Cloud Security Framework For Encrypted File Sharing With QR Authentication

Dr.A.Karunamurthy[1], R.Anandhi[2]
[1] Professor, Department of computer Applications, Sri Manakula Vinayagar Engineering College(Autonomous), Puducherry 605008, India
[2]Post Graduate student, Department of computer Applications,Sri Manakula Vinayagar Engineering College (Autonomous), Puducherry 605008, India

Abstract- *With the growing adoption of cloud storage for secure file management, ensuring data confidentiality, integrity, and controlled access has become increasingly important. This project, "Cloud Security Framework for Encrypted File Sharing with QR Authentication," introduces a secure and scalable system that integrates AES (Advanced Encryption Standard) encryption and QR code-based authentication to protect sensitive information. Files are encrypted before being uploaded to the cloud, and a unique encryption key is generated for each file, ensuring data remains secure during storage and transmission. QR code-based verification ensures that only authenticated users can access and decrypt the files. To further enhance data integrity, the system leverages homomorphic hashing over an FTP-based cloud infrastructure, enabling secure and verifiable file integrity checks without decrypting the data. The system also maintains detailed logs of file access attempts and approvals, ensuring transparency and accountability. By combining encryption, QR-based authentication, and cloud storage with secure integrity validation, this framework provides a robust, efficient, and secure solution for managing and sharing sensitive files in cloud environments. The development of this project leverages HTML and Bootstrap (CSS) for the front-end, ensuring a visually appealing and responsive design. On the backend, Java and MySQL have been employed to establish a robust foundation for data management and system functionality.*

KeyWords: *Cloud Security, Encrypted File Sharing, QR Code Authentication, AES Encryption,Data Confidentiality, Data Integrity,Controlled Access, Homomorphic Hashing, FTP-based Cloud Infrastructure ,File Integrity Verification, Authentication , Secure File Transmission*

I. INTRODUCTION

Nowadays, cloud storage has become a popular way to store and share files because it is easy to use and allows access from anywhere. But as more and more important and personal data is being stored in the cloud, security has become a serious concern. To solve these issues, our project titled "Cloud Security Framework for Encrypted File Sharing with QR Authentication" introduces a secure system for uploading, storing, and sharing files over the cloud. In this system, files are first encrypted using AES (Advanced Encryption Standard) before uploading, so that no one can read the file without the correct key. Along with this, QR code-based authentication is used to make sure that only the right person can access and decrypt the file. By combining encryption, QR code verification, secure cloud storage, and activity tracking, our project aims to provide a complete and safe platform for sharing sensitive information online.

In the modern digital era, cloud computing has revolutionized data storage and sharing by offering scalable, on-demand resources. However, with the growing reliance on cloud services, security and privacy have become critical concerns, especially when handling sensitive information. Traditional authentication mechanisms and file-sharing methods often fall short in preventing unauthorized access and data breaches.

To address these challenges, this paper proposes a Cloud Security Framework for Encrypted File Sharing with QR Authentication, aimed at enhancing the confidentiality, integrity, and availability of data shared over the cloud. The framework integrates advanced encryption techniques for secure file storage and QR code-based two-factor authentication to ensure that only authorized users can access the shared files.

II. LITERATURE SURVEY

Several studies and existing systems have addressed the challenges of secure data sharing in cloud environments. The use of AES encryption has been widely adopted for ensuring confidentiality of data during storage and transmission, as seen in various secure file-sharing solutions. Research by Goyal (2020) emphasizes the importance of client-side encryption in giving users control over their data before it reaches the cloud. In terms of authentication, Liu et al. (2021) proposed a QR code-based authentication

