Sampath .M, 2025, 13:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

# Secure Real-Time File Sharing Using Blockchain Technology

**Sampath .M, Adhwaith Anilkumar, Akshay Ajay, Arjun Balagopalan**
Cyber Security
Mahendra Engineering College, Anna University

**Abstract- In the modern digital age, secure and efficient file sharing is paramount. This paper presents a blockchain-based real-time file sharing application that ensures data integrity, confidentiality, and decentralized access. The system is developed using Python and JavaScript, with Flask as the backend framework and AES encryption for data security. Users can sign up, upload files, share them via links, and download them, all while transactions are logged immutably on a blockchain. This application offers a reliable, transparent, and tamper-proof solution for personal and organizational data exchange.**

**Keywords- Blockchain, File Sharing, Data Integrity, AES Encryption, Secure Upload, Decentralized Storage, Flask, Python Security, Real-Time Collaboration.**

## I. INTRODUCTION

In today's hyper-connected digital landscape, the need for secure, efficient, and tamper-proof file sharing mechanisms has become more crucial than ever. Individuals, businesses, and institutions across sectors exchange sensitive information daily, making them vulnerable to cyber threats, unauthorized access, and data manipulation. Traditional file sharing systems, predominantly centralized in nature, suffer from inherent limitations including single points of failure, data breaches, lack of transparency, and limited traceability of file operations.

To address these challenges, our project introduces a Blockchain-Based File Sharing Application, designed to offer a decentralized, immutable, and secure platform for file exchange. This system combines blockchain technology, AES encryption, and real-time web interfaces to ensure that users can confidently store, share, and retrieve files while maintaining full control over data provenance and access.

The inception of this project is driven by growing global awareness around digital security and privacy. While conventional cloud storage services offer convenience, they require users to place their trust in third-party providers. This trust model has proven fragile in the wake of high-profile data leaks and cyberattacks. Blockchain, a distributed ledger technology, disrupts this paradigm by introducing a trustless environment—where data integrity and operation history are guaranteed through cryptographic principles and decentralized consensus mechanisms.

Each action in the application—such as uploading, downloading, sharing, or deleting a file—is recorded as a new block in the blockchain. This creates an immutable audit trail that not only deters unauthorized tampering but also enhances transparency across the system. The blockchain ledger is stored locally and updated in real time, allowing users to visualize and verify file operations directly from their dashboards.

To further safeguard data, the application integrates Advanced Encryption Standard (AES) encryption for all files stored on the server. Files are encrypted during upload and decrypted only upon

download by authorized users, ensuring that data remains unintelligible to unauthorized parties even if access to the storage system is compromised.
User authentication and session management are handled through secure, token-based mechanisms using JSON Web Tokens (JWT). This ensures that only authenticated users can perform file operations, while session information is securely maintained without persistent reliance on cookies or server-side sessions.

From a usability perspective, the application features an intuitive and responsive web interface developed using HTML, CSS, and JavaScript. Users can register, log in, upload files, share them with other users, manage incoming file requests, and view a live log of blockchain entries—providing a seamless and interactive user experience.

The system architecture adheres to modern web development standards, leveraging Flask for backend API routing, SQLite for persistent data storage, and a modular MVC-like structure to ensure maintainability and scalability. Furthermore, file sharing between users is facilitated through a request-approval workflow that mimics real-world access control systems, where users can request files, and owners retain full discretion over approvals.

This application is not only a technical demonstration of blockchain's applicability beyond cryptocurrencies but also a practical tool that can be adopted by organizations and individuals seeking enhanced file security, transparency, and user-centric data control. Future iterations may extend functionality by integrating IPFS for distributed file storage, enabling public/private key cryptography for advanced access management, or deploying on decentralized networks for full infrastructure decentralization.

The Blockchain File Sharing Application follows a modular and scalable architecture, drawing on the principles of the Model-View-Controller (MVC) pattern to separate concerns and promote maintainability.

