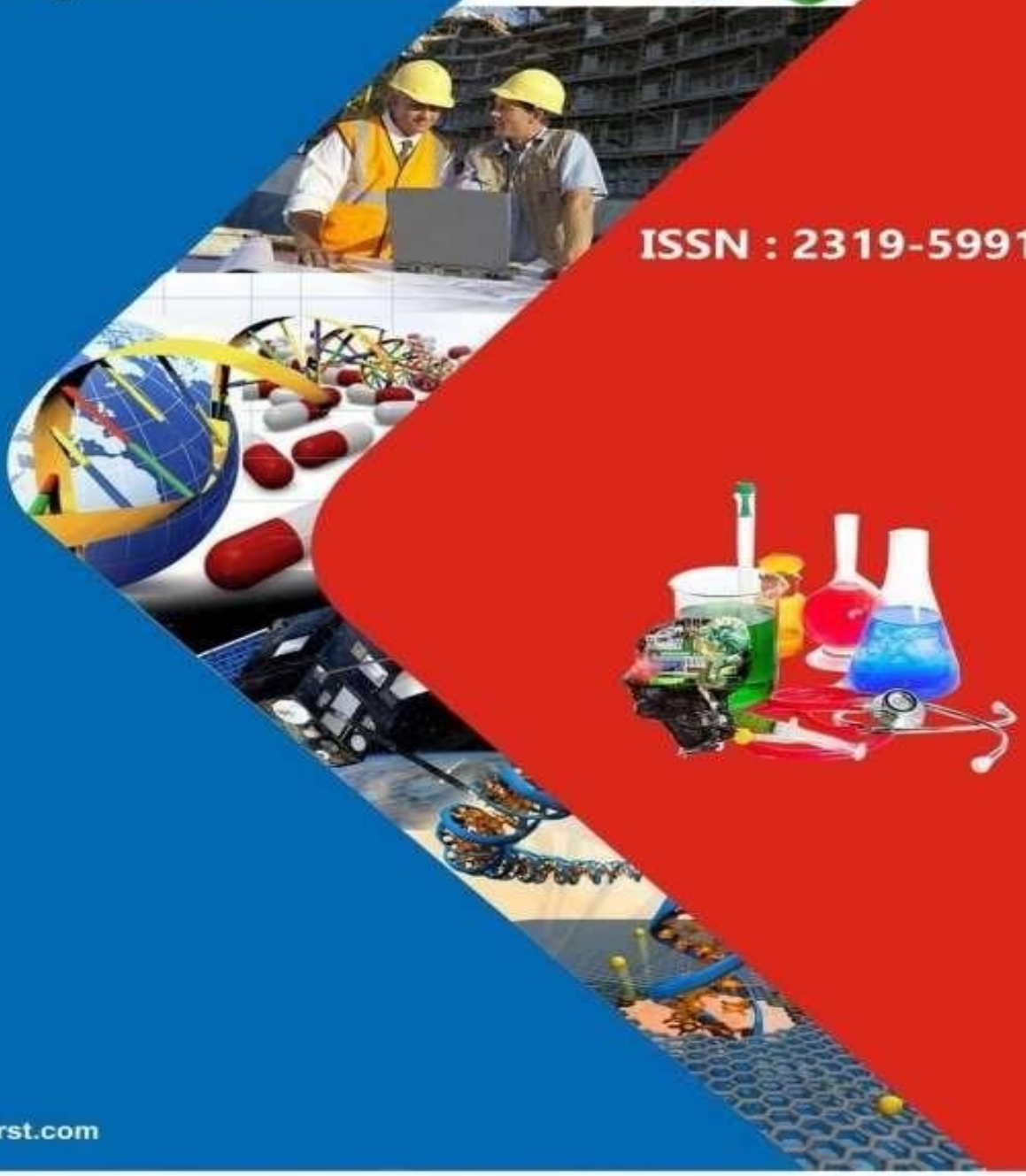
International Journal of Engineering Research and Science & Technology

