World Journal of Advanced Research and Reviews

eISSN: 2581-9615 CODEN (USA): WJARAI
Cross Ref DOI: 10.30574/wjarr
Journal homepage: https://wjarr.com/

(Review Article)

Secure Messaging Application Using Java Cryptographic Architecture (JCA)

Oladipupo Dopamu *, Chika Okonkwo, Samuel Adeniji and Femi Oke

Department of Computer Sciences, Western Illinois University, Macomb Illinois USA.

World Journal of Advanced Research and Reviews, 2024, 22(02), 2056–2063

Publication history: Received on 20 April 2024 revised on 28 May 2024; accepted on 30 May 2024

Article DOI: https://doi.org/10.30574/wjarr.2024.22.2.1670

Abstract

In today's digital era, ensuring the security and privacy of online communications has become paramount. With the increasing prevalence of cyber threats, the need for robust solutions to protect sensitive information is more critical than ever. This paper presents the development of a Secure Messaging Application that leverages the Java Cryptographic Architecture (JCA) to address these security concerns. The application integrates a suite of cryptographic techniques, including symmetric key encryption, asymmetric key encryption, cryptographic hashes, and digital signatures, to guarantee the confidentiality, integrity, authenticity, and non-repudiation of messages exchanged between users.

The Secure Messaging App employs symmetric key encryption to securely encode messages, while asymmetric key encryption is used for safe key exchange and to ensure that only the intended recipient can decrypt the messages. Cryptographic hashes provide a means to verify the integrity of the messages, ensuring that they have not been altered during transmission. Digital signatures are used to authenticate the sender, thereby preventing any denial of having sent the message and ensuring its authenticity.

Keywords: Secure Messaging; Cryptography; Java Cryptographic Architecture; Symmetric Encryption; Asymmetric Encryption; Digital Signatures; Hashing

1. Introduction

The advent of digital communication has revolutionized the way information is exchanged, bringing unprecedented convenience and efficiency to personal and professional interactions. However, this transformation has also introduced significant security challenges, as sensitive information transmitted over digital networks is susceptible to interception, tampering, and unauthorized access. Ensuring the confidentiality, integrity, and authenticity of messages has thus become a critical concern in the digital age. Cryptographic algorithms offer robust solutions to these security challenges by providing mechanisms for secure communication.

This paper details the development of a Secure Messaging Application that leverages the Java Cryptographic Architecture (JCA) to address these pressing security concerns. The application is meticulously designed to offer secure communication by employing various cryptographic techniques, including public key generation, symmetric key encryption, cryptographic hashes, digital signatures, and base64 encoding.

The Secure Messaging App is a Java-based application specifically engineered to facilitate secure communication among users through the integration of different cryptographic methodologies. Symmetric key cryptography is utilized for efficient and secure message encryption, while asymmetric key cryptography ensures secure key exchange and non-repudiation. Hashing techniques provide data integrity by generating unique message digests, and digital signatures authenticate the sender's identity, guaranteeing the message's authenticity.

* Corresponding author: Oladipupo Dopamu

To ensure that the application's security features are accessible only to legitimate users, the app requires users to register with a unique username. Once registered, users are equipped with cryptographic key pairs that enable them to securely exchange messages with other registered users. This user-centric approach ensures that all communications within the application are protected by end-to-end encryption, providing a high level of security for the exchanged messages.

1.1. Application setup

The Secure Messaging Application can be executed using any Integrated Development Environment (IDE) such as Visual Studio Code or Jgrasp. The following steps outline the process to set up and run the application:

- Open the Java File: Locate and open the SecureMessagingApp.java file in your IDE.
- Compile the Java File: Compile the file to check for any syntax errors and ensure the code is executable.
- Run the Application: Execute the compiled file to launch the application interface.
- User Choices: The application presents four options: Register as a new user, Send a message, View a message, Quit.

1.2. Functionality - how it works

1.2.1. Register as a New User

Users can register with a unique username. Upon registration, the application generates a key pair (public and private keys) for the user and stores the user's public key in the database. This process ensures that each user has a distinct cryptographic identity within the system.