**Backend Framework**

The backend is built using Flask, a lightweight yet powerful Python web framework. Flask handles routing, API endpoints, session management, file operations, blockchain logic, and database interactions. It is structured to serve both HTML templates and RESTful JSON responses, enabling dynamic, real-time user interaction.

**Blockchain Engine**

**At the heart of the application lies a custom-built blockchain ledger, implemented in Python. Each file-related action (upload, download, share, delete) results in the creation of a new block. A block contains:**

- Index: Position in the chain
- Timestamp: Exact time of action
- Data: Description of the action (e.g., "user A shared file.txt with user B")
- Previous Hash: Ensures chain linkage
- Current Hash: Ensures data integrity

This blockchain is not a consensus-based network like Bitcoin but rather a private ledger, tailored for transparency and traceability within the application.

**Encryption Layer**

To ensure file confidentiality, the application uses AES (Advanced Encryption Standard) to encrypt files before saving them on the server. Upon download, files are decrypted on-the-fly and sent to the authorized user. This approach ensures that file contents are secure even in the event of storage compromise.

**Authentication and Authorization**

The application uses JWT (JSON Web Tokens) to manage user sessions securely. Upon successful login, a token is issued to the user, which is stored in the session. This token is used to verify the identity of users for all subsequent actions without exposing credentials.

**Database Layer**

The application uses SQLite (or optionally MySQL) for persistent data storage. The schema includes:

- **Users:** Stores login credentials and profile info.
- **Files:** Metadata about uploaded files, including encrypted names and ownership.

- **ShareRequests:** Pending or accepted/rejected file sharing requests between users.
- **Blockchain (in-memory):** Reconstructed at runtime or optionally persisted for audit.

**Frontend Interface**

**The user interface is built with HTML, CSS, and JavaScript. It features:**

- A login/signup portal
- A dashboard to upload/download/delete files
- File sharing forms
- A real-time blockchain visualization panel, which fetches the live blockchain data and displays it in block-by-block cards

  The frontend also includes JavaScript code to asynchronously fetch blockchain data and update the UI in real time, ensuring a responsive experience.

## II. WORKING PROCESS

**Working Process of the Blockchain-Based File Sharing Application**

The Blockchain-Based File Sharing Application is designed to provide a secure, transparent, and tamper-proof platform for uploading, downloading, sharing, and auditing files. The system integrates blockchain technology with encryption, access control, and a real-time web interface to enable users to exchange files while maintaining privacy and trust. This section outlines the functional components of the system and describes how each element contributes to the secure handling and tracking of file operations.

Secure User Authentication and Session Handling

The working process begins with secure user authentication. The application supports a signup/login system built using Flask and SQLite. During signup, user credentials are securely hashed using industry-standard algorithms (e.g., SHA256 or bcrypt), ensuring that passwords are never stored in plaintext. Upon successful login, the user receives a JSON Web Token (JWT), which is stored in a session. JWT is used to validate each user request during the session without requiring repeated credential entry, thereby maintaining both security and usability.

**File Upload with AES Encryption**

**Once logged in, users are directed to the dashboard where they can upload files. When a user selects a file to upload, the system performs the following steps:**

- The file is read into memory and encrypted using Advanced Encryption Standard (AES).
- A unique AES key is generated per session or per user, depending on configuration.
- The encrypted file is saved to the server's file sytem.
- Metadata including the original filename, encrypted filename, owner, and timestamp is stored in the database.
- This ensures that even if a file is accessed outside the app environment, its contents are unreadable without decryption.

**Simultaneously, the system generates a new block in the blockchain to record the upload event. Each block includes:**

Action (e.g., "Upload")

- Filename
- Username
- Timestamp
- Previous block’s hash
- Current block’s hash

The blockchain acts as an immutable log, ensuring that file operations are fully auditable and tamper-proof.

