METHODOLOGY FORM

1. Team No.: 1

2. Project Title: Secure File Sharing System using Blockchain

3. Proposed Method:

In a group of organizations, several organizations can share data in the form of files and synergies with their operations. A blockchain network created among multiple organizations, each of the organizations will host an Identity and Interfacing Server (IIS), Smart contract, and blockchain ledger. IIS maintains the identity details in the identity database and is also the interfacing point with the smart contract. A smart contract is a program, which contains the business logic of the proposed file-sharing mechanism and is installed on each of the organizations. The blockchain ledger maintains transactions in the form of blocks.

To preserve the privacy of traceable encryption in blockchain, previous works proposed a system in which authenticity and non-repudiation of digital content are guaranteed. The problem tackled by the authors is the secret key of the user, which when shared with other entities does not hold the specific information of the user. In case the shared key is corrupted or abused, it makes it difficult to analyze the source of the secret key. Moreover, leakage of confidential information in access control is a bottleneck for existing systems. Therefore, previous works have integrated the privacy protection algorithm such as attribute-based encryption (ABE) to secure the secret keys.

The proposed method provides secure file-sharing across a group of organizations using blockchain. It provides confidentiality, integrity, and availability of shared files. It ensures end-to-end encryption of the files. The content ID of the shared file is stored on the blockchain in a tamper-resistant way. The encrypted file and file metadata are stored in a distributed fashion on the distributed storage and blockchain ledger respectively.

4. Proposed Method Illustration:

This method combines blockchain technology, distributed file storage, encryption mechanisms, and access control to create a secure, transparent, and efficient file-sharing system for a group of organizations. Each step contributes to the overall goal of providing a trustworthy and collaborative environment for inter-organizational file sharing.

a. Blockchain Integration:

- Establish a blockchain network using Hyperledger Fabric, configuring nodes for participating organizations within the.

- Develop smart contracts using Hyperledger Fabric to define the rules and conditions for secure file-sharing transactions within the blockchain network.

- Initiate a blockchain transaction to record the file-sharing event, including relevant metadata, on the distributed ledger, providing a tamper-resistant record of the transaction.

- Leverage the transparency of the blockchain to allow all members to trace and verify the file-sharing transactions, enhancing overall trust and accountability.

b. Cryptography:

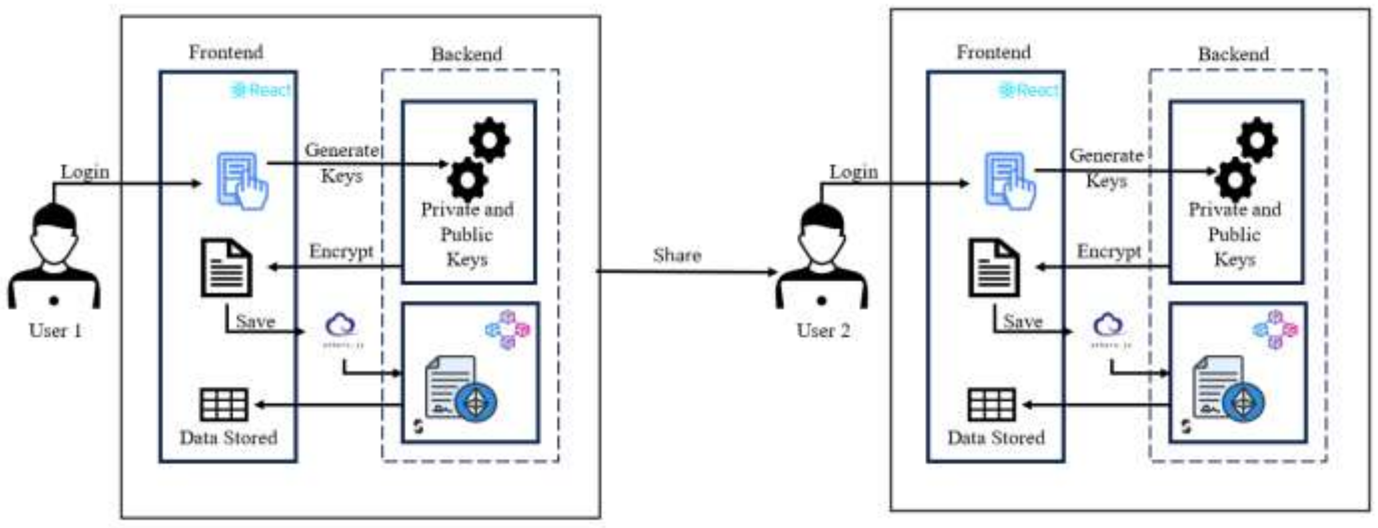
- When a user initiates a file upload, the system encrypts the file using end-to-end encryption mechanisms, ensuring confidentiality during transit and storage.

c. User Interface:

- Implement a robust identity management system to ensure that only authenticated and authorized users from organizations can participate in the file-sharing network.

- Develop a user interface using HTML, CSS, and JS, providing a seamless and intuitive experience for both administrators and end-users to interact with the file-sharing system.

d. Architecture:



e. Code:

```
pragma solidity ^0.8.0;
contract FileSharingContract {
struct SharedFile {
address owner;
string fileHash;
uint256 timestamp;
}
mapping(string => SharedFile) public sharedFiles;
event FileShared(address indexed owner, string fileHash, uint256 timestamp);
function storeFile(string memory fileHash) public {
require(sharedFiles[fileHash].timestamp == 0, "File already exists");
SharedFile memory newFile = SharedFile({
owner: msg.sender,
fileHash: fileHash,
timestamp: block.timestamp
});
sharedFiles[fileHash] = newFile;
}
function shareFile(string memory fileHash) public {
require(sharedFiles[fileHash].timestamp != 0, "File does not exist");
emit FileShared(msg.sender, fileHash, block.timestamp);
}
function getFileInfo(string memory fileHash) public view returns (address, uint256) {
SharedFile memory file = sharedFiles[fileHash];
require(file.timestamp != 0, "File does not exist");
return (file.owner, file.timestamp);
}
}
```

Signature of Supervisor: