DOCUMENT ACCESS CONTROL USING BLOCKCHAIN

Prof. Bhagat Inamdar
Dept. of CSE
KLS VDIT, Haliyal

Dr. Venkatesh Shankar
Dept. of CSE
KLS VDIT, Haliyal

Mr. Deepak Badiger
Dept. of CSE
KLS VDIT, Haliyal

Mr. Elisha C Uppalapati
Dept. of CSE
KLS VDIT, Haliyal

Mr. Shivanand Malkapuri
Dept. of CSE
KLS VDIT, Haliyal

Mr. Prasann Daddikar
Dept. of CSE
KLS VDIT, Haliyal

Abstract:- In the era of digital transformation, securing access to sensitive document is essential. Traditional systems are centralized, exposing them to failures are centralized access control system built on blockchain, utilizing smart contracts for secure, transparent and tamper- proof document management. The system enables secure authentication, role-based permissions, and immutable access logs. Evaluations enhanced trust and security in sectors such as healthcare, education and enterprise IT.

Keywords: Blockchain, Access Control, Smart Contracts, Documents Security, Decentralized System, RBAC, Immutable Ledger, Data Privacy.

I. Introduction

As organization increasingly rely on digital documents for daily operation and strategic decision- making, securing access to these resources becomes critical. Traditional access control mechanisms are like Access Control Lists (ALCs) and Role-Based Access Control (RBAC) are often centralized, making them susceptible to single point of failure, unauthorized access and administrative inefficiencies.

This proposes a decentralized document access control system leveraging blockchain technology. By utilizing blockchain's immutable and transparent ledger along with programmable smart contracts, the system ensures authentication, fine-grained policy enforcement, and verifiable access history without the need for central authorities.

II. System Architecture

The proposed system employs a modular, blockchain based architecture to deliver secure, decentralized and auditable document access control. It is composed of three core layers:

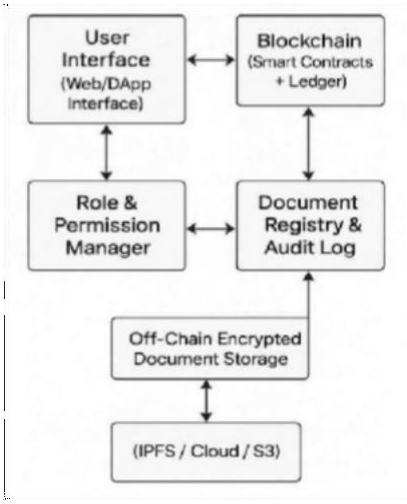

Fig. 1. System Architecture

Key Components:

- **Authentication Module:** Verifies users via public-key cryptography using Ethereum wallet addresses.
- **Role Manage:** Enable admins to assign user roles (e.g. admin, editor, viewer), defining their access rights.
- **Access Control Smart Contract:** Encodes and enforces document access rules, logs each access request immutably.
- **Document Registry:** Stores metadata (e.g. document hash, owner, access list) on-chain while keeping contents off-chain.
- **Audit Log:** Maintains a tamper-proof log of access events for compliances and forensics.
- **Off-Chain Storage:** Uses IPFS to store encryption documents, decryption keys are managed via smart contracts.

Workflow Summary

- **Document Upload:** Owner uploads a document, its hash is stored on-chain, while the file is encrypted and stored off-chain.
- **Granting Access:** Access rights are granted by assigning roles via the interface and smart contracts update permissions.
- **Access Request:** A user requests access through a smart contract, which verifies role and returns access credentials.
- **Access Logging:** Every access attempt is recorded on-chain with timestamp and user ID.

III. Technical Aspects

A. Blockchain Technology

Blockchain is a distributed ledger that maintains secure, transparent and immutable records through cryptographic hashing and consensus mechanisms (e.g. Proof of Work, Proof of Stack). While public blockchains offer transparency, private and consortium blockchain provide better control and performance provide better control and performance making of them more suitable for enterprise document management. In this system, blockchain is used to manage access policies and logs, not to store documents themselves, ensuring that no entity can alter records unilaterally.

