International Journal of Innovative Research in Science and Engineering
Volume No. 12, Issue No. 03, Year 2026
ISSN : 2454-9665

SECURE DOCUMENT UPLOADER USING ADVANCED ENCRYPTION STANDARD

Mr. A. Balaji[#1], *Assistant Professor, Department of Computer Science and Engineering, Tirumala Engineering College, Jonnalagadda, Andra Pradesh, India - 522601.*

K. Varshitha[#2], K. Alekhya[#3], K. Sri Lakshmi[#4], SK. Abdul Razak[#5]

Abstract— With the rapid growth of digital communication, secure file transfer has become a critical requirement. Traditional file uploading systems lack strong security mechanisms, making them vulnerable to cyber threats such as unauthorized access, data breaches, and malware attacks. This paper presents a Secure Document Uploader System using Advanced Encryption Standard (AES) to ensure data confidentiality and integrity. The system integrates secure authentication, encrypted file storage, HTTPS communication, and role-based access control. Files are encrypted before storage and decrypted only for authorized users. Experimental results show that the system provides high security with minimal performance overhead, making it suitable for real-world applications such as organizations, educational institutions, and cloud storage systems.

Keywords—Secure File Upload, AES Encryption, Cybersecurity, HTTPS, Authentication, Data Protection, Web Security

I. INTRODUCTION

In today's digital era, file sharing has become an integral part of communication and operations across various sectors such as education, business, healthcare, and governance. Organizations frequently exchange sensitive data including documents, images, financial records, and confidential reports over the internet. However, traditional file upload systems are often designed without robust security mechanisms, making them highly vulnerable to cyber threats such as data interception, unauthorized access, malware injection, and man-in-the-middle attacks.

Most existing systems rely on basic HTTP protocols, which transmit data in plain text, and they do not implement encryption for files stored on servers. As a result, attackers can easily capture sensitive information during transmission or gain access to stored files if the system is compromised. These weaknesses lead to serious security risks including data breaches, identity theft, unauthorized file access, and the spread of malicious files within the system.

To overcome these challenges, a secure file uploading system must incorporate multiple layers of protection. This includes encryption at both rest and in transit to ensure data confidentiality, strong authentication mechanisms to verify user identity, role-based access control to restrict unauthorized access, and file validation techniques to prevent malicious uploads. Additionally, secure communication protocols such as HTTPS should be used to safeguard data during transmission.

This paper proposes a Secure File Uploader System that integrates Advanced Encryption Standard (AES) for file encryption, HTTPS for secure communication, and robust authentication mechanisms to enhance system security. The proposed solution aims to provide a reliable, efficient, and scalable platform for secure file transfer and storage, thereby reducing vulnerabilities and ensuring data protection in modern web applications.

II. LITERATURE SURVEY

Security in file upload systems is supported by research in cryptography, web security, and access control. W. Stallings, in *Cryptography and Network Security*, explains cryptographic algorithms such as AES, DES, and RSA, highlighting their role in ensuring data confidentiality and integrity. Similarly, Bruce Schneier's *Applied Cryptography* focuses on secure communication protocols and key management, though implementation can be complex.

The Advanced Encryption Standard (AES), introduced by NIST, is widely used for secure data encryption due to its strong security and efficiency. In addition, OWASP Web Security Guidelines provide best practices to prevent common vulnerabilities such as SQL injection, cross-site scripting (XSS), and session hijacking, which are essential for secure web application development.

Secure communication protocols like SSH emphasize encryption during data transmission and have influenced the use of HTTPS with SSL/TLS in modern systems. Cloud security frameworks further stress the importance of protecting data both at rest and in transit. Moreover, Role-Based Access Control (RBAC) enhances security by restricting system access based on user roles.

III. EXISTING SYSTEM

Traditional file upload systems are commonly used in many web applications but lack adequate security mechanisms. These systems typically rely on the HTTP protocol, which does not provide encryption, resulting in data being transmitted in plain text. Uploaded files are often stored directly on the server without encryption, making them vulnerable to unauthorized access. Additionally, such systems use weak authentication methods and do not implement proper file validation techniques, allowing malicious files to be uploaded.

Due to these limitations, traditional systems face several security risks. They do not provide encryption at rest or during data transmission, making sensitive information easily accessible to attackers. They are also vulnerable to common web attacks such as SQL injection due to improper input validation. Furthermore, the absence of role-based access control allows users to access data beyond their authorization level. These weaknesses make traditional file upload systems highly insecure and unsuitable for handling sensitive information.

