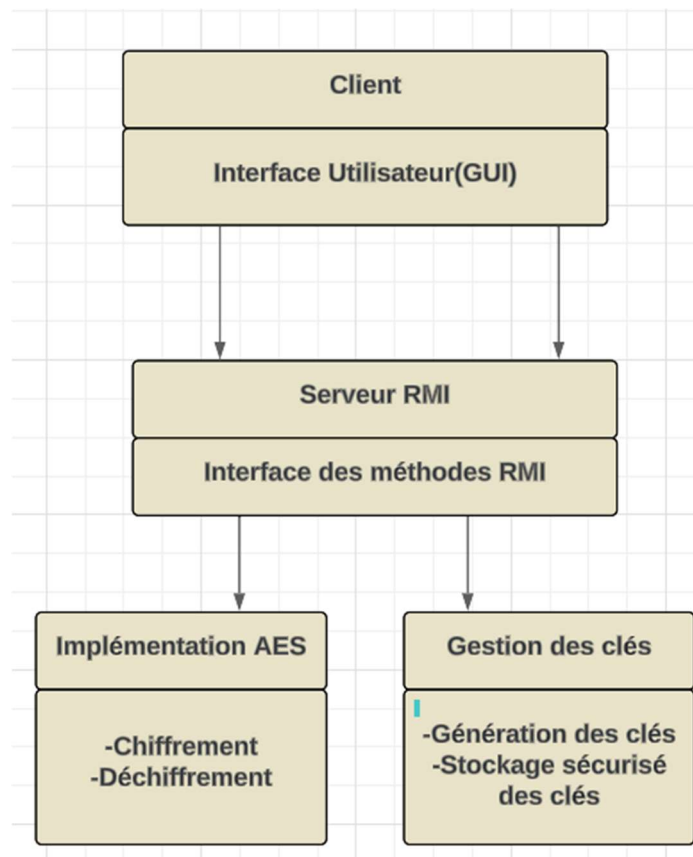
La **gestion des clés de chiffrement** est essentielle pour assurer la sécurité des données chiffrées. Les aspects clés de la gestion des clés comprennent:

- **Génération de clés** : Création de clés de chiffrement robustes et aléatoires.
- **Stockage de clés** : Protection des clés de chiffrement en les stockant de manière sécurisée.
- **Distribution de clés** : Transmission sécurisée des clés entre les parties autorisées.
- **Rotation des clés** : Mise à jour périodique des clés de chiffrement pour maintenir la sécurité.
- **Révocation de clés** : Retrait des clés compromises ou non utilisées pour empêcher leur usage non autorisé.

III. Architecture de l'Application

3.1 Diagramme de l'architecture

Le diagramme de l'architecture de l'application montre les composants principaux et leur interaction. Ce diagramme illustre comment les différentes parties de l'application travaillent ensemble pour fournir des fonctionnalités de chiffrement et de déchiffrement.



3.2 Description des composants

3.2.1 Interface utilisateur

L'interface utilisateur (UI) permet aux utilisateurs d'interagir avec l'application de manière conviviale. Elle peut être implémentée en utilisant une interface graphique (GUI) ou une interface en ligne de commande (CLI). Les principales fonctionnalités de l'interface utilisateur incluent :

- Sélection de l'algorithme de chiffrement (AES).
- Saisie du texte à chiffrer ou à déchiffrer.
- Affichage des résultats de chiffrement et de déchiffrement.
- Gestion des clés de chiffrement (génération, sélection, stockage).

3.2.2 Serveur RMI

Le serveur RMI gère les appels de méthodes à distance, permettant aux clients d'invoquer des méthodes de chiffrement et de déchiffrement sur le serveur. Les principales responsabilités du serveur RMI incluent:

- Enregistrement des objets distants avec le registre RMI.
- Exposition des méthodes de chiffrement et de déchiffrement via des interfaces distantes.
- Gestion des sessions client pour assurer la sécurité et l'authentification.

