IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Secure File Sharing With QR Code Generation, Time-Bound Access Control, And Data Encryption

Prof. Sharad Jadhav, Mr. Piyush Kulkarni, Mr. Devesh Kashikar, Miss Shreya Kumari, Mr. Sumit Ingale
Department of Computer Engineering
Dr. D. Y. Patil College of Engineering and Innovation, Pune, India

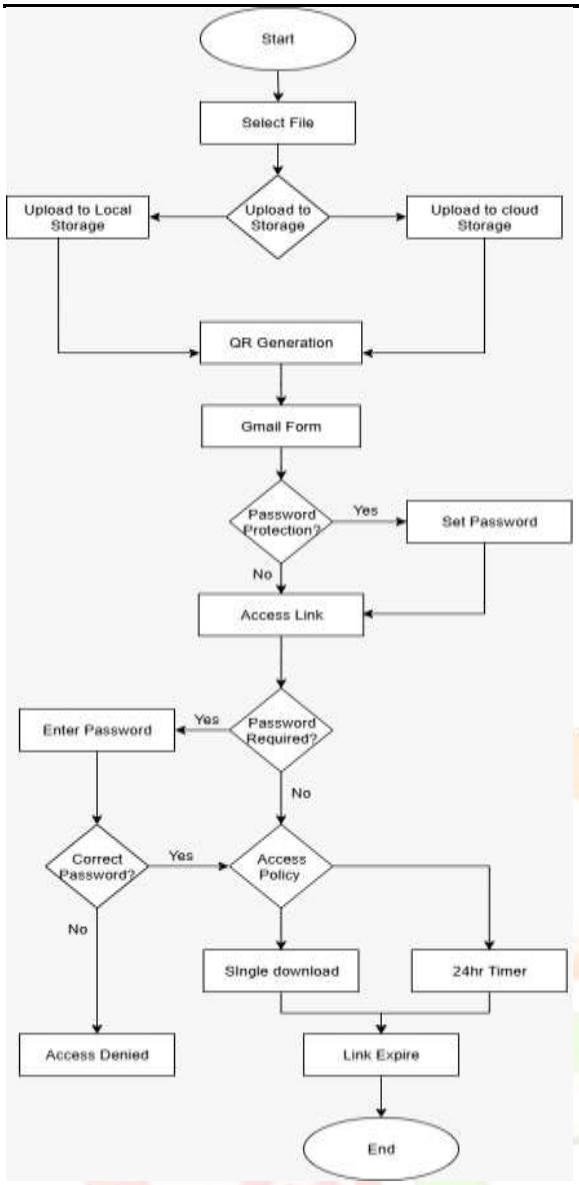
Abstract

In today's highly digital world, **data security** has become paramount for all information systems. Conventional file-sharing methods often lack sufficient protection, leaving sensitive information vulnerable to unauthorized access or long-term exposure. This paper introduces a **secure file-sharing framework** that effectively integrates **QR code generation**, optional **encryption**, and **time-bound access control**. The proposed system employs **AES encryption** for strong confidentiality, **dynamic QR codes** for accessible portability, and a **time-based expiry** mechanism to automatically revoke authorization. This combination ensures that files are only accessible to verified users within a strictly limited time window. Implemented using **React and Firebase**, the architecture delivers a secure, usable, and scalable solution suitable for secure data transmission in academic, corporate, and personal environments.

I. Introduction

The rapid growth of digital communication and cloud technologies has fundamentally changed how individuals and organizations exchange information. While offering immense convenience, this shift has also brought about increased security risks. **Unauthorized data access, interception, and misuse** are significant concerns with traditional sharing methods, such as persistent cloud links or email attachments. A critical flaw in these older methods is the **indefinite validity** of the sharing link, making it susceptible to unauthorized redistribution over time.

QR codes are a promising alternative because of their ability to encode digital information into a scannable and portable format, facilitating secure and temporary data sharing. However, static QR codes still present a risk if they are intercepted. To effectively counter this threat, our proposed model integrates **robust cryptographic protection** and strict **time-bound access control** measures, ensuring file sharing is both secure and precisely controlled.



A. Objectives of the Study

The core objectives guiding the development of this secure file-sharing system were:

1. To design a secure file-sharing system that uses **QR code generation** for simplified access.
2. To provide **optional, robust AES-based encryption** specifically for highly sensitive data.
3. To enforce a strict **time-bound access policy** for all shared files to limit exposure duration.
4. To ensure the final system achieves an optimal **balance between essential security features and end-user usability**.

This comprehensive strategy addresses common challenges in digital file sharing by guaranteeing **confidentiality, authenticity, and strictly limited-time accessibility**.