**Blockchain Construction and Logging**

Each block added to the blockchain is cryptographically linked to the previous block. The application maintains a dynamic in-memory blockchain data structure, initialized with a genesis block. Whenever a file is uploaded, downloaded, shared, or deleted, a new block is created. Each block stores:

- The transaction type (e.g., upload, share, download)
- The involved users
- The affected file
- A precise UNIX timestamp
- A hash value generated using SHA-256

This linked list of hashes forms a tamper-resistant chain of custody, making it impossible to alter any record without invalidating the entire chain.

Users can view the full blockchain ledger in their dashboard through a live-rendered block explorer, implemented in JavaScript. Each block is displayed in a scrollable container, providing real-time visibility into actions performed on the system.

**File Sharing and Access Control Workflow**

**A key feature of the application is user-to-user file sharing. The sharing mechanism includes a request-approval system:**

- **Request:** A user selects a file they own and enters the recipient's username. A sharing request is sent and stored in the database as "pending".
- **Approval:** The recipient is notified of incoming requests and can choose to accept or reject.
- **Transfer:** Upon approval, the file metadata is cloned to the recipient's account. The actual encrypted file remains on the server, reducing duplication.
- **Logging:** Every step—request creation, approval, and final delivery—is logged in the blockchain.
- This process ensures that all file-sharing actions are deliberate and authorized, maintaining both traceability and consent-based access.
- File Download and On-the-Fly Decryption
  When a user decides to download a file:
- The system verifies ownership or sharing permission.
- It locates the encrypted file and decrypts it on the server using the user's AES key.
  The decrypted file is then streamed to the user's browser for download.
- This method avoids persistent decrypted file storage, reducing attack surfaces and improving data security.
- Every successful download event is recorded as a new block in the blockchain, capturing the username, file name, and exact timestamp. This provides a reliable digital trail for post-activity audits.

**File Deletion and Audit Trail**

Authorized users can also delete their uploaded or received files. Upon deletion:

- The file is removed from the storage directory.
- The associated metadata entry is deleted from the database.

**A new block is added to the blockchain to log the deletion event.**

This log ensures that no action goes unrecorded, helping administrators and users maintain a complete audit trail of their activities.

**Real-Time Blockchain Visualization**

To improve user transparency, the dashboard features a real-time blockchain viewer, implemented with JavaScript and Flask. The viewer fetches the current blockchain state via the /blockchain route and renders each block dynamically in the browser. Each block card includes:

- Block Index
- Timestamp
- Action Summary
- Hash and Previous Hash
- User involved

This live blockchain interface ensures that users can verify their actions and observe the complete operational history of the application.

**Secure Data Handling and Privacy Measures**

In addition to encryption and authentication, the system implements several data privacy and security measures, including.

- Secure session cookies
- Server-side validation of all user actions
- Input sanitation to prevent injection attacks
- Role-based access control to enforce ownership and sharing permissions

These practices help ensure that the platform is resilient to common web vulnerabilities while preserving user trust and confidentiality.

**Summary**

**The working process of the blockchain file sharing system is a coordinated workflow that integrates encryption, decentralized logging, and access control. From file upload to sharing, download, and deletion, each action is tightly monitored and recorded. The system ensures that:**

- Files are secure (via AES)
- Actions are traceable (via blockchain)
- Permissions are enforced (via request-based sharing)
- Users are protected (via JWT and access control)

- Together, these components form a robust, user-centric platform that redefines how secure file sharing can be achieved in a decentralized and accountable manner.
-

## III. RESULTS AND DISCUSSION

This section evaluates the outcomes and insights gained from developing and testing the Blockchain-Based File Sharing Application. The application demonstrates how emerging technologies such as blockchain and encryption can be harnessed through lightweight, open-source frameworks like Flask and PyCryptodome to deliver a secure and transparent file exchange system. Through a structured and feature-complete implementation, the application provides real-time traceability, user-controlled access, and cryptographic assurance in handling sensitive files.