1.2.2. Send a Message

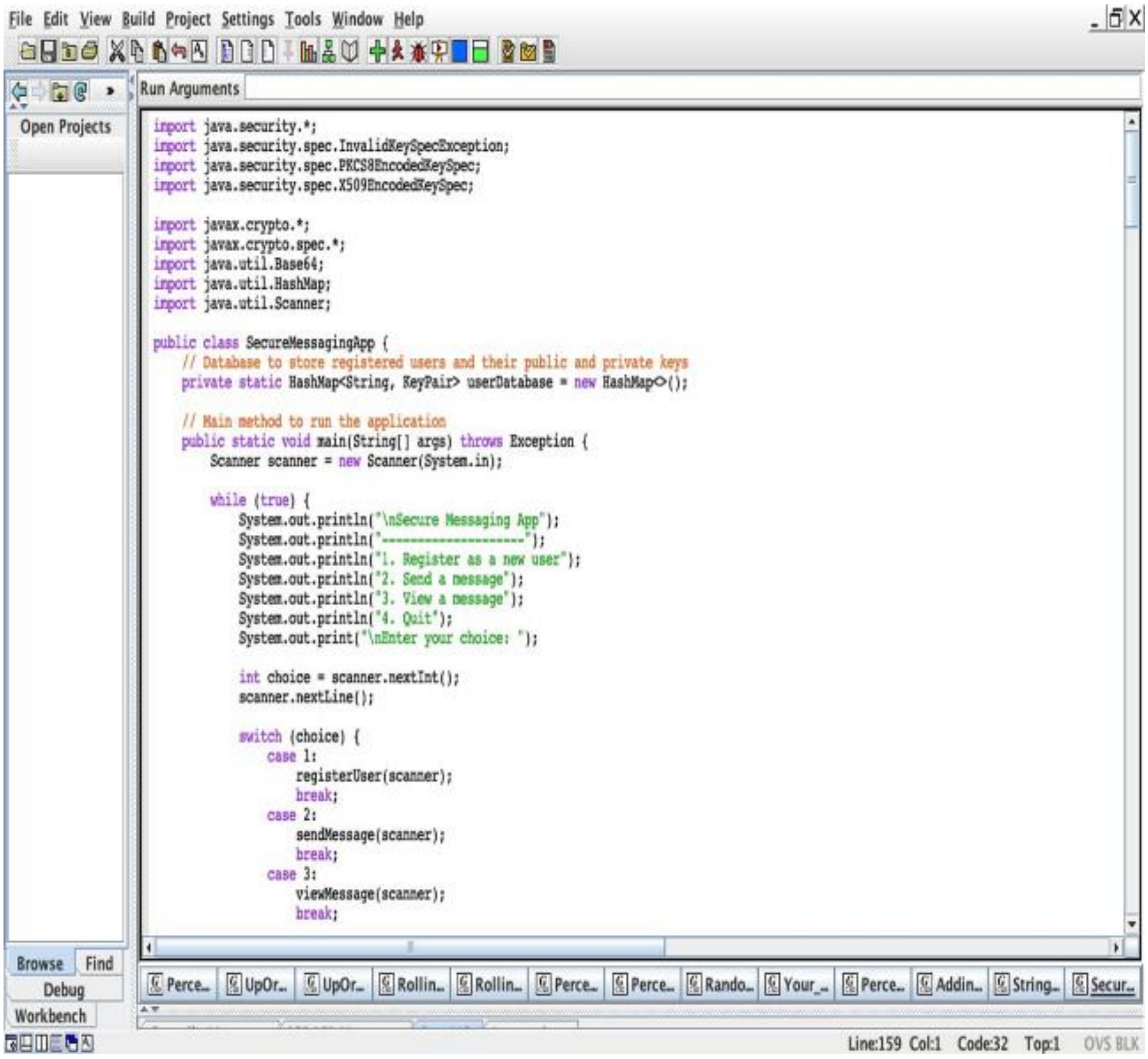
To send a message, the user selects a recipient from the list of registered users. The message is encrypted using a newly generated symmetric key with the Advanced Encryption Standard (AES) algorithm. This symmetric key is then encrypted with the recipient's public key using the RSA algorithm. Additionally, the application computes a SHA-256 hash of the message and signs this hash with the sender’s private key, ensuring both integrity and non-repudiation. The final message package includes the encrypted message, the encrypted symmetric key, the message hash, and the digital signature.

