IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.53

Volume 5, Issue 1, January 2025

Secure SMS Sending using Encryption and Decryption Method

Akansha More[1], Mohini Patil[2], Jayshree Rikame[3], Komal Gorde[4], Prof. Narendra Joshi[5], Prof. Yogesh Bhalerao[6]

Students, Department of Cloud Technology, and Information Technology[1,2,3,4]

Guide, Department of Cloud Technology and Information Security[5,6]

Sandip University, Nashik, India

Abstract*: In real world, Message communication has grown in popularity and frequency as people work remotely and enterprises move the cloud. Nowadays, message communication via the network is very risky because there are many threats. However, at any time when employees use technologies to share messages between devices, there are security risk involved. Message communication can also introduce risks of hacking and loss of sensitive information. Hence, secure message communication projects are required as business today face multiple security threats in highly competitive environment.*

Keywords: Microservice Architecture, Service Scalability, Automated Deployment

I. INTRODUCTION

Secure SMS sending using encryption and decryption involves converting a plain text message into a scrambled, unreadable code (ciphertext) before transmitting it via SMS, which can only be decoded back to the original message by the intended recipient using a secret key, effectively protecting the sensitive information from being intercepted and read by unauthorized parties during transit; this process utilizes cryptographic algorithms to achieve confidentiality in SMS communication.

The major objectives of our project are:

- Objective 1: To develop a secure SMS application using encryption techniques.
- Objective 1: To develop a secure SMS application using encryption techniques.
- Objective 2: To use encryption for SMS messages thus making unauthorized access impossible and making sure that just the right receiver can read the message for the purpose of confidentiality.
- Objective 3: To deploy error-checking systems that enable the receiver to verify the correct message was not changed in any way during transmission so that data integrity is maintained.
- Objective 4: To deploy means by which the sender and receiver are to be identified in order to deter fraudulent communication or unauthorized access.
- Objective 5: To employ secure encryption techniques on the data from the point of sending to the point of delivery, making it unreadable to all except the receiver.

1. Project Objective

The goal is to develop a mobile application that allows users to send SMS messages securely by encrypting the content before transmission and decrypting the content on the recipient's device. This ensures privacy and protects against unauthorized access to the message data during transmission.

2. Features and Functionality

- **User Authentication:** Users must authenticate themselves via a login system (using credentials or biometric authentication) to access the app.
- **Encryption Mechanism:** The app will use encryption algorithms (e.g., AES, RSA) to securely encode the SMS content before sending it. The encryption should be strong enough to protect the data from interception or unauthorized access during transmission.

- **Decryption Mechanism:** The recipient will receive an encrypted message and must have the appropriate decryption key or credentials to read the message. Decryption should occur in real-time, allowing the recipient to view the original content securely.
- **Message Sending/Receiving:** Users will be able to send encrypted SMS messages to contacts stored within the app. The app should be able to receive encrypted messages and decrypt them on the recipient's device.
- **Key Management:** Secure key exchange protocols should be used to ensure both sender and recipient have the necessary encryption/decryption keys. This could involve public-private key pair systems or a

II. SCOPE

When designing a SMS Sending App using encryption and decryption methods, the scope of the project can be broken down into several key components and objectives. Below is an outline of the scope:

1. Project Objective

The goal is to develop a mobile application that allows users to send SMS messages securely by encrypting the content before transmission and decrypting the content on the recipient's device. This ensures privacy and protects against unauthorized access to the message data during transmission.

2. Features and Functionality

- **User Authentication:** Users must authenticate themselves via a login system (using credentials or biometric authentication) to access the app.
- **Encryption Mechanism:** The app will use encryption algorithms (e.g., AES, RSA) to securely encode the SMS content before sending it. The encryption should be strong enough to protect the data from interception or unauthorized access during transmission.
- **Decryption Mechanism:** The recipient will receive an encrypted message and must have the appropriate decryption key or credentials to read the message. Decryption should occur in real-time, allowing the recipient to view the original content securely.

III. PLANNING AND PROTOTYPING

SMS Sending App Using Encryption & Decryption: Planning & Prototype

To build a secure SMS sending app with encryption and decryption, it is essential to lay out a clear plan for the app's architecture, the encryption methods, key management, and the user flow. Below, I will walk you through the planning steps, followed by a high-level prototype for the app.

1. Planning Phase

1.1 Core Features and Functionality

- User Authentication: Ensure users can securely log into the app (e.g., using passwords or biometrics).
- Message Encryption: Encrypt the message content before sending it via SMS.
- Message Decryption: Decrypt the received message on the recipient's device.
- Secure Key Management: Use encryption keys (RSA and AES) for securing the message and key exchange.

