

BSSPD: A BLOCKCHAIN-BASED SECURITY SHARING SCHEME FOR PERSONAL DATA WITH FINE-GRAINED ACCESS CONTROL

X.S Asha Shiny¹, Danamagari Anusha², L. Varshitha Reddy², Medicharla Chakravarthi²,
Bejjanki Vighnesh Reddy²

^{1,2}Department of Information Technology

^{1,2}CMR Engineering College, Kandlakoya, Medchal, Hyderabad.

ABSTRACT

Privacy protection and open sharing are the core of data governance in the AI-driven era. A common data-sharing management platform is indispensable in the existing data-sharing solutions, and users upload their data to the cloud server for storage and dissemination. However, from the moment users upload the data to the server, they will lose absolute ownership of their data, and security and privacy will become a critical issue. Although data encryption and access control are considered up-and-coming technologies in protecting personal data security on the cloud server, they alleviate this problem to a certain extent. However, it still depends too much on a third-party organization's credibility, the Cloud Service Provider (CSP). In this paper, we combined blockchain, ciphertext-policy attribute-based encryption (CP-ABE) and Inter Planetary File System (IPFS) to address this problem to propose a blockchain-based security sharing scheme for personal data named BSSPD. In this user-centric scheme, the data owner encrypts the sharing data and stores it on IPFS, which maximizes the scheme's decentralization. The address and the decryption key of the shared data will be encrypted with CP-ABE according to the specific access policy, and the data owner uses blockchain to publish his data-related information and distribute keys for data users. Only the data user whose attributes meet the access policy can download and decrypt the data. The data owner has fine-grained access control over his data, and BSSPD supports an attribute-level revocation of a specific data user without affecting others. To further protect the data user's privacy, the ciphertext keyword search is used when retrieving data. We analyzed the security of the BSSPD and simulated our scheme on the EOS blockchain, which proved that our scheme is feasible. Meanwhile, we provided a thorough analysis of the storage and computing overhead, which proved that BSSPD has a good performance.

Keywords: Block chain, Security sharing scheme, Fine grained access control.

1. INTRODUCTION

The development of 5G and Internet of Things technology provides a large amount of training data for the rapid implementation of artificial intelligence (AI). At the same time, data security and privacy protection have become the most interesting topics in data governance and sharing. Powerful data mining and analysis have brought potential threats to personal privacy protection. Traditionally, most people choose to outsource their data to cloud servers for sharing and dissemination. However, most of the data stored in the cloud is very sensitive, especially those data generated by IoT devices that are closely related to human life. These data have their particularities and may contain personal-related information such as life, work, and healthcare; once personal data is stolen or leaked illegally and linked to the data owner's real identity, it may bring great trouble to an individual. Therefore, integrating data and generating value while ensuring data security and privacy have become a significant challenge for all contemporary companies that use big data and AI.

At present, researchers have proposed many secure sharing schemes in the cloud environment [1–3]. These schemes seem to solve the security and privacy issues during data sharing. Nevertheless, these

schemes all have a standard feature: they are overly dependent on the Cloud Service Provider (CSP). They believe that the CSP is a trusted third-party organization, and the CSP is semi trustable, which means that the CSP will be curious about the data but will not destroy it. It means that the following situations are always inevitable:

1. The CSP itself may make profits from the user's private data, or its insiders may do evil and cause the user's privacy disclosure. Although some methods, such as attribute-based encryption algorithms, can achieve user-defined access policies that seem user centric, these methods still require a trusted third party to generate and manage user keys. It is impossible to exclude the possibility of collusion between these trusted centers. All these will lead to the fact that once the data owners upload their data to the cloud server, they will no longer have their data's absolute possession.
2. The data is centrally stored on cloud servers and managed by the CSP. An inevitable single point of failure may lead that users cannot obtain their data generally by using the cloud service. The CSP can improve data security and service stability by utilizing disaster recovery backup. However, some irresistible factors will prevent users from using cloud services to obtain their data, such as political factors
3. To provide better service, the CSP needs to spend more money to buy servers, hire better employees, rent the data center venues, and so on. These costs are increasing gradually, and the CSP cost is also increasing and the construction of the management platform. Users ultimately pay the operating costs of the CSP.