B. Smart Contracts

Smart contracts are self-executing programs written in Solidity and deployed on Ethereum- compatible blockchains. They automate access control operations, including permission checks and activity logging, without relying on centralized authorities. Each contract defines access rules, enforces them in real time and logs every request immutably.

C. Role-Based Access Control (RBAC)

RBAC simplifies permission management by assigning roles (e.g. admin, editor and viewer) instead of individual access rights. These roles are linked to permissions within the smart contract, allowing flexible and scalable policy management. Changing access is as simple as reassigning roles, reducing administrative overhead.

D. Blockchain Network Types

- **Public:** Open and transparent, but less efficient.
- **Private:** Restricted to trusted nodes, offers better performances.
- **Consortium:** Controlled by a group organization, balances trust and efficiency.

This system prefers private or consortium blockchain to optimize for privacy, scalability, and control in enterprise use cases.

E. Cryptographic Techniques

Security is enforced through:

- Public key Cryptography is uses for authentication and digital signatures.
- SHA-256 Hashing for data integrity.
- AES Encryption for off-chain document confidentiality.

IV. Implementation

The implementation focused on integrating smart contracts, cryptographic security, off-chain storage and a user-friendly interface.

A. Technology Stack

- Blockchain: Ethereum (via local hardhat or Sepolia Mainet Testnet).
- Smart Contracts: Written in Solidity
- Frontend: React.js with Web3.js for blockchain interaction.
- Wallet Integration: Metadata for user authentication.
- Off-Chain Storage: Iner Planetary File System (IPFS).
- Encryption: AES (Advanced Encryption Standard) for file confidentiality.

B. Smart Contracts

Smart contracts serve as the core logic for access control and auditing. Key functionalities include:

- **Role Management:** Assigning and revoking user roles (admin, editor, viewer).
- **Access Control:** Granting, revoking and verifying access to documents based on roles.
- **Audit Logging:** Logging each access

attempt immutably using blockchain events. Contracts were deployed on a local Ethereum network but are compatible with testnets like
Ethereum Sepolia or the Hardhat.

C. Document Handling and IPFS Integration

- **Pre-Upload:** Files are hashed (SHA-256) and encrypted (AES).
- **Upload:** Encrypted files are stored on IPFS, their CID and hash are saved in the smart contract.
- **Access Retrieval:** AES keys are securely shared off-chain, users fetch and decrypt files locally.

D. User Interface

The web interface provides:

- Metadata based login.
- Document upload and metadata registration.
- Role assignment and permission management.
- Blockchain-backed audit log viewer.
- Real-time contract interaction using Web3.js.

E. Example Workflow

1. Admin assigns a role (e.g. Editor) to a user.
2. Document owner uploads an encrypted file and registers its metadata.
3. Owner grants access via the smart contract.
4. The assigned user connects via Metadate and accesses the files.
5. Each access event is immutably logged on the blockchain.

F. Testing and Debugging

➢ Metadata was configured to connect to the local blockchain for real-world testing via the browser.