mechanism, which significantly improves user verification while maintaining ease of access, especially for mobile users. For ensuring data integrity, Wang (2013) introduced the concept of homomorphic hashing to enable verification of data without the need for decryption, a breakthrough in maintaining both privacy and integrity. Additionally, the role of secure FTP infrastructures in cloud deployment was explored by Kumar (2019), highlighting the importance of layered security when using traditional protocols. Finally, the significance of access logging and audit trails for ensuring transparency and accountability in cloud environments was underlined by Zhang (2018), suggesting that secure systems should include robust logging mechanisms for monitoring and compliance. These foundational works inform and support the design of the proposed Cloud Security Framework, combining encryption, authentication, and integrity validation for secure file sharing. Security in cloud computing continues to be a critical concern as data breaches and unauthorized access incidents rise. Numerous researchers have emphasized the need for multi-layered security architectures that incorporate both encryption and robust access control. For example, Singh and Chandrasekaran (2017) proposed a hybrid encryption model combining AES with RSA to balance performance and key management. Their approach provided enhanced protection for sensitive documents stored in public cloud platforms. Additionally, studies have explored the integration of biometric and QR code-based techniques for two-factor authentication, demonstrating increased resistance to spoofing and phishing attacks. The use of dynamic QR codes, which regenerate periodically, further strengthens the security posture against session hijacking and replay attacks. In terms of data verification and trust, modern cloud systems are increasingly implementing provable data possession (PDP) and auditable logging techniques. Ateniese et al. (2007) pioneered the PDP model, which allows cloud users to verify that their files are still intact and stored correctly without retrieving the full dataset. This concept laid the groundwork for later innovations such as homomorphic verification techniques, which allow computations on encrypted data. Moreover, researchers like Ren et al. (2016) have introduced blockchain-based audit trails for cloud data access, ensuring immutability and verifiability of logs. These techniques are particularly useful in multi-user environments where accountability and traceability are essential for regulatory compliance and user trust.

III.PROBLEM STATEMENT

With the rapid adoption of cloud storage for personal and organizational data, ensuring secure, private, and controlled file sharing has become a major challenge. Traditional cloud storage services often lack adequate security mechanisms to prevent unauthorized access, data breaches, and tampering, especially during file transmission and storage. Moreover, basic authentication methods (e.g., passwords) are increasingly vulnerable to attacks, compromising data confidentiality. In addition, many cloud platforms fail to provide efficient methods for verifying the integrity of stored files without exposing the data.There is a critical need for a secure, scalable, and user-friendly framework that ensures end-to-end encryption, reliable user authentication, and tamper-proof integrity validation, while maintaining performance and usability. This project aims to address these issues by developing a Cloud Security Framework that integrates AES encryption, QR code-based authentication, and homomorphic hashing over a cloud storage infrastructure, ensuring secure file sharing, integrity verification, and access transparency Current cloud storage services often place the responsibility of data security solely on the service provider, which may not align with the user's expectations or compliance requirements, especially in sensitive sectors like healthcare, finance, and government. Users have limited control over how their data is encrypted, stored, and accessed. Moreover, file-sharing mechanisms provided by default in many platforms lack granular access control and do not offer adequate visibility into who accessed what data and when. This lack of transparency and control raises concerns about data ownership, accountability, and traceability. Furthermore, while encryption provides data confidentiality, it introduces a new challenge—how to verify the integrity of encrypted files without decrypting them. Traditional hashing methods require plaintext access, making them unsuitable for encrypted cloud data. In addition, systems that rely solely on passwords or token-based authentication are susceptible to compromise, especially in phishing or credential-stuffing attacks. The absence of multi-factor and context-aware authentication further weakens data security. These limitations underline the need for a comprehensive framework that not only encrypts files before upload but also ensures secure access through QR code authentication, non-intrusive integrity checks, and detailed access logging to maintain trust, security, and transparency in cloud file sharing.

IV. PROPOSED SYSTEM

The proposed system aims to provide a highly secure and reliable way to share files using cloud storage. In this system, files are encrypted using AES (Advanced Encryption Standard) before they are uploaded to the cloud. This ensures the system even more secure, QR code-based authentication is used. Only the user who scans the correct QR code can access and decrypt the file. This adds an extra layer of protection beyond regular login. The system also uses homomorphic hashing, which allows the user to check the integrity of a file

without the need to decrypt it. This saves time and maintains security. Additionally, every file access or activity is recorded in a log, so users and administrators can keep track of who accessed which file and when. This ensures full transparency and accountability. With these features, the proposed system offers a complete, secure, and user-friendly solution for storing and sharing sensitive files in the cloud.