- **User Authentication and Dashboard**
  The system incorporates a robust user authentication mechanism using hashed passwords and session-based management. The dashboard acts as the user's central interface, presenting options to upload, share, delete, and download files.
- **Highlights:**
- Successful authentication results in a session initiation that persists securely via Flask session cookies.
- Access to the dashboard and all file-related functions is gated by user authentication.
  Incorrect login attempts are gracefully handled with minimal data leakage, maintaining security while guiding users with helpful feedback.
- This login framework proved stable across multiple user sessions and ensured unauthorized users could not access protected routes.
- **File Upload and Encryption Workflow**
  One of the core security features of the system is the AES encryption of files during upload. Each uploaded file undergoes encryption using a secure AES key before being stored on the server.
- **Outcomes:**
- Files stored on the backend were confirmed to be unreadable in raw form, validating encryption integrity.
- During download, files were correctly decrypted on the fly and matched the original files byte-for-byte.
- Uploads of varying file types and sizes (e.g., .txt, .png, .pdf, up to 10MB) were supported without performance degradation.
- This process demonstrated strong data confidentiality and established a foundation for secure file storage.
- **Blockchain Logging and Integrity Verification**
- **Every user action—whether uploading, downloading, sharing, or deleting a file—is recorded as a block in the application's internal blockchain. The blockchain structure ensures that each block contains:**
- Index
- Timestamp
- User action data
- Previous hash
- Current hash (SHA-256)
- **Observations:**
- Blockchain entries were created in real-time and immediately visible in the UI.
- Hash values correctly linked each block, preserving chain continuity.
- Tampering with block data broke the chain, validating immutability.
- This provided a tamper-proof audit trail, allowing any user or administrator to verify historical actions with complete transparency.
- **File Sharing and Request Management**
  The application implements a request-based sharing mechanism. Users can initiate file sharing with others, which creates a pending share request. The recipient can choose to accept or reject the request.
- **Results:**
- Approved requests correctly cloned file references under the recipient's account without duplicating physical storage.
- Blockchain entries logged all key events: request sent, accepted, and file downloaded.
- Rejected requests were cleanly removed and did not grant access.

- This workflow enhanced user trust, ensured full consent in file access, and allowed secure peer-to-peer interactions within the platform.

- **Blockchain Visualization**
  To improve transparency and usability, the blockchain ledger is rendered visually within the dashboard. The visualization dynamically loads via JavaScript and displays block attributes such as timestamps, actions, and hashes.
- **Key Insights:**
- The viewer updated in real-time as new blocks were added.
- Layout remained clean and readable even as the chain grew to 30+ blocks.
- Each block was styled with unique borders, colors, and indentation to simulate a true ledger view.
- This component reinforced transparency and usability by showing users a clear, real-time record of actions in the system.
- **Summary of Core Results**
  Module Outcome
  Authentication Stable, secure login/logout using Flask sessions
  AES File Encryption Effective at protecting file contents at rest and in transit
  Blockchain Logging Tamper-proof logs for all file operations
  File Sharing System Controlled request/approval flow with full audit trail
  Blockchain UI Clean, real-time blockchain display for end-users
  Error Handling All operations returned appropriate messages on failure or misuse
- **Practical Benefits**
- **Increased Data Integrity:** Blockchain logs ensure no user action is lost or altered, promoting full accountability.
- **Improved Security:** AES encryption and role-based access control protect files from unauthorized viewing.
- **Decentralized Audit Trail:** The blockchain replaces traditional server-side logs with a cryptographically verifiable chain of events.
- **User Control:** File owners retain complete authority over who accesses their files and under what conditions.
- **Educational Value:** The app serves as a working prototype to understand real-world use of blockchain outside of cryptocurrency.
- **Limitations**
- **Despite its strengths, the current implementation has some known limitations:**
- No Distributed Storage: Files are stored on a centralized server; integrating IPFS or cloud storage would enhance decentralization.
- Blockchain Persistence: The blockchain exists only in memory; server restarts reset the chain unless manually backed up.
- AES Key Management: Key distribution is implicit and not currently hardened with public-private key encryption.
- Lack of Email Notifications: Users must manually check for file requests rather than receiving alerts.