IV. PROPOSED SYSTEM

The proposed Secure File Uploader System is designed to overcome the limitations of

traditional file upload systems by incorporating advanced security mechanisms. The system ensures secure file transfer and storage by implementing encryption, authentication, and access control techniques. It uses HTTPS protocol with SSL/TLS to protect data during transmission, preventing interception and man-in-the-middle attacks.

In this system, files are encrypted using the Advanced Encryption Standard (AES) before being stored on the server, ensuring data confidentiality even if unauthorized access occurs at the storage level. Strong user authentication mechanisms, including password hashing and secure session management, are implemented to verify user identity. Additionally, role-based access control (RBAC) is used to restrict access to files based on user roles, ensuring that only authorized users can upload, download, or manage files.

The system also incorporates file validation techniques such as checking file type and size to prevent malicious uploads. All user inputs are properly validated to protect against attacks like SQL injection and cross-site scripting (XSS). The overall architecture follows a client-server model where the client interface allows user interaction, while the server handles encryption, storage, and security processes.

By integrating multiple layers of security, the proposed system provides improved confidentiality, integrity, and availability of data. It offers a reliable, scalable, and efficient solution for secure file uploading and sharing in modern web applications.

Fig: System Architecture

Modules of Proposed System

The proposed Secure File Uploader System is divided into several modules to ensure efficient functionality and strong security.

1. User Authentication Module

This module handles user registration and login. It verifies user credentials using secure password hashing and manages user sessions. Only authenticated users are allowed to access the system.

2. File Upload Module

This module allows users to upload files to the system. It receives the file from the user interface and forwards it for validation and encryption before storage.

3. File Validation Module

Before uploading, the system checks the file type, size, and content to prevent

malicious files. Only allowed file formats are accepted, ensuring system safety.

4. Encryption Module

This module encrypts files using the Advanced Encryption Standard (AES) before storing them on the server. It ensures that files remain secure and unreadable without the correct decryption key.

5. File Storage Module

Encrypted files are stored securely in the server. File metadata such as file name, size, and upload details are stored in the database.

6. File Download and Decryption Module

When a user requests a file, this module retrieves the encrypted file and decrypts it using the appropriate key before delivering it to the user.

7. Access Control Module (RBAC)

This module restricts access to files based on user roles. It ensures that only authorized users can view, upload, or download files.

8. Security and Communication Module

This module ensures secure data transmission using HTTPS and protects the system from attacks like SQL injection and cross-site scripting (XSS).

V. METHODOLOGY

The methodology of the Secure File Uploader System is centered around the use of the Advanced Encryption Standard (AES) to ensure secure file storage and transmission. The system follows a step-by-step process where security is applied at every stage of file handling.

Initially, the user registers and logs into the system through a secure authentication process. Passwords are hashed before being stored in the database to prevent unauthorized access. Once authenticated, the user can upload files through the web interface.

Before uploading, the file is validated by checking its type and size to prevent malicious content. After validation, the file is processed by the encryption module, where AES is applied. AES is a symmetric key encryption algorithm that uses a single secret key for both encryption and decryption. In this system, a unique 128-bit key is generated for each file. The file data is converted into binary format and encrypted through multiple rounds of transformation, producing ciphertext that is unreadable without the correct key.

The encrypted file is then stored securely on the server, while the encryption key and file metadata are stored in the database in a protected manner. During transmission, HTTPS with SSL/TLS is used to ensure that the data is encrypted in transit.

When a user requests to download a file, the system first verifies the user's access permissions. The encrypted file is retrieved from storage and passed to the decryption process. Using the same AES key, the ciphertext is converted back into the original file format and delivered to the user.

By integrating AES encryption into the workflow, the system ensures strong data confidentiality and protection against unauthorized access. This methodology provides a secure and efficient approach for handling sensitive files in web-based applications.