1.2.3. View a Message

When a user receives a message, the application decrypts the symmetric key using the recipient's private key and then decrypts the message using the symmetric key. The integrity and authenticity of the message are verified using the message hash and the sender’s public key to confirm the digital signature.

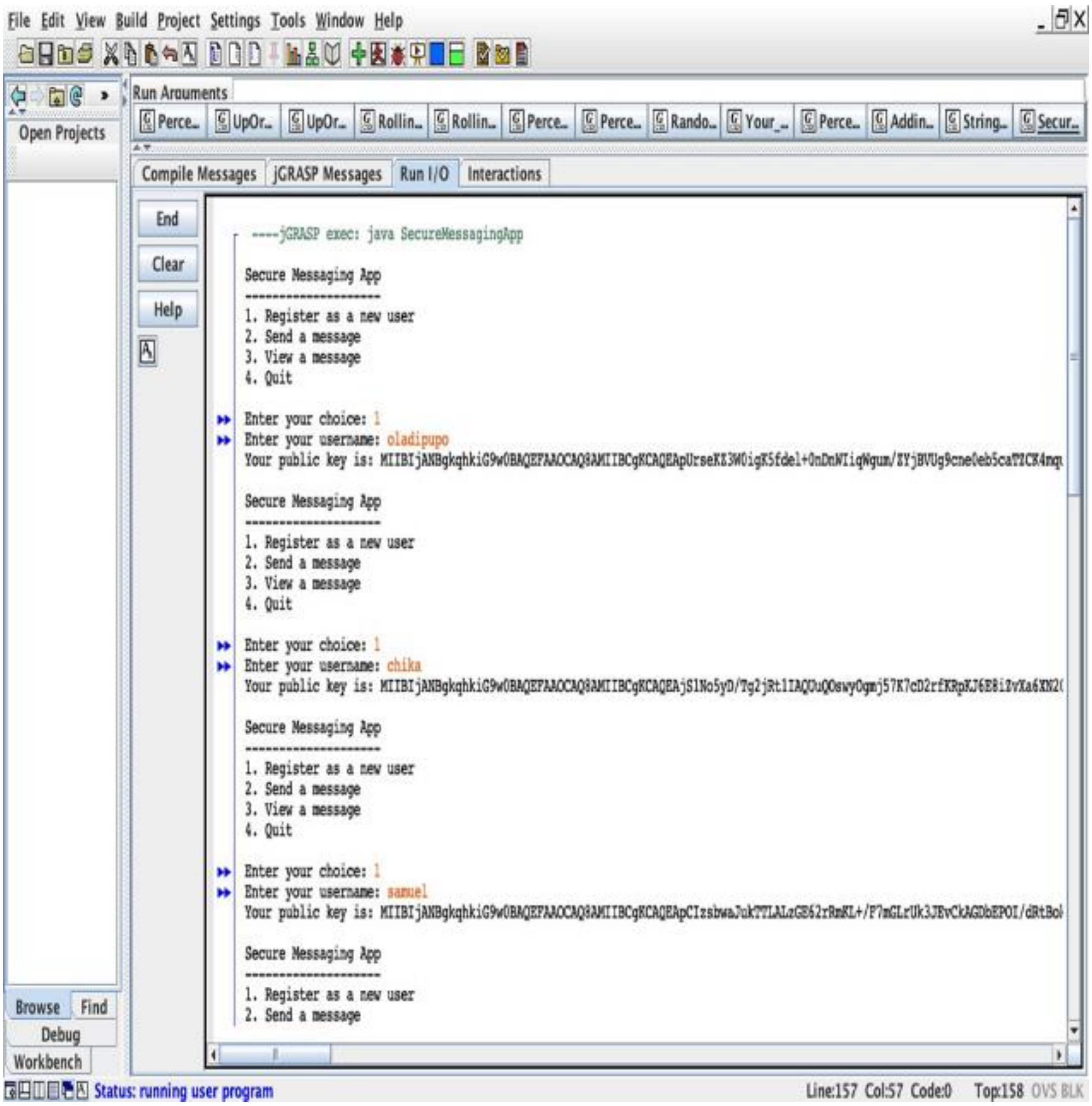
1.2.4. Quit

This option allows users to securely exit the application, ensuring that all cryptographic keys and session information are appropriately handled to prevent any potential security risks.