Advantages of proposed system

- **Strong File Protection:**
  Files are encrypted using AES before uploading, which keeps the data safe from unauthorized access.
- **QR Code-Based Authentication:**
  Only users who scan the valid QR code can access and decrypt the file, making login more secure than just using a password.
- **File Integrity Checking Without Decryption:**
  Homomorphic hashing allows users to check if a file has been changed or tampered with, without needing to decrypt it.
- **Complete Activity Tracking:**
  All file access, approvals, and downloads are recorded, helping users monitor their data and ensuring accountability.
- **User-Friendly and Scalable:**
  The system is easy to use and can be expanded for larger user bases or more files without losing performance or security.

V. ARCHITECTURAL DESIGN

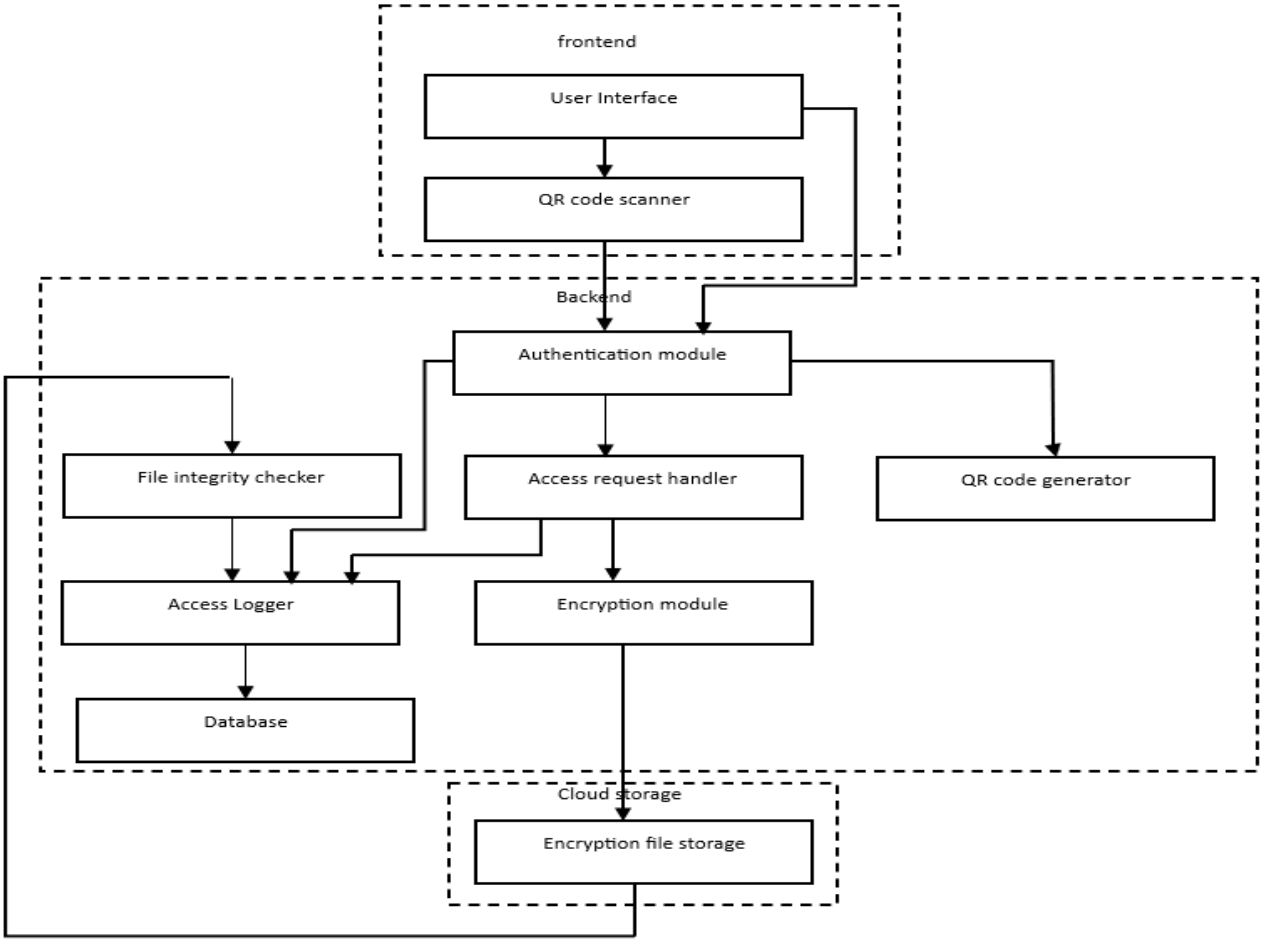

Fig1:Architectural design diagram

The architectural design of this project is based on a layered and modular architecture, ensuring separation of concerns, enhanced security, and scalability. At the top level is the Presentation Layer, which includes the user interface (web or mobile) for both file owners and recipients. This layer handles user inputs for login, file upload, encryption, QR code scanning, and file download. It communicates with the backend through secure APIs, ensuring that no sensitive

operations like encryption or key handling occur directly in the front-end.

Beneath the interface lies the Application Layer, which contains the core business logic of the system. This layer includes services such as the Encryption/Decryption Module, QR Code Generator, User Authentication Service, and Key Management Service. These components interact to ensure that files are encrypted before upload, QR codes are generated securely, and only authenticated users can access the decryption keys. The use of one-time QR codes, token validation, and expiration mechanisms ensures that even if a QR code is leaked, it cannot be reused or exploited.

VI. TECHNOLOGIES USED

Java (Backend Logic & Security) Java is used as the primary backend language due to its strong security features, platform independence, and robust support for file handling and networking. In this project, Java is responsible for implementing core functionalities such as **AES encryption and decryption**, **QR code generation**, **user authentication**, and **interaction with the FTP-based cloud storage**. Its object-oriented nature also helps in building a modular and maintainable codebase.

MySQL (Database Management) MySQL is used as the **relational database system** to manage and store all critical information including **user credentials**, **file metadata**, **encryption keys**, and **access logs**. The database ensures data consistency, fast retrieval, and secure storage, supporting key operations such as user verification and tracking access activity.