IJERST

ISSN : 2319-5991



www.ijerst.com

Email: editor@ijerst.com or editor.ijerst@gmail.com

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

SECURE FILE TRANSFER USING AES & RSA ALGORITHMS

JANYAVULA SAI KUMAR

Dr. V. BHASKARA MURTHY, Professor & HOD,

Department of MCA,

B. V. Raju Colleg, Vishnupur, Bhimavaram

ADIKAVI NANNAYA UNIVERSITY RAJAMAHENDRAVARAM

I.INTRODUCTION

1.1 About the Project

The project titled **"Secure File Transfer Using RSA and AES Algorithms"** aims to develop a secure, efficient, and user-friendly system for transferring files over a network. It uses a hybrid cryptographic approach by integrating **AES**, a symmetric encryption algorithm, and **RSA**, an asymmetric encryption algorithm, to ensure both the confidentiality of the file and the secure transmission of the encryption key. In this system, the file content is encrypted using AES for fast and strong encryption, while the AES key is securely encrypted using RSA before being shared with the receiver. The receiver then decrypts the AES key using their private RSA key and uses it to decrypt the actual file. This dual-layered encryption model ensures that the file can be transferred safely even over an insecure or public network.

1.2 Purpose

The main purpose of this project is to provide a secure method for transferring files that protects sensitive data from unauthorized access, interception, and tampering. By using a combination of symmetric and asymmetric encryption, the system ensures that only the intended recipient can access the original file. The project demonstrates how cryptographic algorithms can be practically implemented to address real-world security challenges in data transmission. It is designed to be reliable, scalable, and applicable in various fields such as corporate communication, cloud storage, government data sharing, and academic document transfer.

1.3 Scope

This project covers the design, development, and implementation of a secure file transfer system using hybrid cryptography. The scope includes:

AES-based file encryption and decryption. RSA-based secure key exchange mechanism. A basic interface for users to upload, encrypt, download, and decrypt files. Protection against data breaches during file transmission. Demonstration of end-to-end encryption concepts using modern

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

cryptographic standards.

1.4 Motivation

The motivation behind this project comes from the increasing number of cybersecurity threats and data breaches that occur during digital communication and file transfer. As businesses and individuals rely more on online platforms to exchange sensitive information, the need for secure communication tools has become essential. Existing methods like traditional FTP or unencrypted email attachments are often not secure enough for transmitting confidential data. This project aims to bridge that gap by offering a simple yet powerful solution using proven cryptographic algorithms. Understanding and implementing secure file transfer also provides valuable hands-on experience in cryptography, which is a core area of modern computer science and cybersecurity.

II. LITERATURE SURVEY

Over the years, various approaches and technologies have been developed to ensure secure file transfer across networks. With the growing dependence on digital communication, researchers and developers have placed significant focus on cryptographic methods to protect data from unauthorized access and cyberattacks. This literature survey reviews existing techniques and technologies related to secure file transmission, with a focus on symmetric and asymmetric encryption algorithms.

Traditional file transfer protocols such as FTP (File Transfer Protocol) and HTTP were not designed with strong security features, making them vulnerable to threats like eavesdropping and data tampering. In response to these issues, secure variants like SFTP (SSH File Transfer Protocol) and FTPS (FTP Secure) were introduced, offering encrypted channels for communication. However, these methods primarily focus on the secure transmission channel, rather than the encryption of the file contents themselves. Several studies have shown that combining symmetric and asymmetric encryption algorithms can significantly enhance the security of data transfer systems. AES (Advanced Encryption Standard), introduced by the National Institute of Standards and Technology (NIST), has become the most widely used symmetric encryption algorithm due to its strength, speed, and efficiency in encrypting large volumes of data. On the other hand, RSA, developed by Rivest, Shamir, and Adleman, remains a cornerstone of asymmetric encryption, offering a secure means of key distribution and digital signatures.

Researchers have explored hybrid encryption systems that use AES for data encryption and RSA for key exchange to combine the advantages of both methods. In such systems,

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

the main file is encrypted using AES, ensuring fast and secure data protection, while the AES key itself is encrypted using RSA, which eliminates the need to share the key over insecure channels. This model has been successfully implemented in secure email systems, cloud storage solutions, and confidential document transmission applications.

In academic works, hybrid cryptosystems have consistently shown improved performance in terms of both security and efficiency. A common finding across multiple papers is that RSA alone is computationally intensive for large data encryption, whereas AES performs better for bulk data processing. Therefore, the combination of RSA and AES is seen as a practical solution that balances computational load and cryptographic strength.