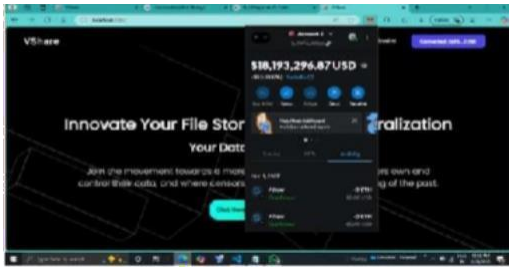

Fig. 2. Connection with Metadata

Fig. 3. Transaction Fees for Uploading

Fig. 4. File Uploaded Successfully

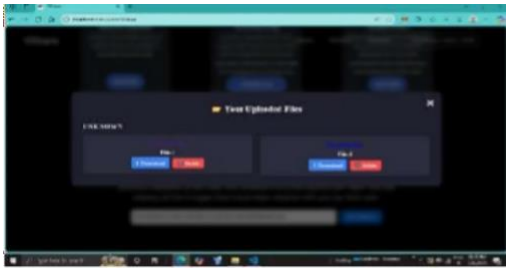

Fig.5. User Uploaded Files

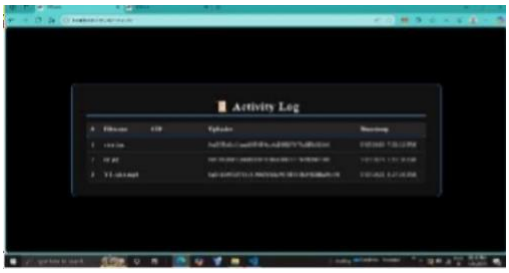

Fig. 6. Activity Log

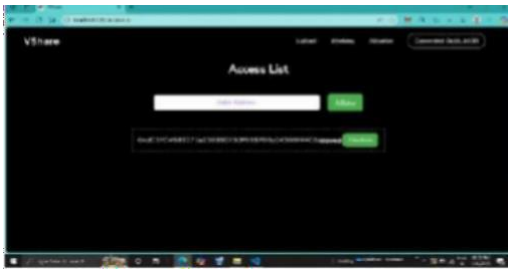

Fig.7. Allow user access

Fig. 8. File sharing transaction

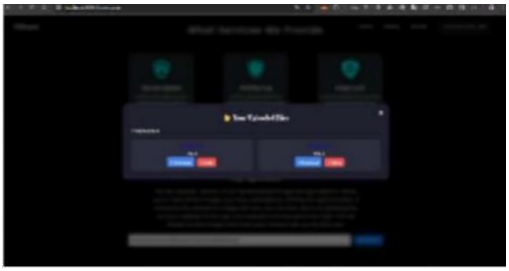

Fig. 9. File accessed by Receiver

V. Use cases

A. Corporate Document Sharing

A manager uploads encrypted financial reports to the system. Team leads, assigned the role of "Financial Reviewer" access the reports via smart contract-granted permissions. All access is logged immutably, ensuring auditability and limiting visibility to authorized personnel.

B. Academic Transcript Verification

Universities upload encrypted transcripts to IPFS and store access metadata on-chain. Students control access and share transcripts with employers or institutions. The blockchain provides transparent access logs, ensuring authenticity and student ownership of academic records.

C. Healthcare Records Access

Hospitals encrypt and upload medical reports, giving patients full control. Patients can temporarily share access (e.g. for 72 hours) with doctors. Smart contracts enforce time-bound access and all access attempts are logged on-chain, ensuring secure and controlled medical data sharing.

VI. Results & Evaluation

The proposed blockchain based document access control system was evaluated on functionality, security, scalability and usability with comparisons to traditional models such as ACL and RBAC.

A. Functional Testing

Core system features were validated on a local Ethereum testnet and IPFS including:

- User authentication via Metadata.
- Role assignment and revocation.
- Document upload and encryption.
- IPFS integration for off-chain storage
- Access granting/revoking via smart contracts.
- Immutable access logging.
- Compared to traditional ACL/RBAC systems, the blockchain solution is decentralized provides immutable audit trails, enforces role-based permissions via smart contracts and demonstrates high tamper resistance and trustlessness.

B. Performance Discussion

- **Latency:**
  - RBAC (~30 ms): Fastest model, checks access by user roles. Simple but less flexible.
  - ABAC (~50 ms): Moderate latency, uses multiple attributes for dynamic access control.
  - Proposed Blockchain Model (~120 ms): Highest latency due to smart contracts, cryptographic

checks and consensus. It offers better security and transparency.

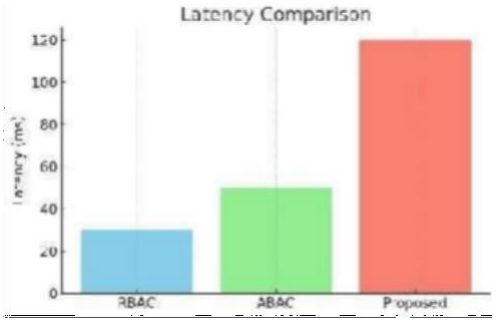

Fig. 10. Latency comparison

- **Throughput:**
  - RBAC:
    - Starting at100 req/sec and decreasing 80 req/sec (as load increases.
    - Highest throughput due to simple role check.
  - ABAC:
    - Starting at 90 req/sec and decreasing 50 req/sec
    - More complex logic causes sharper decline under load.
  - Proposed Blockchain model:
    - Begin with 70 req/sec and dropping to 45 req/sec.
    - Lower throughput due to smart contract execution, transaction validation and cryptographic overheads.