3.2.3 Implémentation AES

L'implémentation AES comprend les classes et les méthodes nécessaires pour effectuer le chiffrement et le déchiffrement des données en utilisant l'algorithme AES. Les principales tâches de ce composant sont:

- Chiffrement du texte en clair en utilisant une clé AES.
- Déchiffrement du texte chiffré en utilisant la même clé AES.
- Gestion des opérations de chiffrement en bloc, en assurant la confidentialité des données.

3.2.4 Gestion des clés

La gestion des clés est essentielle pour maintenir la sécurité de l'application. Ce composant s'occupe de la génération, du stockage, de la distribution et de la révocation des clés de chiffrement. Les principales fonctions de ce composant sont:

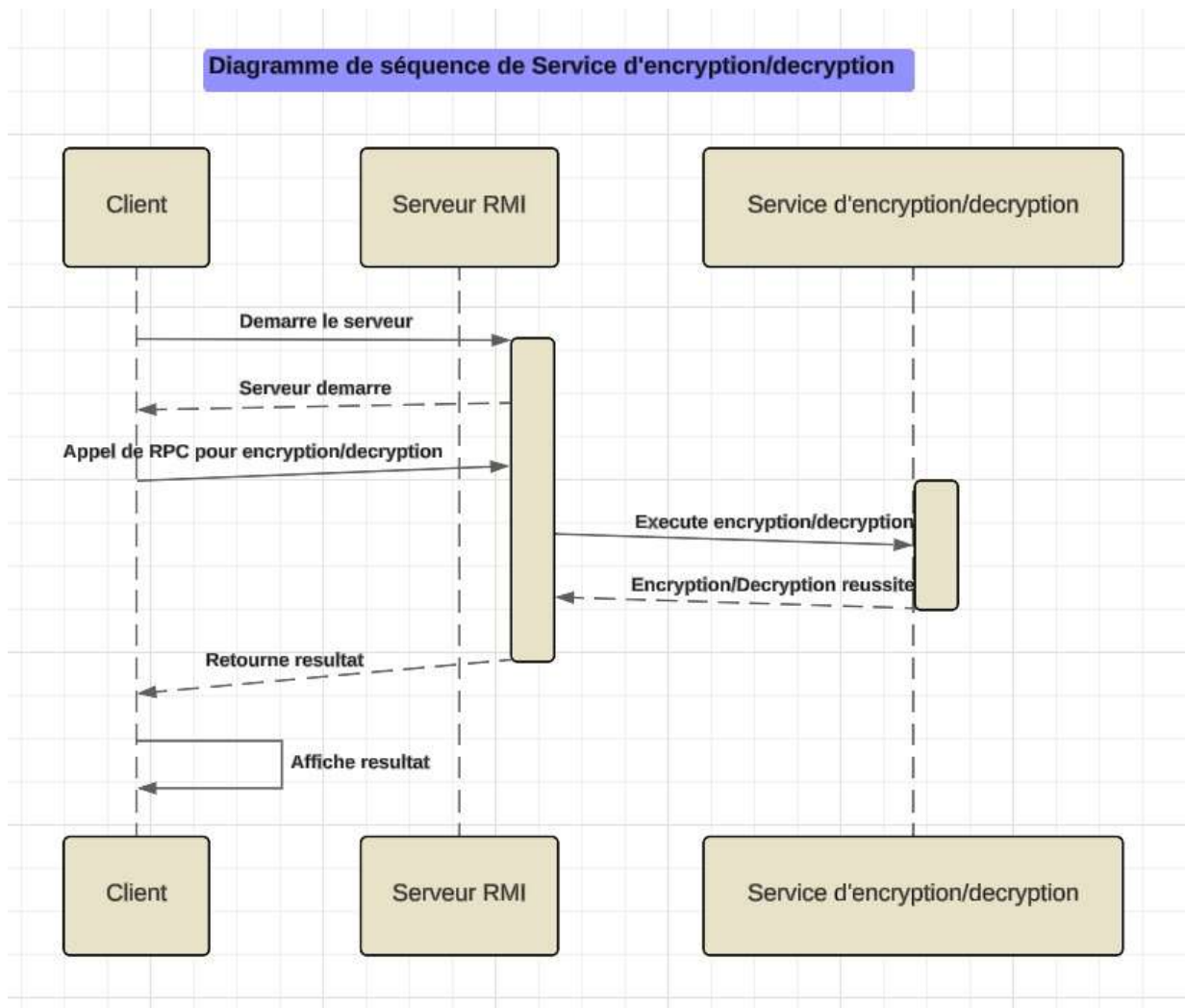
- Génération de clés AES sécurisées.
- Stockage des clés de manière sécurisée pour éviter les accès non autorisés.
- Distribution des clés aux utilisateurs autorisés via des canaux sécurisés.
- Rotation des clés pour maintenir la sécurité à long terme.
- Révocation des clés compromises ou non utilisées pour prévenir leur usage illicite.