From the above point of view, to better protect data security and personal privacy, it is very urgent to design a whole user-centric data-sharing scheme to solve the above problems. In this scheme, we do not need to rely on any trusted third party to store and disseminate data, nor do we worry that the data will be inaccessible. Fortunately, with the emergence and development of Bitcoin [4], as a decentralized and self-organized cryptocurrency, its underlying technology blockchain can elegantly help us realize such a data security sharing scheme [5]. In this paper, we proposed a data sharing scheme based on blockchain. The main contributions of this paper are as follows:

1. A user-centric data security sharing scheme named BSSPD is proposed, which combines blockchain, CP-ABE, and IPFS. The data owner encrypts his sharing data and stores it on IPFS to maximize decentralization, and BSSPD allows the data owners to have fine-grained access control over their data. Moreover, it supports revoking permissions of a specific data user at an attribute level without affecting others
2. In BSSPD, the data owner publishes data-related information and distributes decryption keys for data users through the blockchain. To avoid denial of service attacks, data users need to complete a proof of work (PoW) before registering, which is like the mining process of Bitcoin, and the data owner can adjust the target of PoW according to the number of data users in the system
3. BSSPD sets ciphertext keyword indices for each data related data user. Combined with CP-ABE, it further prevents the privacy disclosure that data labels may cause to the data owner and protects the data user's privacy during retrieval
4. We experimented with our scheme on the EOS blockchain and provided the detailed implementation of algorithms and Smart Contracts. Together with the security analysis, it proved that our scheme is feasible
5. We used five MacBooks to build an EOS private chain in the laboratory environment and simulated our scheme. Analysis of storage and computing overhead proved that BSSPD.

2. LITERATURE SURVEY

Swan et al. [6] studied on the concept of blockchains, a new form of information technology that could have several important future applications. One is blockchain thinking, formulating thinking as a blockchain process. This could have benefits for both artificial intelligence and human enhancement, and their potential integration. Blockchain thinking is outlined here as an input-processing-output computational system.

Zyskind et al. [7] described a decentralized personal data management system that ensures users own and control their data. This paper implemented a protocol that turns a block chain into an automated access-control manager that does not require trust in a third party. Unlike Bit coin, transactions in this system are not strictly financial they are used to carry instructions, such as storing, querying, and sharing data. Finally, this paper discussed possible future extensions to block chains that could harness them into a well-rounded solution for trusted computing problems in society.

Azaria et al. [8] proposed a novel, decentralized record management system to handle EMRs, using blockchain technology. This system gives patients a comprehensive, immutable log and easy access to their medical information across providers and treatment sites. Leveraging unique blockchain properties, MedRec manages authentication, confidentiality, accountability, and data sharing crucial considerations when handling sensitive information. A modular design integrates with providers' existing, local data storage solutions, facilitating interoperability and making our system convenient and adaptable. This paper incentivized medical stakeholders (researchers, public health authorities, etc.) to participate in the network as blockchain "miners".

Xia et al. [9] proposed MeDShare, a system that addresses the issue of medical data sharing among medical big data custodians in a trust-less environment. The system is blockchain-based and provided data provenance, auditing, and controlled the shared medical data in cloud repositories among big data entities. MeDShare monitored entities that access data for malicious use from a data custodian system. In MeDShare, data transitions and sharing from one entity to the other, along with all actions performed on the MeDShare system, are recorded in a tamper-proof manner. The design employed smart contracts and an access control mechanism to effectively track the behavior of the data and revoke access to offending entities on detection of violation of permissions on data.

Dubovitskaya et al. [10] proposed a framework on managing and sharing EMR data for cancer patient care. In collaboration with Stony Brook University Hospital, this work implemented framework in a prototype that ensures privacy, security, availability, and fine-grained access control over EMR data. The proposed work can significantly reduce the turnaround time for EMR sharing, improved the decision making for medical care, and reduced the overall cost.

Liang et al. [11] proposed an innovative user-centric health data sharing solution by utilizing a decentralized and permissioned blockchain to protect privacy using channel formation scheme and enhance the identity management using the membership service supported by the blockchain. A mobile application is deployed to collect health data from personal wearable devices, manual input, and medical devices, and synchronize data to the cloud for data sharing with healthcare providers and health insurance companies.

Fan et al. [12] proposed a scheme based on a blockchain to solve the privacy issues in content-centric mobile networks for 5G. This paper implemented the mutual trust between content providers and users. Besides, the openness and tamper-resistant of the blockchain ledger ensure the access control and privacy of the provider. With the help of a miner, selected from users, this work can maintain the public ledger expediently. Also, in return, this work shared the interesting data with low overhead, network delay and congestion, and then achieve green communication.

Zhang et al. [13] established a mutual trust data sharing framework to break these data barriers. The framework is based on the distributed and temper-proof attributes of blockchain. This work implemented a prototype based on Hyperledger Fabric. The proposed system combined supervision and fine-grained data access control based on smart contracts, which provided a secure and trustless environment for data sharing. This work further compared this system with existing data sharing schemes.