End-to-End Encryption (E2E): Ensure the server never accesses the message content.
SMS Sending and Receiving: Integrate with an SMS gateway (e.g., Twilio) to send and receive messages.

1.2 Functional Requirements

Message Composition:
User can compose a message and send it to a recipient.

Encryption:
The message is encrypted using AES (for fast encryption) and the AES key is further encrypted using the recipient's RSA public key.

Decryption:
The recipient uses their RSA private key to decrypt the AES session key, which is then used to decrypt the message content.

Secure Key Storage:
The RSA private keys are securely stored on each user's device (using Keystore for Android or Secure Enclave for iOS).

1.3 Non-Functional Requirements

- Security: End-to-end encryption should be implemented so that no one (including the server or SMS gateway) can read the message content.
- Performance: The app must provide real-time encryption and decryption without causing delays.
- Usability: Simple and intuitive user interface for composing and reading encrypted messages.
- Scalability: The backend should handle multiple user accounts and manage encryption keys securely.

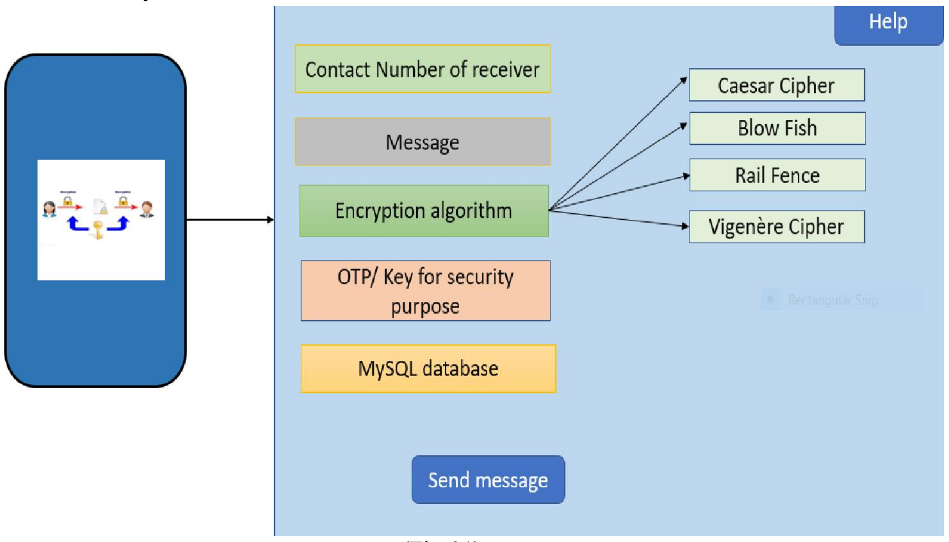
2. Technical Planning

2.1 Encryption Algorithm

AES (Advanced Encryption Standard): Used to encrypt the message content because it's fast and secure.
RSA (Rivest-Shamir-Adleman): Used to encrypt the AES key for secure key exchange. The recipient uses their RSA private key to decrypt the AES key.

2.2 Key Management

Public/Private Key Pair (RSA):
Each user generates an RSA key pair (public and private).
Public keys are shared with others (either stored on the server or exchanged directly).
Private keys are stored securely on the user's device.



(Fig 01)

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 5, Issue 1, January 2025

AES Session Key:
For each message, a unique AES key is generated to encrypt the message.
The AES key is encrypted using the recipient's public RSA key before being sent along with the message.

2.3 Backend

Role: The backend handles user authentication, storing public keys, and routing the encrypted messages to the SMS gateway. The backend should never decrypt the message content.
Database: Can use SQL or NoSQL (e.g., Firebase) to store user data, including public keys.

2.4 SMS Gateway

Integration: Use an SMS gateway like Twilio or Nexmo to send encrypted SMS messages. The gateway handles the transmission of messages but does not decrypt the message content.

3. Prototype Design

3.1 User Interface (UI) Design

Login Screen:
Allow users to log in securely (e.g., via username/password or biometric authentication).
Optionally, support two-factor authentication (2FA) for added security.

Message Compose Screen:
Input field for the recipient's phone number.
Input field for composing the encrypted message.
Send Button to encrypt and send the message.

Message Inbox Screen:
Display a list of received messages.
Upon selection, the app will decrypt the message and show it to the user.

Settings Screen:
Allow users to manage their RSA key pair.
Display an option to regenerate public/private keys if necessary.

3.2 User Flow

Login:
User enters credentials (username/password or biometric).
App verifies credentials and authenticates user.

Sending a Message:
User enters recipient's phone number and message in the compose screen.