AES Encryption

Encryption:
C = AES(K, P)

Decryption:
P = AES^{-1}(K, C)

Where:
P = Plaintext
C = Ciphertext
K = Key

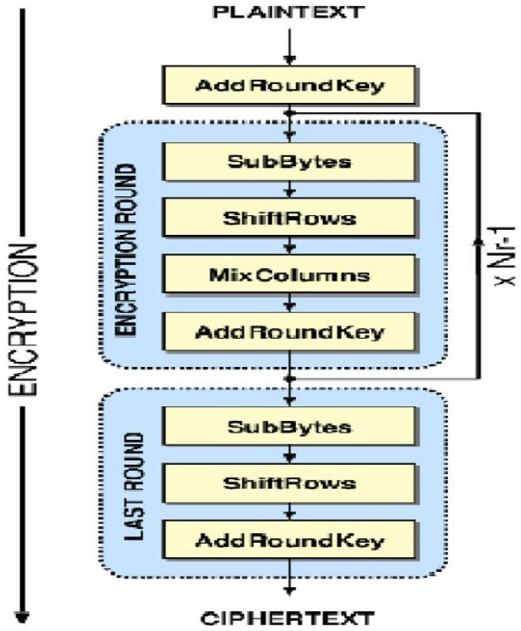

Fig: Working of AES

VI. RESULT ANALYSIS

The performance and efficiency of the Secure File Uploader System were evaluated based on encryption time, decryption time, upload performance, and system response. The results demonstrate that the system provides strong security with acceptable performance overhead.

A. Encryption Performance

The encryption time increases proportionally with file size, showing linear behavior.

File Size (MB)	Encryption Time (sec)
1	0.2
5	0.9
10	1.8
20	3.5
50	8.2

The results indicate that AES encryption follows approximately **O(n)** time complexity, where execution time increases linearly with file size.

B. Decryption Performance

Decryption also shows linear performance and is slightly faster than encryption.

File Size (MB)	Decryption Time (sec)
1	0.18
5	0.85
10	1.7
20	3.3
50	7.9

C. Upload Performance Comparison

A comparison between normal upload and secure upload is shown below:

File Size (MB)	Normal Upload (sec)	Secure Upload (sec)
5	0.5	0.9
10	1.0	1.8
20	2.0	3.5

The secure upload introduces additional time due to encryption, but the overhead is reasonable considering the security benefits.

D. System Response Time

The system maintains efficient response times under normal conditions.

Operation	Response Time (sec)
Login	0.3
Upload (5 MB)	0.9
Download (5 MB)	0.85
Logout	0.1

VII. CONCLUSION

The Secure File Uploader System provides a reliable and efficient solution for secure file management in modern digital environments. With the increasing risk of cyber threats such as unauthorized access and data breaches, the system integrates essential security mechanisms including authentication, encryption, access control, and secure storage.

The system implements secure user authentication using hashed passwords and applies AES encryption to protect files during storage. Role-Based Access Control (RBAC) ensures that users can access only authorized data, while administrators maintain system control. In addition, input validation and security measures help prevent common web vulnerabilities such as SQL injection, cross-site scripting (XSS), and malicious file uploads.

The architecture combines frontend, backend, database, and cryptographic components to provide a complete and scalable solution. The system also offers a user-friendly interface, ensuring ease of use while maintaining strong security standards.

Performance evaluation shows that the system operates efficiently with minimal delay in file upload, encryption, and download processes, while maintaining data integrity and secure communication.

In conclusion, the proposed system successfully achieves a balance between security, performance, and usability. It is suitable for applications in educational institutions, organizations, and small-scale enterprises, and provides a strong foundation for future enhancements such as cloud integration, multi-factor authentication, and advanced threat detection.

VIII. REFERENCES

[1] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 7th ed. Upper Saddle River, NJ, USA: Pearson, 2017.

[2] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd ed. New York, NY, USA: John Wiley & Sons, 1996.

[3] D. R. Stinson and M. B. Paterson, *Cryptography: Theory and Practice*, 4th ed. Boca Raton, FL, USA: CRC Press, 2018.

[4] National Institute of Standards and Technology (NIST), “FIPS PUB 197: Advanced Encryption Standard (AES),” Nov. 2001.

[5] National Institute of Standards and Technology (NIST), “FIPS PUB 180-4: Secure Hash Standard (SHS),” Aug. 2015.

[6] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE*

Transactions on Information Theory, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[7] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120– 126, Feb. 1978.

[8] OWASP Foundation, "OWASP Top 10 – Web Application Security Risks," 2021.

[9] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2002.

[10] M. Bishop, *Computer Security: Art and Science*, 2nd ed. Boston, MA, USA: Addison-Wesley, 2018.

[11] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed. Hoboken, NJ, USA: Wiley, 2018.

[12] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST Special Publication 800-145, Sept. 2011.

[13] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2014.

[14] ISO/IEC 27001, *Information Technology – Security Techniques – Information Security Management Systems – Requirements*, 2013.

[15] W. Stallings and L. Brown, *Computer Security: Principles and Practice*, 4th ed. Upper Saddle River, NJ, USA: Pearson, 2018.