Zhou et al. [14] proposed a double-blind paper review system to preserve the authors and reviewers' anonymity. This system also addressed issues concerning the reviewer's payment, inconsistent review metrics, and biased reviews. The proposed solution utilized the Hyperledger Fabric blockchain with the InterPlanetary File System (IPFS). The blockchain smart contracts provided a base for financial transactions between paper publishers and the reviewers. Hence, this work introduced AcadCoin, a novel cryptocurrency used for supporting said financial transactions.

Patel et al. [15] examined the blockchain concept, which enables parties to establish consensus without relying on a central authority. This work developed a framework for cross-domain image sharing that uses a blockchain as a distributed data store to establish a ledger of radiological studies and patient-defined access permissions. The blockchain framework is shown to eliminate third-party access to protected health information, satisfy many criteria of an interoperable health system, and readily generalize to domains beyond medical imaging.

3. IMPLEMENTATION DETAILS OF OUR SCHEME

To achieve our goal, we will construct a CP-ABE which supports permission revocation and combine it with the EOS blockchain to implement our scheme. This section will elaborate on the details of our Smart Contracts deployed on EOS blockchain and concrete construction of BSSPD.

Smart Contract Design: To make the logic clearer, we divide the Smart Contract in the scheme into two parts: UMContract and DSContract. UMContract is used to manage DUs' identity, while DSContract is used to handle business operations related to data sharing. In the contract, we will use self to represent the account of the DO who created the contract. We will describe the detailed design of these two contracts.

User Management Contract (UMContract): The UMContract is composed of five function interfaces: SetTarget, GetUserByUid, Apply, Register, and Authenticate. We initialize UMContract as follows.

Let three-tuple (A, uid, Pk_{com}) denote a DU, and create a multi_index named table_user for it in which A is an EOS account of the DU, uid is the unique ID assigned by the DO, and Pk_{com} is a public key of the DU used for communication with the DO. Let A be the primary key of table user whose corresponding index is account_idx. Let uid_idx be a secondary index corresponding to uid. Let target be the target value of PoW.

1. *SetTarget:* When UMContract receives action (UMContract, SetTarget, Auth, (newTarget)), this function interface will be triggered to execute. It can only be invoked by the DO who created the contract to adjust the difficulty of PoW. When there are too many users in the system, the DO can increase the difficulty of PoW

```

Input: newTarget
Output: bool
1 if msg.sender is not _self then
2   throw;
3 else
4   target = newTarget;
5   return true;
6 end

```

Algorithm 1: SetTarget.

2. *GetUserUid*: When UMContract receives action (UMContract, GetUserByUid, Auth, (account)), this function interface will be triggered to execute. It is used to get all the information of a DU according to his uid and can only be invoked by the DO who created the contract
3. *Apply*: When UMContract receives action (UMContract, Apply, Auth, (from, pk, nonce)), this function interface will be triggered to execute. It is invoked by the DU to apply for registration in the system
4. *Register*: When UMContract receives action (UMContract, Register, Auth, (account, id)), this function interface will be triggered to execute. It is used to complete the registration of a DU and can only be invoked by the creator of the contract
5. *Authenticate*: When UMContract receives action (UMContract, Authenticate, Auth, (from, method, account, id, args)), this function interface will be triggered to execute. It is used to authenticate the identity of a DU, which is invoked by another contract and returns the result to the invoker.

Date Sharing Contract (DSContract): The DSContract is composed of six function interfaces: *SetPK*, *SetSK*, *AddData*, *PolicyUpdate*, *Search* and *EndSearch*, and *Remove*. We initialize *DSContract* as follows.

Let PK denote the system public parameters. Let two-tuple (A, SK) be the corresponding relationship between the *DU*'s account and his attribute private key, and the multi_index table *sk* is created for it. Let *A* be the primary key of table *sk* whose corresponding index is *ua_idx*. Let two tuples (fid, cf) denote the shared data in which *fid* is the id of shared data and *cf* is the data-related information. Then, create a multi_index *data_table* for it, where *fid* is the primary key and *fid_idx* is the corresponding index. Let four tuples (id, A, t, fid) be an index of DU related to shared data in which *A* is the EOS account of DU, *t* is the search token, and *fid* is the id of shared data in *data_table*, then create a multi_index *search_table* for it. Let *sa_idx*, *t_idx*, *sf_idx* be the secondary indices of *search_table*, corresponding to *A*, *t*, and *fid*, respectively.

```

Input: uid
Output: all information of DU
1 if msg.sender is not _self then
2   throw;
3 else
4   user_row = uid_idx.find(uid);
5   return user_row;
6 end

```

Algorithm 2: GetUserByUid

```

Input: from, pk, nonce
Output: bool
1 u = account_idx.find(from)
2 if u != null then
3   u.Pkcom = pk;
   account_idx.modify(u);
4   return true;
5 else
6   pow = SHA256(SHA256(from | pk | nonce));
7   if pow > target then
8     return false;
9   else
10    u.A = from;
11    u.Pkcom = pk;
12    account_idx.emplace(u);
13    return true;
14  end
15 end

```

Algorithm 3: Apply.

```

Input: account, id
Output: bool
1 if msg.sender is not _self then
2   throw;
3 else
4   u = account_idx.find(account);
5   if u == null then
6     return false;
7   else
8     u.uid = id;
10    account_idx.modify(u);
11    return true;
12  end
13 end

```

Algorithm 4: Register.