B. Secure File Sharing Process Flow

The file-sharing process is systematically structured into distinct, mandatory steps to ensure data integrity and control (as visualized in the provided flow chart):

- **File Selection and Upload:** The user selects a file and chooses to upload it to either **local or cloud storage**.
- **QR Generation:** The system immediately proceeds to **QR code generation** regardless of the storage choice.
- **Password Protection (Optional):** The user can optionally set a password for **additional protection**. If set, the process continues to password setting; otherwise, it moves to generating the Access Link.
- **Access/Download Phase:** A recipient scanning the QR code triggers an access attempt, where the system first checks for a required password. If a password is required, it must be entered and verified.
- **Access Policy Enforcement:** If the password is correct (or not required), the system evaluates the Access Policy, which dictates the file's ephemeral nature, such as a **Single download limit or a 24-hour Timer**.
- **Link Expiration:** Once the policy condition is met (single download complete or 24-hour timer expires), the link reaches the **Link Expire** state, terminating file access.
- **Access Denial:** Access is denied if the password check fails.

II. System Architecture

The proposed secure file-sharing system is built on a **modular architecture** that manages the security lifecycle of a shared file through three core components: the **Encryption Layer, QR Code Generation, and Time-Bound Access Control**. The system uses a **React** frontend for user interaction and a **Firebase** backend for scalable data management and services.

A. Core Components and Security Layers

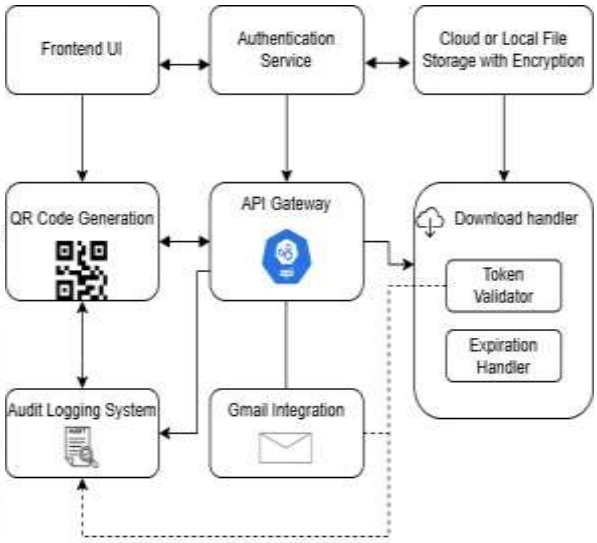
- **Encryption Layer:** This critical layer ensures data confidentiality. It uses the **Advanced Encryption Standard (AES) algorithm** to secure files prior to storage. Users have the flexibility to enable or disable this encryption based on the file's sensitivity.
- **QR Code Generation:** The unique access link or temporary token for the secured file is generated and converted into a **QR code**. This code acts as a portable, temporary access key, replacing long, static links.
- **Time-Bound Access Control:** This essential mechanism employs a **timestamp-based validation** system. It automatically ensures the generated QR code and its associated download link expire after a pre-defined duration, preventing unauthorized or delayed access and automatically revoking authorization.

B. System Flow and Technology Stack

The architectural flow connects several key services:

- **Frontend UI (React):** Provides the intuitive interface for file upload, setting encryption, managing expiry, and displaying the generated QR code.
- **Authentication Service:** Manages user login and identity verification, ensuring only authenticated users can share files.
- **Cloud or Local File Storage:** The repository where encrypted or unencrypted files are stored.
- **API Gateway:** Routes requests and manages link generation and access expiration logic.
- **Download Handler:** The core security enforcement engine. It comprises:
    - **Token Validator:** Verifies the unique token embedded in the QR code.
    - **Expiration Handler:** Checks the link's creation timestamp against the current time to enforce the time-bound policy.
- **Audit Logging System:** Records all critical events, including file uploads, access attempts, and link expirations, offering full traceability.
- **Gmail Integration:** Allows for the secure sharing of the generated QR code or access link directly via email.

The selection of **Firebase** for the backend is a strategic choice, offering high scalability for efficient handling of authentication, link management, and access expiration.



III. Methodology

The system's operation is structured as a clear sequence of steps designed to maintain a high level of security throughout the file-sharing lifecycle.

A. File Upload and Encryption Process

1. **User Upload:** Users initiate the process by uploading files through the application's interface.
2. **Encryption Decision:** The user decides whether to apply the **optional encryption** based on the file's sensitivity.
3. **AES Implementation:** If encryption is chosen, the **AES algorithm** is applied to the data. A unique key is generated at runtime for each specific encryption instance, ensuring that every shared file has a unique encryption key.

B. QR Code Generation and Dissemination

1. **Link Generation:** A unique, temporary access link or token pointing to the securely stored file is generated.
2. **QR Code Encoding:** This unique link is then embedded into a QR code. The implementation utilizes **Python's `qrcode` library** for efficient and reliable generation.
3. **Display and Sharing:** The resulting QR code is either immediately displayed for the user to capture or shared directly with the recipient via integrated, secure channels like the **Gmail integration**.