File Edit View Build Project Settings Tools Window Help
Open Projects
Run Arguments
Browse Find
Debug
Workbench
Perce.. UpOr.. UpOr.. Rollin.. Rollin.. Perce.. Perce.. Rando.. Your_.. Perce.. Addin.. String.. Secur..
Line:159 Col:1 Code:32 Top:1 OVS BLK

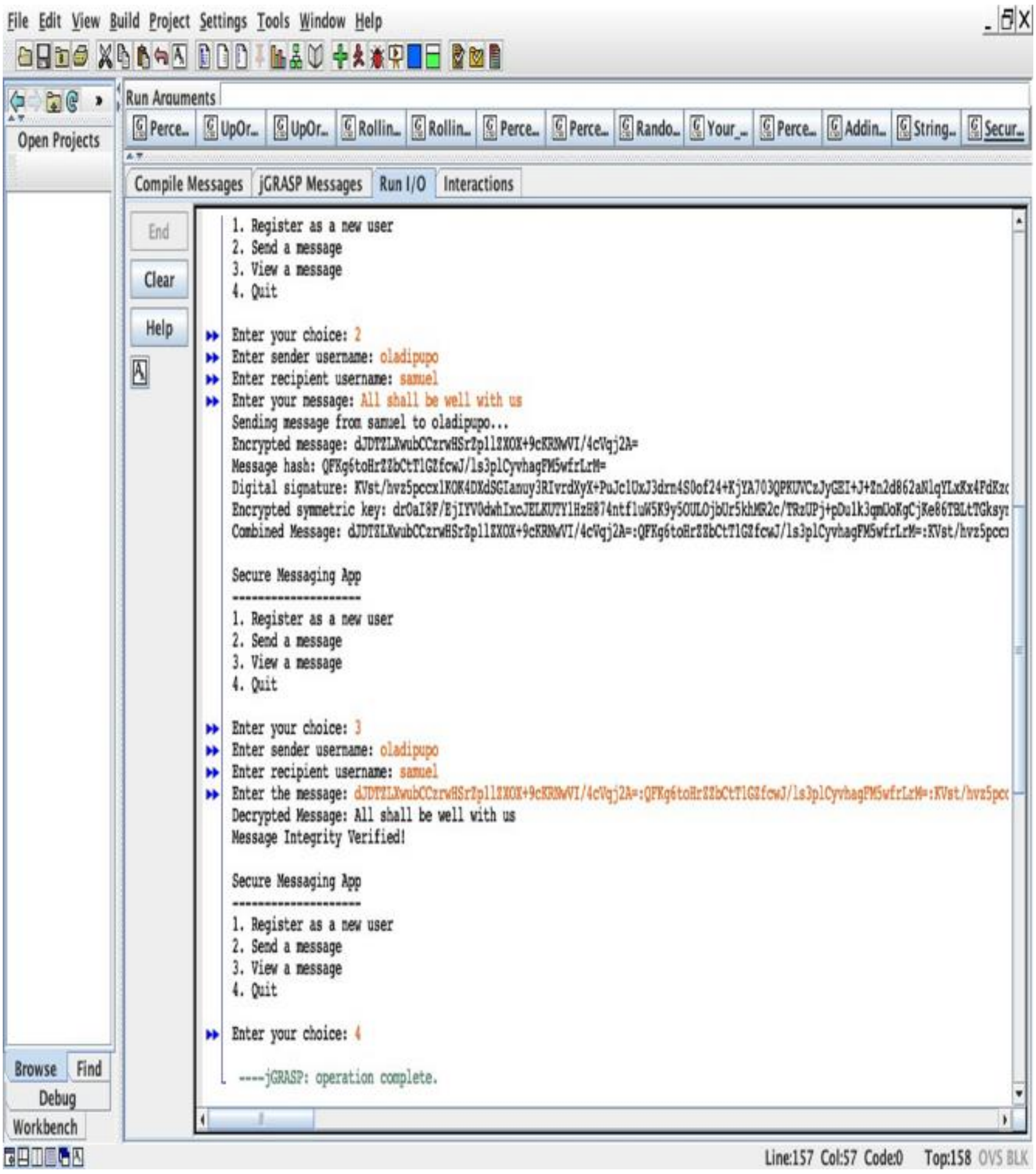
```java
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;

import javax.crypto.*;
import javax.crypto.spec.*;
import java.util.Base64;
import java.util.HashMap;
import java.util.Scanner;

public class SecureMessagingApp {
    // Database to store registered users and their public and private keys
    private static HashMap<String, KeyPair> userDatabase = new HashMap<>();

    // Main method to run the application
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\nSecure Messaging App");
            System.out.println("--------------------");
            System.out.println("1. Register as a new user");
            System.out.println("2. Send a message");
            System.out.println("3. View a message");
            System.out.println("4. Quit");
            System.out.print("\nEnter your choice: ");

            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1:
                    registerUser(scanner);
                    break;
                case 2:
                    sendMessage(scanner);
                    break;
                case 3:
                    viewMessage(scanner);
                    break;
```