III. PROBLEM STATEMENT

In the current digital landscape, several methods and protocols exist for transferring files between users or systems. Traditional file transfer systems such as FTP (File Transfer Protocol), HTTP, and email attachments are still widely used. However, these methods often lack strong built-in security measures, making them vulnerable to various threats like data interception, unauthorized access, and tampering during transmission.

To overcome the limitations of basic file transfer protocols, more secure alternatives such as SFTP (Secure File Transfer Protocol) and FTPS (FTP Secure) have been developed. These protocols add a layer of encryption to the transmission process by using SSL/TLS or SSH to secure the communication channel. While these methods do provide improved security, they focus on protecting the communication channel rather than encrypting the actual file content. Once the connection is established, the data is transmitted in encrypted form, but if the channel is compromised or the server is breached, the files can still be at risk.

Some file-sharing platforms and cloud services such as Google Drive or Dropbox offer basic encryption, but users have limited control over how the data is encrypted and who can access the decryption keys. In many cases, encryption is handled server-side, meaning the service provider has access to the original data, which may pose a privacy concern.

Additionally, some systems use password-protected ZIP files or simple encryption tools to secure files before transmission. However, these methods are often weak, lacking robust encryption standards or secure key management. Passwords can be guessed or cracked, and without proper key exchange mechanisms, the overall security remains compromised.

Most existing systems also do not implement

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

hybrid encryption models, which combine the speed of symmetric algorithms and the security of asymmetric algorithms. As a result, they either suffer from performance issues (if only asymmetric encryption is used) or lack secure key exchange (if only symmetric encryption is used).

3.1 Disadvantages of Existing System

Lack of End-to-End Encryption

Many existing systems secure the transmission channel (e.g., using SSL/TLS), but the actual file content is not encrypted end-to-end. If the server or storage system is compromised, the data may still be exposed.

No Secure Key Exchange

Systems that use only symmetric encryption (like password-protected ZIP files) do not offer secure methods for key exchange. Sharing keys through insecure channels defeats the purpose of encryption.

Dependence on Third-Party Trust

Cloud services such as Google Drive or Dropbox may encrypt files, but the user has limited control over encryption keys. Since the provider manages the encryption and decryption process, there's always a risk of internal data breaches or unauthorized access.

Limited User Authentication

Some file transfer systems do not provide strong user authentication, increasing the risk of unauthorized access or data misuse

Vulnerable to Man-in-the-Middle (MITM) Attacks

Without proper encryption and authentication mechanisms, traditional systems can be easily exploited by attackers who intercept and modify data during transmission.

Inadequate Protection Against Tampering

Many systems lack file integrity checks or digital signatures, making it difficult to detect whether a file has been altered during transfer.

Low Performance with Asymmetric Encryption Only

Systems that rely solely on RSA (asymmetric encryption) for full file encryption experience performance issues, especially when dealing with large files, due to RSA's computational complexity.

IV. PROPOSED SYSTEM

The proposed system is designed to provide a secure way to transfer files over a network by using a combination of two strong encryption algorithms: **AES** and **RSA**. In this system, the file is first encrypted using **AES (Advanced Encryption Standard)**, which is fast and secure for handling large amounts of data. Then, the **AES key** used to encrypt the file is itself encrypted using **RSA (Rivest–Shamir–Adleman)**, which is an asymmetric encryption algorithm that ensures the key is shared safely.

This method is known as **hybrid encryption** because it uses both symmetric and asymmetric techniques. AES provides speed and efficiency, while RSA ensures secure key

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

exchange. The receiver first decrypts the AES key using their RSA private key, and then uses that key to decrypt the actual file.

This approach makes sure that:

Only the intended receiver can access the file. Even if someone intercepts the file during transfer, they cannot read it without the decryption key. The system is fast, secure, and suitable for real-world applications. The proposed system improves upon traditional file transfer methods by offering **end-to-end encryption**, **secure key management**, and **better protection against hacking and data theft**. It is useful for securely sharing sensitive documents, personal files, business data, or any information that should remain private.