- **Opportunities for Enhancement**
- **To evolve the platform into a production-grade tool, the following improvements are recommended:**
- Integrate IPFS for true decentralized file storage.
- Persist Blockchain to Disk or use a lightweight database such as LevelDB for chain storage.
- Public-Key Encryption for key distribution and file-specific access rights.
- Email or SMS Notifications for share requests and activity updates.
- Admin Dashboard to monitor system-wide blockchain logs, user activity, and performance metrics.
- Mobile Responsiveness to support file operations on smaller devices.

## IV. CONCLUSIONS

The Blockchain-Based File Sharing Application offers a powerful solution for enhancing both the security and transparency of digital file exchanges. By leveraging blockchain technology, the system creates an immutable, tamper-proof record of all

file transactions—including uploads, downloads, shares, and deletions—ensuring that every user action is permanently logged and verifiable. This immutable audit trail significantly reduces the risk of data manipulation, providing users with a high level of confidence in the integrity of the system.

In parallel, the application enhances confidentiality and access control through the use of AES encryption and a secure user authentication system. Files are encrypted upon upload and only decrypted during authorized downloads, safeguarding them from unauthorized access even in the event of server compromise. Access to files is strictly governed by ownership and sharing permissions, which are enforced through a request-and-approval model. This model not only ensures that files are shared intentionally, but also provides an opportunity for user-to-user collaboration while maintaining privacy.

The system's web-based dashboard offers a real-time interface for managing file activities and visualizing the blockchain ledger. By incorporating interactive components and intuitive UI elements, the platform ensures that even non-technical users can navigate the system with ease. The inclusion of a real-time blockchain explorer further empowers users to verify their own activity histories and hold others accountable for shared interactions.

Together, these components form a comprehensive and modern solution for secure file sharing—one that replaces the opacity and limitations of centralized systems with transparency, cryptographic security, and user empowerment. The application's modular and open-source architecture makes it highly customizable and suitable for deployment in academic, enterprise, or personal environments where data security is paramount.

This project demonstrates that blockchain is not limited to cryptocurrencies—it can serve as a critical infrastructure component for securing digital assets in collaborative environments. By combining blockchain with encryption, access control, and a user-friendly interface, the system establishes a practical foundation for trustworthy digital communication.

## REFERENCES


1. S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.
2. W. Stallings, Cryptography and Network Security: Principles and Practice, 7th ed., Pearson Education, 2017. ISBN: 9780134444284
3. M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain Technology: Beyond Bitcoin," Applied Innovation Review, no. 2, pp. 6–10, Jun. 2016.
4. A. Gervais, G. Karame, V. Capkun, and S. Capkun, "Is Bitcoin a Decentralized Currency?" IEEE Security & Privacy, vol. 12, no. 3, pp. 54–60, May–Jun. 2014. doi: 10.1109/MSP.2014.22
5. S. M. Khan and K. Salah, "IoT Security: Review, Blockchain Solutions, and Open Challenges," Future Generation Computer Systems, vol. 82, pp. 395–411, May 2018. doi: 10.1016/j.future.2017.11.022
6. K. Christidis and M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," IEEE Access, vol. 4, pp. 2292–2303, 2016. doi: 10.1109/ACCESS.2016.2566339
7. R. Jalali, K. El-Khatib, and C. McGregor, "Smart Contract Implementation Using Blockchain Technology in a Secure File Sharing Model," Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 2509–2516. doi: 10.1109/BigData47090.2019.9006015
8. F. Zhang, A. Miller, and E. Shi, "Programming and Security of Distributed Ledgers," Lecture Notes in Computer Science, vol. 11475, pp. 67–91, Springer, 2019. doi: 10.1007/978-3-030-17653-2_3
9. J. Katz and Y. Lindell, Introduction to Modern Cryptography, 2nd ed., CRC Press, 2014. ISBN: 9781466570269
10. P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," Advances in Cryptology — CRYPTO' 99, Lecture Notes in Computer Science, vol. 1666, pp. 388–397, Springer, 1999. doi: 10.1007/3-540-48405-1_25