Figure 1 Screenshot of the Secure Messaging App


File Edit View Build Project Settings Tools Window Help
Run Arguments
Open Projects
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
Browse
Find
Debug
Workbench
Status: running user program
Line:157 Col:57 Code:0 Top:158 OVS BLK

```
----jGRASP exec: java SecureMessagingApp

Secure Messaging App
--------------------
1. Register as a new user
2. Send a message
3. View a message
4. Quit

Enter your choice: 1
Enter your username: oladipupo
Your public key is: MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEApUrseKZ3W0igK5fdel+0nDnWIiqWgum/ZYjBVUg9cne0eb5caTZCK4mq

Secure Messaging App
--------------------
1. Register as a new user
2. Send a message
3. View a message
4. Quit

Enter your choice: 1
Enter your username: chika
Your public key is: MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAjSlNo5yD/Tg2jRtlIAQUuQ0swyOgmj57K7cD2rfKRpKJ6E8iZvXa6XN2(

Secure Messaging App
--------------------
1. Register as a new user
2. Send a message
3. View a message
4. Quit

Enter your choice: 1
Enter your username: samuel
Your public key is: MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEApCIzsbwaJukTTLALzGE62rRmKL+/P7mGLrUk3JEvCkAGDbEPOI/dRtBo

Secure Messaging App
--------------------
1. Register as a new user
2. Send a message
```

Figure 2 Screenshot of the Output of Secure Messaging App


File Edit View Build Project Settings Tools Window Help
Open Projects
Run Arguments
Perce.. UpOr.. UpOr.. Rollin.. Rollin.. Perce.. Perce.. Rando.. Your_.. Perce.. Addin.. String.. Secur..
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
Browse
Find
Debug
Workbench
Line:157 Col:57 Code:0 Top:158 OVS BLK

```
1. Register as a new user
2. Send a message
3. View a message
4. Quit

Enter your choice: 2
Enter sender username: oladipupo
Enter recipient username: samuel
Enter your message: All shall be well with us
Sending message from samuel to oladipupo...
Encrypted message: dJDTZLXwubCCzrwHSrZpllZXOX+9cKRNWVI/4cVqj2A=
Message hash: QFKg6toHrZZbCtTlGZfcwJ/ls3plCyvhagFM5wfrLrM=
Digital signature: KVst/hvz5pccxlKOK4DXdSGIanuy3RIvrdXyX+PuJclUxJ3drn4S0of24+KjYA703QPKUVCzJyGEI+J+Zn2d862aNlqYLxKx4FdKz
Encrypted symmetric key: dr0aI8F/EjIYV0dwhIxcJELKUTYlHzH874ntfluW5K9y5OULOjbUr5khMR2c/TRzUPj+pDulk3qmJoKgCjKe86TBLtTGksy
Combined Message: dJDTZLXwubCCzrwHSrZpllZXOX+9cKRNWVI/4cVqj2A=:QFKg6toHrZZbCtTlGZfcwJ/ls3plCyvhagFM5wfrLrM=:KVst/hvz5pcc

Secure Messaging App
--------------------
1. Register as a new user
2. Send a message
3. View a message
4. Quit

Enter your choice: 3
Enter sender username: oladipupo
Enter recipient username: samuel
Enter the message: dJDTZLXwubCCzrwHSrZpllZXOX+9cKRNWVI/4cVqj2A=:QFKg6toHrZZbCtTlGZfcwJ/ls3plCyvhagFM5wfrLrM=:KVst/hvz5pc
Decrypted Message: All shall be well with us
Message Integrity Verified!

Secure Messaging App
--------------------
1. Register as a new user
2. Send a message
3. View a message
4. Quit

Enter your choice: 4

 ----jGRASP: operation complete.
```