IV. Conception et Implémentation

4.1 Diagramme de séquence

Le diagramme de séquence illustre la séquence des interactions entre les différents objets de l'application lors de l'exécution d'une fonctionnalité spécifique, telles que le chiffrement ou le déchiffrement du texte.

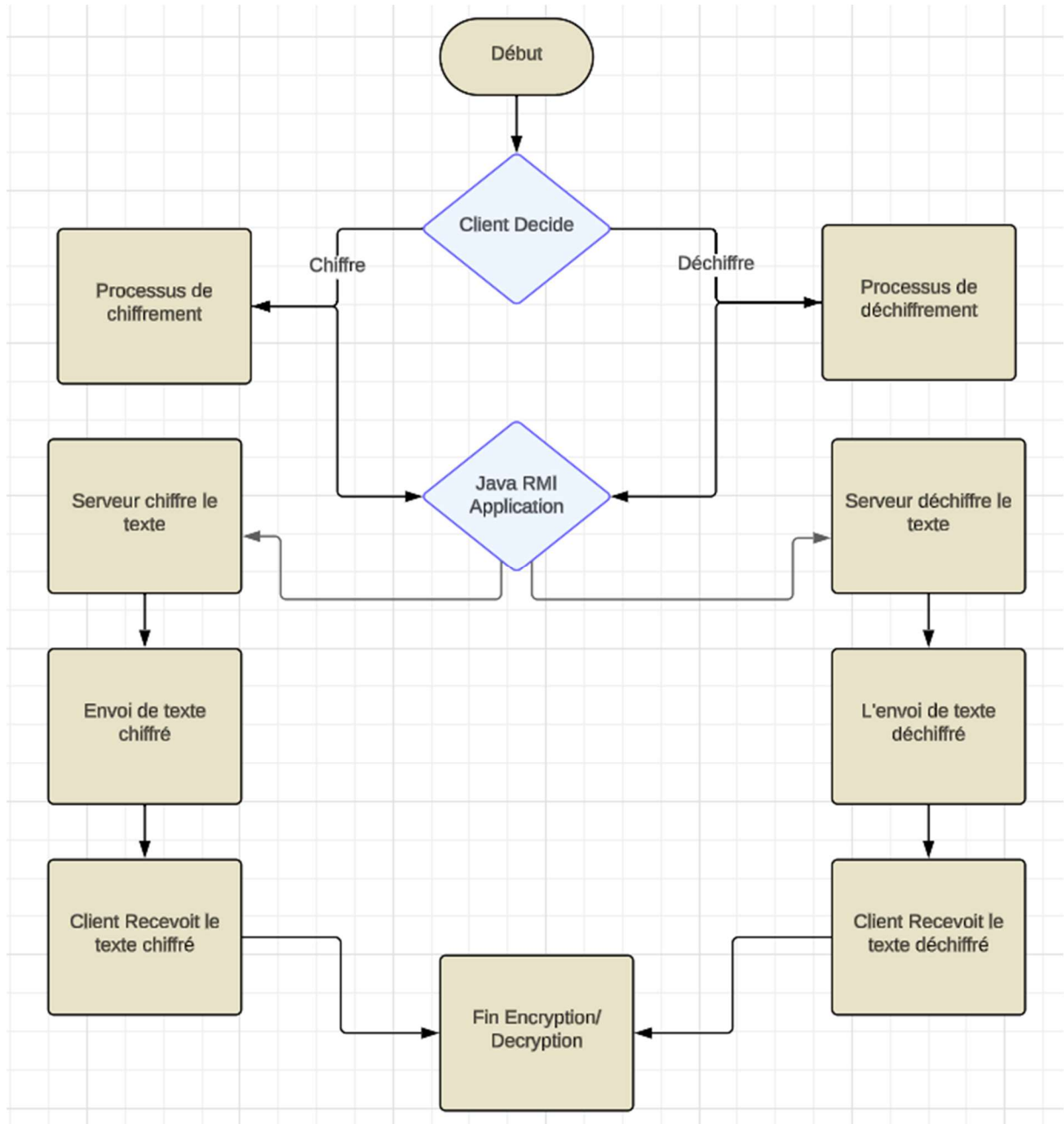
Diagramme de séquence:



4.2 Diagramme d'activités

Le diagramme d'activités décrit le flux d'activités dans l'application pour réaliser une tâche spécifique, comme le chiffrement ou le déchiffrement du texte.

Diagramme d'activités:



4.3 Description détaillée des modules

4.3.1 Module de chiffrement

Le module de chiffrement est responsable de l'application de l'algorithme AES pour chiffrer le texte en clair en utilisant une clé de chiffrement appropriée. Les principales fonctionnalités de ce module incluent la sélection de l'algorithme de chiffrement, la saisie du texte à chiffrer, et l'exécution du processus de chiffrement.

4.3.2 Module de déchiffrement

Le module de déchiffrement est chargé de déchiffrer le texte chiffré en utilisant la même clé de chiffrement utilisée pour le chiffrement. Il assure que seuls les utilisateurs autorisés ayant la clé appropriée peuvent déchiffrer le texte et accéder aux données originales en clair.

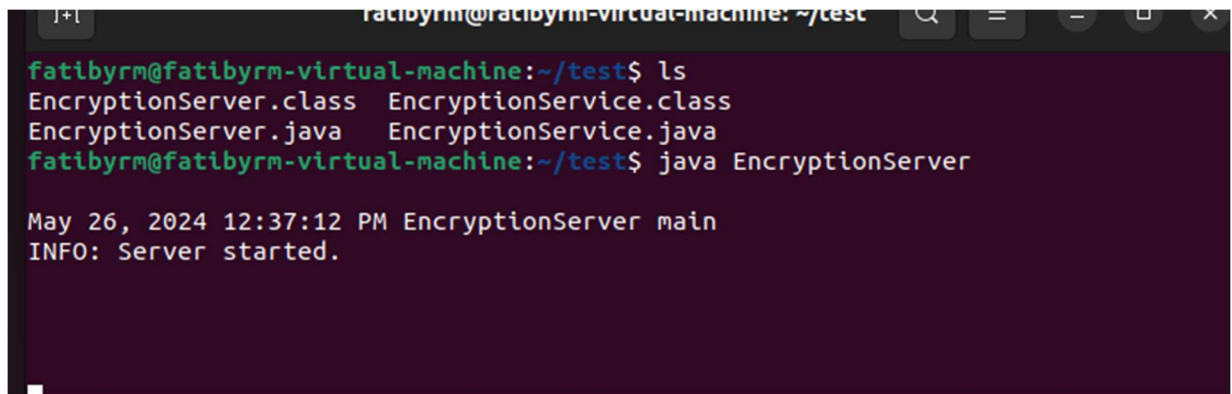
4.3.3 Source Code

Voici la source code: <https://github.com/FatimaBouyarmane/RPC-Encrypt-Decrypt>

VI. Test et Validation

5.1.1 Activation du Serveur RMI

Cette capture d'écran montre que le serveur RMI est actif et en attente de connexions. Cela inclut le lancement du serveur, l'enregistrement des objets distants, et la confirmation que le serveur est prêt à accepter les appels de méthode à distance.

A screenshot of a terminal window with a dark background. The prompt is 'fatibyrn@fatibyrn-virtual-machine:~/test'. The user enters 'ls' and the output shows 'EncryptionServer.class', 'EncryptionService.class', 'EncryptionServer.java', and 'EncryptionService.java'. Then the user enters 'java EncryptionServer' and the output shows 'May 26, 2024 12:37:12 PM EncryptionServer main' and 'INFO: Server started.'