HTML (Web Structure) HTML (HyperText Markup Language) forms the **foundation of the front-end interface**, defining the structure of web pages used for login, registration, file upload, and access control. It enables the integration of input forms, display elements, and links that connect users to system functionality.

CSS & Bootstrap (Styling & Responsiveness) CSS is used to style the web pages and improve the overall user experience. The project also uses **Bootstrap**, a popular CSS framework, to create a **responsive and mobile-friendly interface**. With its pre-built components and grid system, Bootstrap enhances visual appeal and ensures that the application functions well across different devices.

JavaScript (Interactivity & Frontend Logic) JavaScript adds **dynamic behavior and interactivity** to the front end. It is used for client-side form validation, QR code scanning and rendering, and asynchronous interactions. JavaScript ensures a smoother and more responsive user experience by minimizing page reloads and handling events in real time.

VII. PROPOSED TECHNIQUES

Step 1: User Registration and Login

- Users register through a secure web interface.
- Login credentials are stored securely in the **MySQL database**.
- During login, credentials are verified, and a secure session is initiated.

Step 2: File Upload with AES Encryption

- The user selects a file to upload.
- Before uploading, the file is encrypted on the client or server side using **AES (Advanced Encryption Standard)**.
- A **unique encryption key** is generated for each file.
- The encrypted file is then prepared for storage.

Step 3: Secure File Storage via FTP

- The encrypted file is uploaded to the **FTP-based cloud storage server**.
- The system ensures secure transmission using encryption protocols layered over FTP.
- Metadata (file name, encryption key, timestamp, owner info) is stored in the **MySQL database**.

Step 4: File Access Request by User

- When a user wants to access a file, a request is initiated via the web interface.
- The system checks access permissions and prepares a **QR code for authentication**.

Step 5: QR Code-Based Authentication

- A **unique QR code** is generated for the requested file session.
- The user scans the QR code using a registered mobile device.
- The scanned code is validated to confirm the user's identity and session legitimacy.

Step 6: Integrity Verification Using Homomorphic Hashing

- Before file decryption or download, the system performs **homomorphic hashing**.
- It verifies that the encrypted file in storage has not been altered or tampered with.
- This ensures **data integrity without needing decryption**.