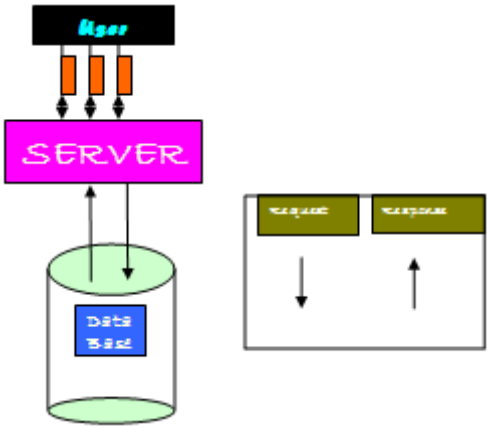
4.1 Advantages of proposed system

Uses strong encryption (AES) to protect file contents from unauthorized access. Encrypts the AES key with RSA, ensuring secure key sharing between sender and receiver. Combines the speed of AES and the security of RSA, making the system fast and safe. Provides end-to-end encryption, keeping the file secure during the entire transfer process. Prevents data theft even if the file is intercepted during transmission. Ensures that only the intended receiver can decrypt and access the file. Protects sensitive and confidential data effectively. Easy to use with a simple process for uploading, encrypting, and downloading files. Scalable and suitable for personal, academic, and professional use. Can be extended in the future with extra features like digital signatures and file integrity checks.

V. SYSTEM ARCHITECTURE

Architecture flow:

Below architecture diagram represents mainly flow of requests from users to database through servers. In this scenario overall system is designed in three tires separately using three layers called presentation layer, business logic layer and data link layer. This project was developed using 3-tire architecture.



System Architecture

VI. IMPLEMENTATION

6.1 ADMIN

In this application the admin is one of the Modules and here the admin can directly login with the application. And the admin can upload the files and view uploaded files and view request and send response.

6.2 USER

In this application the user should register with the application then only he can access into his

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

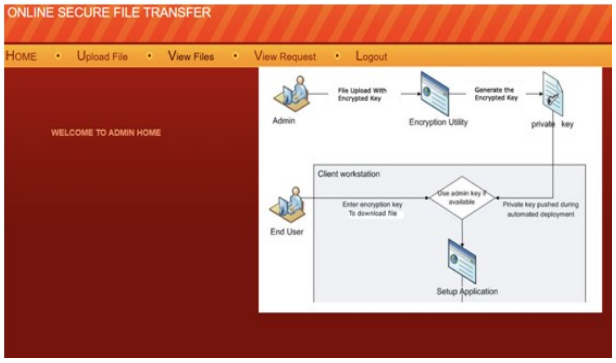
home page. Here the user can view all the files and request for the file to download the files. Here using that file-key the user can able to download the file.

VII RESULT ANALYSIS

Home page:

Home page screen

Admin Home page:



Admin Home page screen

View Request page:

VIII. CONCLUSION

The implementation of secure file transfer using RSA and AES algorithms effectively addresses the critical need for data confidentiality and protection during transmission. AES, being a fast and efficient symmetric encryption algorithm, is ideal for encrypting large files, while RSA, an asymmetric encryption algorithm, ensures the secure exchange of the AES key between sender and receiver. This hybrid encryption approach combines the strengths of both algorithms, offering a practical solution for real-world file transfer scenarios. It ensures that even if the data is intercepted, it cannot be accessed without the proper decryption keys. The project demonstrates the importance of cryptography in securing digital communication and shows how modern encryption standards can be applied to build robust and secure systems. Overall, this method enhances the trust, integrity, and safety of data exchange across networks.

IX. FUTURE WORK

For future work, several enhancements can be made to improve the efficiency, usability, and security of the secure file transfer system. One important area is the implementation of digital signatures, which would allow both sender and receiver to verify the authenticity of files and ensure data integrity. This would prevent unauthorized changes to the file content and confirm the identity of the

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

sender.

Another improvement could involve building a more user-friendly interface, making the system accessible to non-technical users while maintaining strong security in the background. Adding support for file transfer logging and audit trails would also enhance system transparency and accountability.

From a performance perspective, the system could be optimized to handle large-scale file transfers, possibly using compression techniques before encryption to reduce file size and transfer time. Multi-threading or parallel processing can also be introduced to speed up the encryption and decryption processes.