La console affiche un message de confirmation que le serveur est prêt à accepter les connexions et les appels de méthode à distance.

5.1.2 L'interface GUI

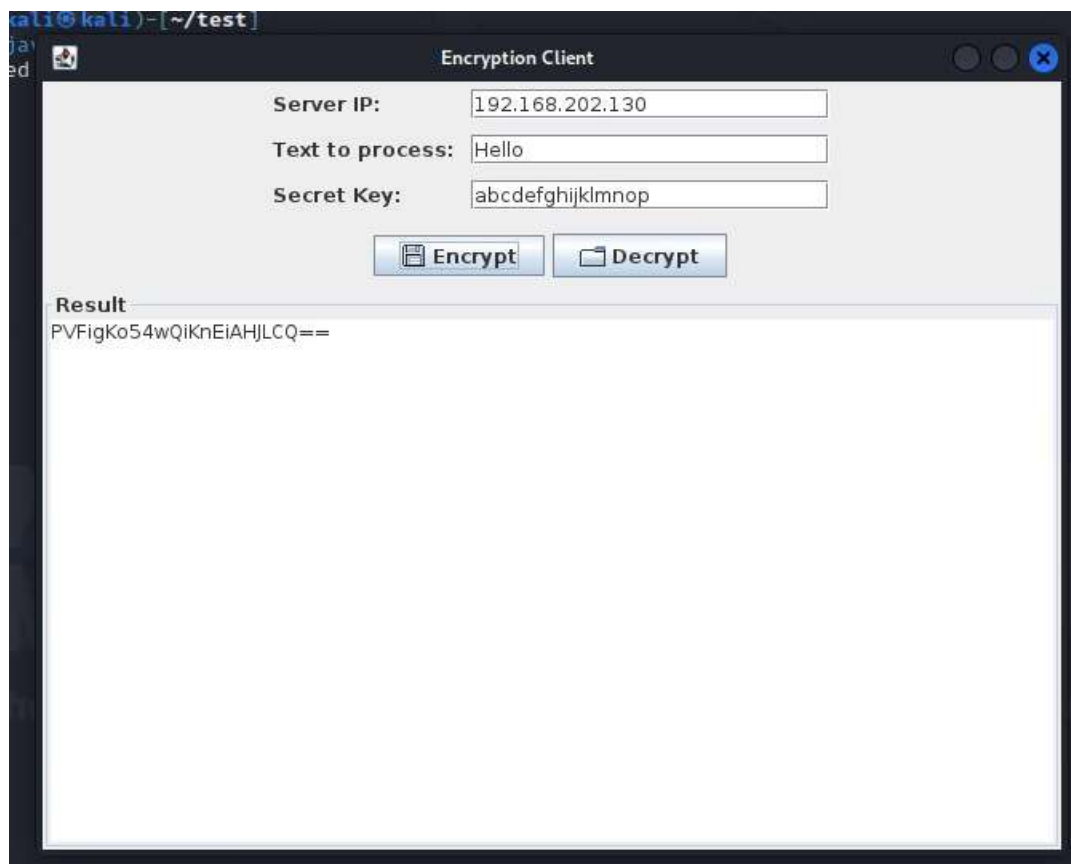
L'interface utilisateur (GUI) de l'application permet aux utilisateurs d'interagir facilement avec les fonctionnalités de chiffrement et de déchiffrement. Voici une description des composants de l'interface utilisateur :

Champs de saisie : Ces champs permettent à l'utilisateur de saisir le texte à chiffrer ou à déchiffrer. Ils acceptent le texte en clair pour le chiffrement et le texte chiffré pour le déchiffrement, Un champ pour le saisie de clé privé et un autre pour spécifier l'ip adresse de serveur.

Boutons pour le chiffrement et le déchiffrement : Deux boutons distincts permettent à l'utilisateur de lancer l'opération de chiffrement ou de déchiffrement. Lorsqu'un bouton est cliqué, l'application envoie la requête correspondante au serveur RMI.