Step 7: File Decryption and Download

- Once the file's integrity is confirmed and the user is authenticated, the file is decrypted using the corresponding **AES key**.
- The decrypted file is then made available for download to the user.

Step 8: Logging and Monitoring

- Every upload, access request, authentication attempt, and file operation is **logged in the MySQL database**.
- Admins can review logs for **auditing, accountability, and detecting suspicious activities**.

VIII. CONCLUSION AND FUTURE ENHANCEMENTS

This project, "Cloud Security Framework for Encrypted File Sharing with QR Authentication," successfully provides a secure way to share sensitive files using cloud storage. By combining AES encryption, QR code-based authentication, and homomorphic hashing, the system ensures that files are protected from unauthorized access and tampering. Users can confidently upload, share, and access files knowing their data is safe. The detailed logging and monitoring features add transparency and help prevent misuse. Overall, this system improves the security and trustworthiness of cloud file sharing, making it suitable for personal and professional use.

The Cloud Security Framework for Encrypted File Sharing with QR Authentication successfully combines advanced cryptographic techniques with modern authentication methods to address the growing concerns of data security in cloud environments. By implementing end-to-end encryption, the system ensures that files remain confidential throughout their lifecycle — from the moment they are uploaded, during storage, and finally at the point of access or download. This eliminates the risks posed by unauthorized access, data breaches, and interception, which are prevalent in traditional cloud file-sharing systems.

Future Enhancements

The Application can have the following enhancements. In this project, many future plans may be implemented to this process. However future thinking has expanded hugely in scope and substance. This system will expand to more future plans.

- Multi-Factor Authentication Adding other authentication methods like OTP or biometric verification for extra security.
- Mobile Application Support Developing mobile apps to allow users to securely access files on their smartphones.
- Advanced Encryption Techniques Using newer encryption algorithms to further strengthen data protection.
- Real-Time Alerts Sending instant notifications to users when their files are accessed or requested.
- Cloud Storage Integration Supporting multiple cloud storage services like AWS, Google Cloud, or Azure for better scalability.

REFERENCES

[1] Stallings, William. Cryptography and Network Security: Principles and Practice. 7th Edition, Pearson, 2017. — A comprehensive resource on cryptographic algorithms and secure communication protocols.

[2] Kaufman, Charlie, Radia Perlman, and Mike Speciner. Network Security: Private Communication in a Public World. 2nd Edition, Prentice Hall, 2002. — This book covers fundamental concepts of network security, including encryption and authentication methods.

[3] Schneier, Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 20th Anniversary Edition, Wiley, 2015. — A classic reference for practical cryptography implementations.

[4] Dierks, Tim, and Eric Rescorla. "The Transport Layer Security (TLS) Protocol Version 1.2." RFC 5246, IETF, 2008. — The official specification of TLS, important for secure communication channels.

[5] National Institute of Standards and Technology (NIST). Special Publication 800-63B: Digital Identity Guidelines — Authentication and Lifecycle Management, 2017. — Guidelines on digital identity and authentication best practices.

[6] Zhang, Wei, et al. "Secure and Efficient Cloud Storage Systems with Encrypted Data Sharing." IEEE Transactions on Cloud Computing, vol. 6, no. 3, 2018, pp. 733–745. — Research paper discussing encrypted data sharing techniques in cloud storage.

[7] Gao, Ying, and Yan Qian. "QR Code-Based Authentication Systems: A Survey." Journal of Network and Computer Applications, vol. 123, 2018, pp. 17-28. — Survey paper on various QR code authentication methods and applications.

[8] Microsoft Azure Documentation. "Best Practices for Cloud Security." Accessed April 2025. https://docs.microsoft.com/en-us/azure/security/fundamentals/ — Industry best practices for implementing security in cloud environments.

[9] OWASP Foundation. OWASP Top Ten Web Application Security Risks, 2021. — A critical resource for understanding common web application vulnerabilities and mitigation strategies.

[10] Google Cloud Platform. “Encryption at Rest and in Transit.” Accessed April 2025. https://cloud.google.com/security/encryption-at-rest
— Details on Google Cloud’s encryption methods for data protection