Figure 3 Screenshot of the Output of Secure Messaging App

1.3. Cryptographic techniques

Cryptographic techniques are the backbone of secure digital communication. The Secure Messaging App leverages several advanced cryptographic methods to ensure the confidentiality, integrity, authenticity, and non-repudiation of messages exchanged between users. These techniques include symmetric key cryptography, asymmetric key cryptography, hashing, and digital signatures, each serving a specific purpose in the overall security framework of the application.

1.3.1. Symmetric Key Cryptography

The application employs the Advanced Encryption Standard (AES) algorithm with a 128-bit key length for symmetric encryption. AES is widely regarded as one of the most secure encryption methods available, having been adopted as the encryption standard by the U.S. government. In symmetric key cryptography, the same key is used for both encryption and decryption, which makes it essential to keep the key secret.

In the Secure Messaging App, each message is encrypted with a unique symmetric key. This dynamic generation of keys for each message ensures that even if an attacker gains access to one symmetric key, they cannot use it to decrypt other messages. The 128-bit key length offers a robust level of security, making brute-force attacks infeasible. This method guarantees that the content of the messages remains confidential and secure during transmission.

1.3.2. Asymmetric Key Cryptography

The RSA algorithm is used for asymmetric encryption within the application, facilitating secure key exchange. Asymmetric cryptography, also known as public-key cryptography, uses a pair of keys: a public key and a private key. The public key is used for encryption, while the private key is used for decryption.

When a user sends a message, the application encrypts the symmetric key with the recipient's public key. This process ensures that only the intended recipient, who possesses the corresponding private key, can decrypt the symmetric key and subsequently the message. RSA encryption is crucial for securely sharing the symmetric key over an untrusted network, preventing unauthorized access to the key and the encrypted data.

1.3.3. Hashing

The Secure Messaging App employs the SHA-256 hash function to generate a unique hash of the message content. Hashing is a one-way cryptographic process that transforms data into a fixed-size string of characters, which is typically a digest that represents the data.

SHA-256, part of the SHA-2 family, produces a 256-bit hash value. This hash ensures data integrity by allowing recipients to verify that the message has not been altered during transmission. Even a slight change in the original message will result in a vastly different hash, making it easy to detect tampering. Hashing provides a means of ensuring that the data remains unchanged from the sender to the recipient.

1.3.4. Digital Signatures

Digital signatures in the Secure Messaging App are created using the sender’s private key to sign the message hash. This technique guarantees authenticity and non-repudiation. When the recipient receives a message, they can use the sender's public key to verify the digital signature, thus confirming the sender's identity and the integrity of the message.

Digital signatures provide proof of the sender's identity and ensure that the sender cannot deny having sent the message (non-repudiation). This aspect is particularly important in legal and financial communications, where the authenticity and integrity of the message are critical.

1.3.5. Uniqueness

The Secure Messaging App stands out by integrating multiple cryptographic techniques to provide comprehensive end-to-end security. The dynamic generation of symmetric keys for each message ensures that even if an attacker compromises one key, they cannot decrypt other messages. This approach minimizes the risk associated with key exposure. Additionally, the use of digital signatures ensures non-repudiation, making it impossible for users to deny sending a message. This multi-layered security strategy addresses various potential vulnerabilities and enhances the overall robustness of the application.

1.3.6. Usefulness

This application is particularly useful for individuals and organizations requiring secure communication channels to exchange sensitive information. The app's end-to-end security ensures that only intended recipients can decrypt and access the messages. It also maintains message integrity and authenticity, making it a reliable tool for secure communication in various professional and personal contexts. Whether it's used for corporate communications, legal correspondence, or personal messaging, the Secure Messaging App offers a high level of protection against eavesdropping, tampering, and impersonation.

2. Conclusion

The Secure Messaging App developed using the Java Cryptographic Architecture (JCA) provides a robust and comprehensive solution for secure digital communication. In today's interconnected world, the need for secure communication channels is more critical than ever. This application addresses this need by employing a combination of advanced cryptographic techniques, including symmetric encryption with AES, asymmetric encryption with RSA, cryptographic hashing with SHA-256, and digital signatures.