Affichage des résultats : Une zone de texte ou un label affiche le texte chiffré ou déchiffré, permettant à l'utilisateur de voir immédiatement le résultat de l'opération.

Exemple de Chiffrement :



The screenshot shows a terminal window with the command `kali@kali:~/test` and an application window titled "Encryption Client". The application has three input fields: "Server IP:" with the value "192.168.202.130", "Text to process:" with the value "Hello", and "Secret Key:" with the value "abcdefghijklmnop". Below these fields are two buttons: "Encrypt" and "Decrypt". The "Result" section displays the output: "PVFigKo54wQiknEiAHJLCQ==".

Server IP: 192.168.202.130

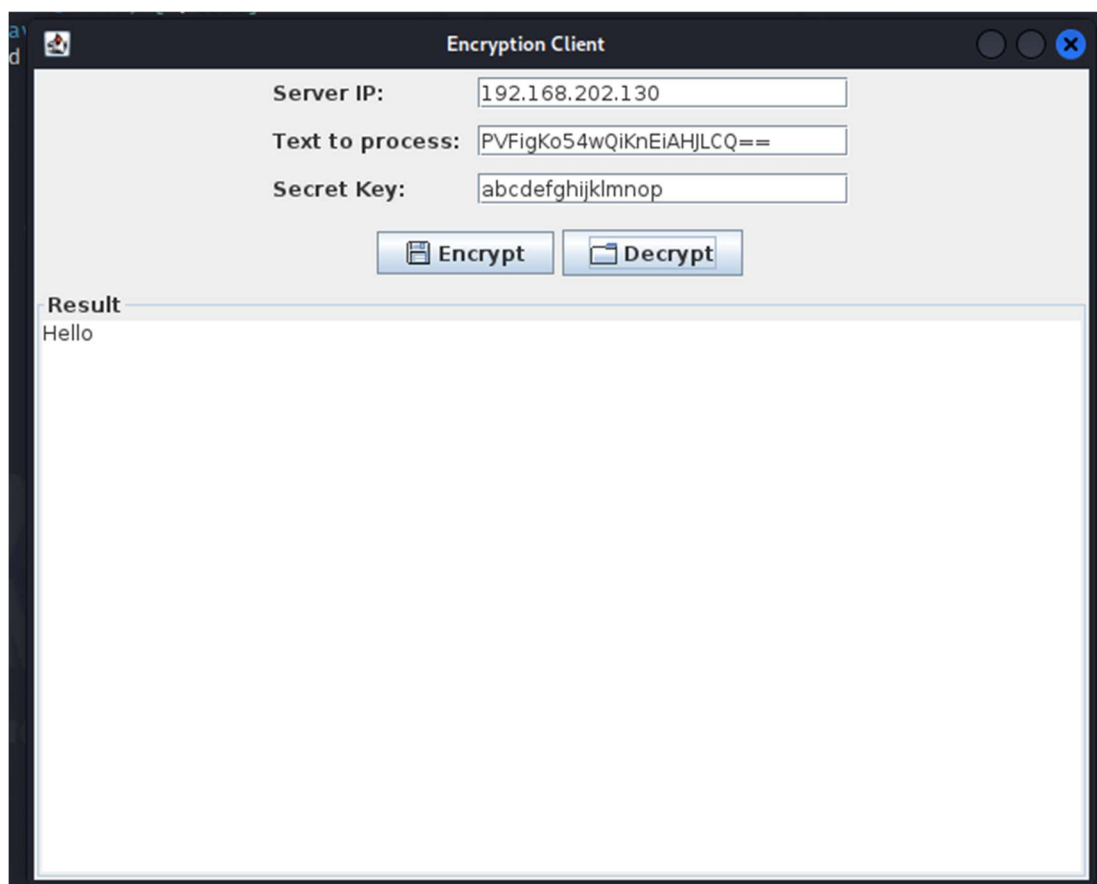
Text to process: Hello

Secret Key: abcdefghijklmnop

Encrypt Decrypt

Result
PVFigKo54wQiknEiAHJLCQ==

Exemple de Déchiffrement :



The screenshot shows the same "Encryption Client" application window. The "Server IP:" field remains "192.168.202.130". The "Text to process:" field now contains the encrypted text "PVFigKo54wQiknEiAHJLCQ==". The "Secret Key:" field remains "abcdefghijklmnop". The "Result" section displays the output: "Hello".