C. Access Control and Validation

1. **Scanning and Request:** The recipient scans the QR code, which triggers an access request to the system via the embedded link.
2. **Credential Verification:** The system first verifies the recipient's user credentials, if they were required.
3. **Timestamp Validation:** The Download Handler's **Expiration Handler** then rigorously checks the access link's timestamp validity. Access is only granted if the credentials are valid and the link is within its defined time window.

D. Automatic Expiry and Data Deletion

1. **Enforcement:** The **time-bound access control** mechanism strictly enforces the pre-set expiry time (e.g., 24 hours or a single download).
2. **Invalidation:** Once the expiration time is reached, the access link and the associated QR code automatically become **permanently invalid**.
3. **Optional Deletion:** In high-security contexts, files may be **automatically deleted** from the database after link expiration to further mitigate long-term exposure risks.

This systematic methodology ensures that even if a QR code is intercepted, the file cannot be accessed without proper authentication and cannot be reused after its stipulated expiration.

IV. Results and Discussion

The implemented system successfully achieves its primary goal: significantly enhancing data security and user control during digital file sharing.

A. Security Effectiveness

- **Confidentiality:** Testing confirmed that the integration of **AES encryption** is highly effective, preventing unauthorized access even if the access link is exposed to third parties. The strong encryption ensures the data remains unintelligible without the correct decryption key.
- **Access Control:** The **time-bound expiry mechanisms** successfully invalidated old QR codes. This feature directly solves the critical vulnerability of persistent access links, eliminating long-term exposure risks associated with traditional sharing.
- **Traceability:** The **Audit Logging System** provides full traceability of all sharing events, greatly enhancing security governance.

B. Performance and Usability

- **Performance:** Implemented on the scalable **Firebase** platform, the system performed efficiently under moderate user load with minimal latency. This architecture is well-suited for various use cases, from personal to corporate.
- **Trade-offs:** There is a minor trade-off between usability and security, as the extra steps for setting encryption and expiry introduce slight complexity to the user workflow. However, the overall user interface was found to be intuitive, largely due to the visual simplicity and convenience of **QR code-based sharing**.

C. Comparative Advantage

Compared to conventional methods like sharing via unprotected cloud storage links or email attachments, the proposed system offers clear advantages:

- **Higher Confidentiality:** Guaranteed by the optional AES encryption.
- **Enhanced Traceability:** Provided by the Audit Logging System.
- **Limited Exposure:** Ensured by the automatic,time-based link expiration.

The system provides a robust and practical solution that effectively secures sensitive information in a digital environment.

V. Conclusion

This paper has detailed a comprehensive and secure approach to digital file sharing, built on the synergistic integration of **QR code generation, optional AES encryption, and strict time-bound access control**. The system's key features include:

- **Confidentiality:** Ensured by the reliable AES encryption algorithm.
- **Security against Reuse:** Safeguarded by dynamic QR codes and automatic link expiration, which prevents unauthorized access reuse.
- **Scalability:** Achieved through the robust and user-friendly Firebase-React implementation.

The framework successfully addresses the critical security shortcomings of traditional file-sharing methods by ensuring the **ephemeral and controlled distribution of data**.

A. Future Enhancements

Future development will focus on integrating advanced technologies to further improve the system's robustness and trustworthiness:

- **Biometric Authentication:** Incorporating biometric checks for enhanced user identity verification before file access.
- **Blockchain-based Auditing:** Utilizing decentralized ledger technology to provide an **immutable and highly trustworthy record** of all sharing and access events.
- **Location-aware Access Restrictions:** Implementing geographical constraints to ensure files can only be accessed from pre-approved or defined physical locations.

These enhancements will continue to solidify the system's position as a state-of-the-art solution for secure digital data transmission.

References

[1]https://www.researchgate.net/publication/332826493_Secure_QR-Code_Based_Message_Sharing_System_Using_Cryptography_and_Steganography

[2]https://www.researchgate.net/publication/377580981_Secure_Link_Sharing_through_QR_Code_Encryption_Using_Visual_Cryptography

[3]https://www.researchgate.net/publication/352801461_Secure_and_Usable_QR_Codes_for_Healthcare_Systems_The_Case_of_Covid-19_Pandemic

[4]https://www.researchgate.net/publication381332357_Secure_Authentication_via_Encrypted_QR_Code

[5]https://journal.uib.ac.id/index.php/joint/article/view/6090.

[6]https://www.mdpi.com/2073-8994/10/4/95

.[7]https://ijsart.com/cloud-security-framework-for-encrypted-file-sharing-with-qr-authentication-103758.

[8]https://www.irjmets.com/upload_newfiles/irjmets70700086238/paper_file/irjmets70700086238.pdf.

[9] https://www.qrclip.io/

[10]https://scanova.io/blog/password-protected-qr-codes/

[11]https://qrmark.com/blog/document-security-qr-code-guide/

[12]https://www.researchgate.net/publication/339975365_A_Secure_QR_Code_System_for_Sharing_Personal_Confidential_Information

[13]https://www.researchgate.net/publication/263146774_QR_Code_Security_A_Survey_of_Attacks_and_Challenges_for_Usable_Security.