Additionally, integrating the system with cloud storage services like Google Drive or Dropbox would allow secure file sharing over cloud platforms. For wider adoption, mobile and web-based versions of the application can be developed to enable secure file transfer from any device or location. As cyber threats continue to evolve, research into advanced encryption algorithms and post-quantum cryptography can be explored to future-proof the system against emerging threats. Furthermore, integrating AI-based threat detection could help monitor and prevent suspicious file transfers in real-time.

X..BIBILIOGRAPHY

1. "UML Distilled: A Brief Guide to the Standard Object Modelling Language" by Martin Fowler
2. "Learning UML 2.0" by Russ Miles and Kim Hamilton
3. "The Unified Modelling Language User Guide" by Grady Booch, James Rumbaugh, and Ivar
4. "Head First Object-Oriented Analysis and Design" by Brett McLaughlin, Gary Pollice & David West
5. "Object-Oriented Programming in Java" by Richard L. Halterman
6. "Object-Oriented Analysis and Design with Applications" by Grady Booch
7. "Thinking in Java" by Bruce Eckel
8. "Programming Principles in Java: Architecting & Engineering Software" by Stephen Gilbert & Bill

Web References

1. **Baeldung. (2021).** Java AES Encryption and Decryption**.** Retrieved from https://www.baeldung.com/java-aes-encryption-decryption
2. **Stack Overflow. (2012).** Encrypting AES key with RSA public key**.** Retrieved from https://stackoverflow.com/questions/9658921/encrypting-aes-key-with-rsa-public-key
3. **CodeJava. (2019).** File Encryption and Decryption in Java – Simple Example. Retrieved from https://www.codejava.net/coding/file-encryption-and-decryption-simple-example
4. **GitHub. (n.d.).** Secure File Transfer System Using RSA and AES Encryption. Retrieved

International Journal of Engineering Research and Science & Technology
ISSN 2319-5991 www.ijerst.com
Vol. 21, Issue 2, 2025

from https://github.com/brunocamps/SecureFileTransfer

5. **GitHub. (n.d.).** Secure File Transfer Using Hybrid Cryptography. Retrieved from https://github.com/tejack3098/Secure-File-Sharing-Using-Hybrid-Cryptography
6. **YouTube. (2021).** AES and RSA Encryption in Java – Secure Communication Example. Retrieved from https://www.youtube.com/watch?v=WPeiI6ISo7U

References

1. Sierra, K., & Bates, B. (2005). Head First Java (2nd ed.). O'Reilly Media.
2. Schildt, H. (2018). Java: The Complete Reference (11th ed.). McGraw-Hill Education.
3. Bloch, J. (2018). Effective Java (3rd ed.). Addison-Wesley.
4. Hook, D. (2005). Beginning Cryptography with Java. Worx.
5. Stallings, W. (2017). Cryptography and Network Security: Principles and Practice (7th ed.). Pearson.
6. Goetz, B. (2006). Java Concurrency in Practice. Addison-Wesley.
7. Oaks, S. (2001). Java Security (2nd ed.). O'Reilly Media.
8. Harold, E. R. (2004). Java Network Programming (3rd ed.). O'Reilly Media.
9. Baeldung. (2021). Java AES Encryption and Decryption. Retrieved from https://www.baeldung.com/java-aes-encryption-decryption
10. Stack Overflow. (2012). Encrypting AES key with RSA public key. Retrieved from https://stackoverflow.com/questions/9658921/encrypting-aes-key-with-rsa-public-key
11. CodeJava. (2019). File Encryption and Decryption in Java – Simple Example. Retrieved from https://www.codejava.net/coding/file-encryption-and-decryption-simple-example
12. GitHub. (n.d.). Secure File Transfer System Using RSA and AES Encryption. Retrieved from https://github.com/brunocamps/SecureFileTransfer
13. GitHub. (n.d.). Secure File Sharing Using Hybrid Cryptography. Retrieved from https://github.com/tejack3098/Secure-File-Sharing-Using-Hybrid-Cryptography
14. YouTube. (2021). AES and RSA Encryption in Java – Secure Communication Example. Retrieved from https://www.youtube.com/watch?v=WPeiI6ISo7U