Server IP: 192.168.202.130

Text to process: PVFigKo54wQiknEiAHJLCQ==

Secret Key: abcdefghijklmnop

Encrypt Decrypt

Result
Hello

VII. Conclusion

6.1 Résumé des réalisations

Ce projet a permis de développer une application de gestion de chiffrement et déchiffrement de texte utilisant Java RMI et l'algorithme AES. Les principales réalisations incluent :

Implémentation de Java RMI : Développement d'un serveur RMI et d'un client capable de communiquer efficacement.

Intégration de l'algorithme AES : Utilisation de l'AES pour chiffrer et déchiffrer des textes de manière sécurisée.

Gestion des clés de chiffrement : Mise en place d'un système sécurisé pour générer, stocker et utiliser des clés de chiffrement.

Interface utilisateur conviviale : Développement d'une interface GUI permettant aux utilisateurs d'interagir facilement avec les fonctionnalités de l'application.

6.2 Limitations et défis

Bien que le projet ait atteint ses objectifs principaux, certaines limitations et défis ont été rencontrés :

Performances pour les grands textes : Les performances diminuent légèrement avec des textes de grande taille, ce qui pourrait être optimisé.

Complexité de la gestion des clés : La gestion sécurisée des clés de chiffrement est complexe et nécessite des améliorations continues pour garantir la sécurité.

Robustesse contre les attaques avancées : Bien que l'application soit sécurisée contre les attaques courantes, des tests supplémentaires sont nécessaires pour garantir une robustesse contre des attaques plus sophistiquées.

6.3 Perspectives d'amélioration

Pour améliorer l'application à l'avenir, plusieurs pistes peuvent être explorées :

Optimisation des performances : Optimiser les algorithmes de chiffrement et déchiffrement pour gérer plus efficacement de grands volumes de données.

Amélioration de la gestion des clés : Développer des mécanismes plus avancés pour la gestion des clés, y compris la rotation automatique des clés et le stockage sécurisé.

Renforcement de la sécurité : Effectuer des tests de sécurité avancés pour identifier et corriger les vulnérabilités potentielles.

Extensions des fonctionnalités : Ajouter des fonctionnalités supplémentaires, comme le chiffrement de fichiers, pour élargir les cas d'utilisation de l'application.

Conclusion Générale

En conclusion, le développement de l'application de gestion de chiffrement et déchiffrement de texte avec Java RMI et l'algorithme AES a représenté une exploration profonde des domaines cruciaux de la sécurité et de la communication distribuée. L'intégration réussie de Java RMI a permis une communication transparente entre le client et le serveur, facilitant ainsi les opérations de chiffrement et déchiffrement avec une fiabilité accrue. De plus, l'utilisation sécurisée de l'algorithme AES a garanti un niveau élevé de confidentialité des données, élément essentiel dans les environnements informatiques modernes axés sur la protection des informations sensibles.

Au cours du développement, plusieurs défis ont été identifiés, notamment en ce qui concerne les performances pour le traitement de grands volumes de données et la complexité de la gestion des clés de chiffrement. Ces défis soulignent l'importance de l'optimisation continue et de l'amélioration des processus pour garantir une application robuste et sécurisée.

Malgré ces limitations, les contributions de l'application sont significatives. La conception d'une interface utilisateur conviviale offre une expérience transparente aux utilisateurs, renforçant ainsi leur interaction avec les fonctionnalités de l'application. De plus, les perspectives d'amélioration identifiées, telles que l'optimisation des performances et le renforcement de la sécurité, fournissent un cadre solide pour l'évolution future de l'application, visant à répondre aux besoins croissants en matière de sécurité et de confidentialité des données dans un environnement numérique en constante évolution.

En résumé, l'application de gestion de chiffrement et déchiffrement de texte représente une étape cruciale dans la sécurisation des communications et des données sensibles, tout en soulignant l'importance de l'innovation continue pour relever les défis complexes de la sécurité informatique.