NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE-575025

THE INSTITUTION OF ENGINEERS, NITK CHAPTER
Institutional Member ID: **IM0001491**
https://ie.nitk.ac.in

IE CODE

Project Title: Privara: Decentralized File Storage and Sharing Application

---

Team Number: Team 8

Problem statement: Traditional file storage and sharing methods rely on centralized servers, making them vulnerable to data breaches, unauthorized access, and single points of failure. Users often face privacy concerns and lack control over their data, while service providers can impose restrictions and censorship. Additionally, secure file sharing is often complex and inefficient**.**

Proposed solution:

Privara leverages **blockchain technology** and **decentralized storage** to provide a highly secure and privacy-focused file storage and sharing system. The solution ensures that user data remains encrypted, tamper-proof, and accessible only to authorized users.

- **Decentralized File Storage with IPFS:** Files are encrypted using AES-GCM before being stored on the **InterPlanetary File System (IPFS)**, ensuring data integrity and availability without relying on a centralized entity. IPFS provides a distributed, content-addressable storage mechanism, making files resistant to censorship and single points of failure.

- **Smart Contract-based Metadata & Access Control:** A **Solidity smart contract** deployed on the blockchain manages file metadata, including the **Content Identifier (CID)**, file ownership details, and access permissions. The smart contract ensures:
    - **Tamper-proof metadata storage**, preventing unauthorized modifications.
    - **Secure access control**, ensuring that only intended recipients can decrypt and retrieve shared files.
    - **Decentralized enforcement of security policies**, eliminating the need for a central authority.

- **End-to-End Encryption for Secure File Sharing:**
    - The AES encryption key used to secure each file is encrypted with the **user's public key** via **NaCl cryptographic encryption**, ensuring confidentiality.
    - When a file is shared, the AES key is re-encrypted with the recipient's **public key**, ensuring that only the intended recipient can decrypt the file.
    - The smart contract records access permissions and ensures that only authorized users can retrieve encrypted AES keys.

- **Decentralized & Trustless File Access:** Users decrypt their own uploaded files using their **private key**. For shared files, the recipient retrieves the encrypted AES key from the smart contract, decrypts it using their private key, and then decrypts the file. This trustless mechanism ensures data security without reliance on third parties.

Technologies used:

- **Frontend**: React.js, Web3.js
- **Backend**: Solidity smart contracts deployed on Ethereum-compatible networks
- **Encryption**: AES-GCM for file encryption, NaCl for key encryption
- **Storage**: IPFS for decentralized file storage
- **Wallet Integration**: MetaMask for authentication and transaction signing
- **Smart Contract Execution**: Ethereum Virtual Machine (EVM)

Methodology:

1. **User Authentication**:
    - Users connect their Ethereum wallet and set a passphrase for securing their private key.
2. **File Upload & Encryption**:
    - The user selects a file, which is then encrypted using AES-GCM.
    - A unique AES key is generated, encrypted using the user's public key, and stored in the smart contract.
    - The encrypted file is uploaded to IPFS, and the CID is recorded on-chain.
3. **File Sharing**:
    - The sender selects a file and provides the recipient's public key.
    - The AES key is decrypted using the sender's private key, then re-encrypted using the recipient's public key.
    - The smart contract updates access permissions, ensuring only the recipient can decrypt the AES key and access the file.
4. **File Decryption & Retrieval**:
    - The recipient fetches the encrypted AES key from the smart contract.

- ○ The AES key is decrypted using their private key.
- ○ The encrypted file is retrieved from IPFS and decrypted using the AES key.

Results:

- Successfully implemented decentralized file storage with AES encryption for enhanced security.
- Secure file sharing with NaCl encryption, ensuring only authorized users can access shared data.
- Immutable metadata storage on blockchain, ensuring transparent and tamper-proof access control.
- Efficient and scalable storage using IPFS, reducing reliance on centralized servers.

Future work:

- Implement a decentralized identity system (DID) for enhanced user authentication.
- Introduce access revocation and multi-user sharing functionality.
- Explore zk-SNARKs for privacy-preserving file access verification.
- Develop mobile support for broader accessibility.

Key Learnings:

- Effective use of cryptographic techniques for decentralized file security.
- Smart contract-based access control mechanisms enhance security and eliminate reliance on third parties.

- Challenges in optimizing performance while ensuring security in decentralized applications.

References:

- IPFS Documentation: https://docs.ipfs.io
- Ethereum Solidity Documentation: https://soliditylang.org
- NaCl Cryptography Library: https://nacl.cr.yp.to/
- Youtube : Learn Blockchain, Solidity, and Full Stack Web3 Development with JavaScript – 32-Hour Course

---

Team Members:

M LAKSHYA - 231IT037
MITHUN PATIL V N - 231CS234
PRAVEEN YADAV- 231CS243