The app:
Encrypts the message using AES.
Encrypts the AES session key using the recipient's public RSA key. The app sends the encrypted message and the encrypted AES key to the server, which forwards it to the SMS gateway. SMS gateway sends the encrypted message to the recipient.
Receiving a Message:
The recipient receives the encrypted SMS.
The recipient's app retrieves the encrypted message and AES key. The app decrypts the AES key using the recipient's private RSA key. The app uses the decrypted AES key to decrypt the message. The app displays the decrypted message.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 5, Issue 1, January 2025

4. Prototype Implementation
4.1 Tools and Technologies
Mobile Development:
Android:
Java, with Android Keystore for secure key storage.
Cross-Platform: Flutter with platform channels for native encryption.

Database:
Firebase for real-time data storage or PostgreSQL for traditional SQL storage.

IV. CHALLENGES AND CONSIDERATIONS

1. Encryption Algorithm Selection
Challenge: Choosing an appropriate encryption algorithm is crucial for security and performance. Common algorithms include AES (Advanced Encryption Standard) and RSA (Rivest–Shamir–Adleman).

Considerations:
AES: Symmetric encryption, fast and efficient, suitable for encrypting large amounts of data like SMS
RSA: Asymmetric encryption, more secure for key exchange, but slower and better suited for smaller pieces of data (e.g., encrypting the symmetric key itself).
Key Management: Using AES would require securely storing and exchanging the encryption keys, which can be a challenge in a mobile environment.

2. Key Management and Secure Storage
Challenge: Ensuring that encryption keys (both public and private keys, or the symmetric key) are securely stored and not exposed to unauthorized users.
Considerations:
On mobile platforms, utilize Secure Storage mechanisms such as Keychain on iOS or Keystore on Android to store keys securely.
Key Exchange: For secure communication between users, you need a mechanism for exchanging encryption keys securely, like Diffie-Hellman or using public key infrastructure (PKI).
Key Expiration and Rotation: Regularly rotating keys helps mitigate the risks of key compromise.

3. End-to-End Encryption
Challenge: Implementing true end-to-end encryption ensures that only the sender and receiver can read the message.
Considerations: Messages should be encrypted on the sender's device and decrypted on the recipient's device, with no plaintext exposure during transmission.
Use of a secure channel for exchanging public keys (e.g., via a trusted server) to ensure authenticity.

4. Performance Considerations
Challenge: Encryption and decryption operations can introduce latency and performance overhead.

Considerations:
Efficiency: Use efficient algorithms (e.g., AES for symmetric encryption) to minimize the impact on performance, especially on devices with limited processing power.
Payload Size: SMS is limited in size, so consider how encryption will impact message length. Encrypted data will often be larger than plaintext, requiring careful handling**.**

V. CONCLUSION

In conclusion, our project successfully demonstrates the importance and feasibility of secure SMS communication through encryption and decryption techniques.By implementing robust cryptographic algorithms, we have ensured that sensitive information sent via SMS remains confidential, protected from unauthorized access, and secure against potential cyber threats.

This solution not only enhances privacy for individuals but also addresses critical security needs across various sectors, including healthcare, finance, and personal safety.

With continuous advancements in encryption technology, such systems can further evolve to meet the growing demand for secure communication in today's digital age.

VI. ACKNOWLEDEMENT

We would like to express our sincere gratitude to Prof. Yogesh Bhalerao for their invaluable support, guidance, and encouragement throughout this project. Their expertise and insightful feedback were instrumental in the successful completion of this work. We also extend our thanks to the DevOps and software development communities for their comprehensive resources, tutorials, and best practices that shaped the technical direction of this research. Special appreciation goes to Sandip University for their collaboration and critical review of the CI/CD processes outlined in this study. Finally, we acknowledge the efforts of open-source contributors and the maintainers of Docker and Kubernetes for their dedication to creating and improving the tools that empower modern, scalable microservices deployment. Their contributions have been fundamental to the realization of our work in developing a robust event management system

REFERENCES

[1] "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation" by Jez Humble and David Farley

[2] "https://ieeexplore.ieee.org/document/8901380"

[3] "https://eudl.eu/doi/10.4108/eetiot.v9i2.2968"

[4] "https://ieeexplore.ieee.org/document/9426511"

[5]"https://www.researchgate.net/publication/298298027_Secure_SMS_Encryption_Using_RSA_Encryption_Algorithm_on_Android_Message_Application"

[6]https://stackoverflow.com/questions/27577680/how-to-encrypt-decrypt-sent-received-messages-in-android

[7]"https://ieeexplore.ieee.org/document/9395777/"