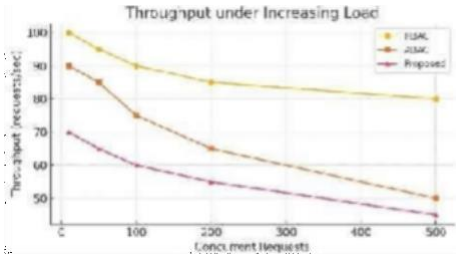

Fig. 11. Throughput Comparison

- **Security Effectiveness**
  - Unauthorized Prevention
    - RBAC (3/5): Basic role checks, moderate protection.
    - ABAC (4/5): More precise with attributes, better control.
    - Proposed (5/5): Strongest security via smart contracts and cryptography.
  - Dynamic Policy Update
    - RBAC (2/5): Static, changes require manual updates.
    - ABAC (5/5): Highly flexible with real-time attribute checks.
    - Proposed (5/5): Smart contracts support automatic, rule-based updates.
  - Auditability
    - RBAC (2/5) and ABAC (1/5):

Weak logs, prone to tampering.

- Proposed (5/5): Immutable blockchain logs ensure full transparency.

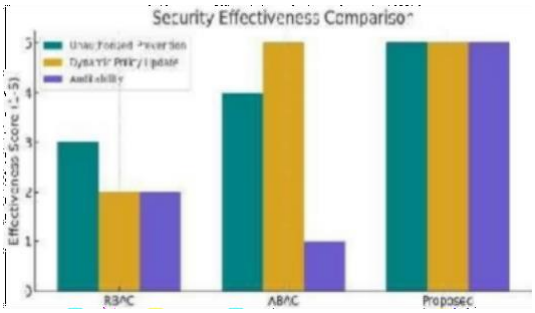

Fig. 12. Security Effectiveness Comparison

VII. Conclusion and Future Work

This work proposed a blockchain based document access control system that enhances security, transparency and trust through smart contracts and immutable audit logs. It overcomes the weaknesses of traditional centralized models by providing tamper-resistant and verifiable access management.

Future work include:

- Improving scalability using Layer 2 solutions or sharding.
- Enhancing privacy with zero-knowledge proofs and other cryptographic methods.
- Simplifying the user experience for non- technical user.
- Enabling interoperability across multiple blockchain platforms.

References:

[1]. "Mastering Blockchain: Unlocking the power of Cryptocurrencies, Smart Contracts and Decentralized Application" by Imran Bashir.
[2]."Blockchain Basic: A Non-Technical Introduction in 25 steps" by Daniel Drescher.
[3]. "Smart Contracts: The Blockchain Technology that will replace Smart Lawyers" by V.S.C.R. Srinivasan.
[4]. "Blockchain-based Access Control for Personal Health Records" by Zhiqiang Wei, Yue Li and Zhiwei Luo (IEEE Access).
[5]. "A Protocol for Economically Sustainable Information Performance" by Sam Williams, Viktor Diordiiev, Lev Berman, India Raybould, Ivan Uemlianin.
[6]. "An Efficient Blockchain-based Framework for File Sharing" by Wanzong Peng, Tongliang Lu, Wenju Peng and Zhongpan Wang.
[7]. "Secured Decentralized Storage system using Blockchain and IPFS" by K. Shruthi, Pandita Meghana, C. Bhanu Prakash, Y. Rushikesh Reddy, Jadhav Meghana.
[8]. "Filecoin: A Decentralized Storage Network":
Protocol labs-https://filecoin.io/filecoin.pdf
[9]. Solidity documentation: https://docs.soliditylang.org/
[10]. Web3.js: https://web3.js.readthedocs.io/
[11]. Metadata: https://metadata.io/
[12]. IPFS: https://ipfs.io