The integration of symmetric and asymmetric encryption ensures that messages remain confidential and secure from unauthorized access. The use of AES for symmetric encryption guarantees that each message is encrypted with a unique key, preventing any potential exposure of multiple messages through a single key compromise. RSA encryption facilitates the secure exchange of these symmetric keys, ensuring that only the intended recipient can decrypt and access the message content.

Cryptographic hashing with SHA-256 provides a robust mechanism for verifying data integrity, ensuring that messages have not been tampered with during transmission. This is complemented by digital signatures, which authenticate the sender and provide non-repudiation, ensuring that the sender cannot deny having sent the message.

The Secure Messaging App is highly beneficial for a wide range of users, from individuals to organizations, who need to exchange sensitive information securely. Its comprehensive security measures make it suitable for various use cases, including corporate communications, legal correspondence, and personal messaging. By ensuring confidentiality, integrity, authenticity, and non-repudiation, the application meets the highest standards of secure communication, providing users with peace of mind in their digital interactions.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Information Technology. security techniques. cryptographic techniques based on elliptic curves [Preprint]. doi:10.3403/02748818.

[2] Cryptography and network security, 7th edition: Stallings: 9789332585225: Amazon.com: Books. Available at: https://www.amazon.com/Cryptography-Network-Security-7Th-Stallings/dp/9332585229 (Accessed: 29 May 2024).

[3] Java Cryptography Architecture (JCA) reference guide. Available at: https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html (Accessed: 28 May 2024).

[4] Asevameh, I.O., Dopamu, O.M. and Adesiyan, J.S. (2024) Enhancing resilience and security in the U.S. power grid against cyber-physical attacks, World Journal of Advanced Research and Reviews. Available at: https://wjarr.com/content/enhancing-resilience-and-security-us-power-grid-against-cyber-physical-attacks (Accessed: 26 May 2024).

[5] Benita Urhobo (2024) 'Understanding the role of artificial intelligence in enhancing GRC practices in cybersecurity', World Journal of Advanced Research and Reviews, 22(2), pp. 269–274. doi:10.30574/wjarr.2024.22.2.1340.

[6] Diffie, W. and Hellman, M.E. (2022) 'New Directions in cryptography', Democratizing Cryptography, pp. 365–390. doi:10.1145/3549993.3550007.

[7] Dopamu, O., Adesiyan, J. and Oke, F. (2024) Artificial Intelligence and US Financial Institutions: Review of AI-Assisted Regulatory Compliance for cybersecurity, World Journal of Advanced Research and Reviews. Available at: https://wjarr.com/content/artificial-intelligence-and-us-financial-institutions-review-ai-assisted-regulatory (Accessed: 28 May 2024).

[8] IEvangelist Overview of encryption, signatures, and hash algorithms in .net - .NET, Overview of encryption, signatures, and hash algorithms in .NET - .NET | Microsoft Learn. Available at: https://learn.microsoft.com/en-us/dotnet/standard/security/cryptographic-services (Accessed: 28 May 2024).

[9] M Dopamu, O. (2024) 'Cloud - based ransomware attack on US financial institutions: An in - depth analysis of tactics and counter measures', International Journal of Science and Research (IJSR), 13(2), pp. 1872–1881. doi:10.21275/sr24226020353.

[10] Wali, A., Ravichandran, H. and Das, S. (2024) 'A 2D cryptographic hash function incorporating homomorphic encryption for secure digital signatures', Advanced Materials [Preprint]. doi:10.1002/adma.202400661.

[11] Xiao, Y. et al. (2020) 'Tutorial: Principles and practices of secure cryptographic coding in Java', 2020 IEEE Secure Development (SecDev) [Preprint]. doi:10.1109/secdev45635.2020.00016.

[12] 'Decentralized secure messaging application using blockchain technology' (2022a) International Research Journal of Modernization in Engineering Technology and Science [Preprint]. doi:10.56726/irjmets30138.

[13] 'Decentralized secure messaging application using blockchain technology' (2022b) International Research Journal of Modernization in Engineering Technology and Science [Preprint]. doi:10.56726/irjmets30138.

[14] 'Secure coding practices for java web applications' (2010) Secure Java, pp. 243–258. doi:10.1201/ebk1439823514-18.