- 1) *SetPK*: When DSContract receives action (DSContract, SetPK, Auth, (newPk)), this function interface will be triggered to execute. It can only be invoked by the DO to set and update the system public parameters.
- 2) *SetSK*: When DSContract receives action (DSContract, SetSK, Auth, (account, sk)), this function interface will be triggered to execute. It can only be invoked by the DO to set and update the private keys of the DU.

```

Input: from, method, account, id, args
Output: null
u = account_idx.find(account)
1 if u != null then
2   if u.id == id then
3     send action (from, method, (_self, true, args));
4   else
5     send action (from, method, (_self, false, args));
6   end
7 else
8   send action (from, method, (_self, false, args));
9 end

```

Algorithm 5: Authenticate.

- 3) *AddData*: When DSContract receives action (DSContract, AddData, Auth, this function interface will be triggered to execute. It is used to publish the sharing data and add the indices for the relevant DUs. There can be multiple index relationships. For clarity, we only add an index for one DU here. It can only be invoked by the DO.
- 4) *PolicyUpdate*: When DSContract receives action (DSContract, PolicyUpdate, Auth, this function interface will be triggered to execute. It can only be invoked by the DO and used to update the access policy for a certain shared data. In this way, the DO can revoke the access permission of a DU to this shared data.

- 5) *Search and End Search*: When DSContract receives action (DSContract, Search, Auth, this function interface will be triggered to execute. These two function interfaces work together to complete the retrieval of shared data. Because we have divided BSSPD into two contracts, it needs to invoke UMContract to verify the identity of the DU during the retrieval.
- 6) *Remove*: When DSContract receives action (DSContract, Remove, Auth, this function interface will be triggered to execute. It is used to remove a shared data and the search indices related to this data. It can only be invoked by the DO.

Other Security Problem

Data Security: Data security includes the confidentiality, integrity, and availability of the shared data. In our scheme, the large capacity sharing data of the DO is encrypted using an efficient asymmetric encryption algorithm such as AES and uploaded to IPFS. The IPFS will split the encrypted data and store them on different IPFS nodes in a distributed manner. The access will be routed through the dynamic hash table maintained by each node, and a certain redundancy mechanism will ensure fault tolerance. Besides, IPFS also provides version control like Git. Thus, data encryption and storage in blocks ensure the confidentiality of the shared data. The integrity is guaranteed by dynamic hash table routing, and the tampered data blocks will not be available. The redundant storage and incentive mechanisms of IPFS ensure that users can access their data at any time. If IPFS is secure, then the data stored on it in our scheme is secure.

Privacy Analysis: In a data-sharing system, privacy includes the content of the DO's shared data and the traces of the DU when using the data. In our scheme, the DO will encrypt the address of the shared data and the corresponding decryption key with CP-ABE according to the established access policy. Then, the ciphertext is stored on the blockchain, and only the DUs whose attribute set satisfies the access policy can obtain the data. The content of the data will not be leaked. For the traces generated by DUs, we encrypt the keywords corresponding to the sharing data. The DU invoked the trapdoor function to calculate the search token for the keyword that he needs to retrieve and then uses the search token for retrieving on the blockchain without revealing any information he wants. More importantly, the user's identity is represented in the form of an address on the blockchain, and the real information of the user will not be exposed.

Fine-Grained Access Control: In our scheme, the fine-grained access control of shared data is realized by CPABE. The DO can make different access policies through LSSS and assign different attributes to DUs. Meanwhile, fine-grained access control should also include fine-grained revocation. The proposed scheme draws on the identity-based broadcast encryption scheme, in which the DO assigns a unique uid for each DU, and the uid will be used as a user attribute, embedded in the ciphertext together with the general attributes. Each general attribute in the ciphertext carries a revocation list, and the DU whose uid in this list no longer has the corresponding attribute, so that it achieves the purpose of directly revoking a DU's attribute.

Avoid a Single Point of Failure: Compared with traditional cloud storage solutions, there is no centralized third party in our proposed scheme. Blockchain and IPFS used in BSSPD are all distributed technologies. Even if some of the nodes fail, the availability of the whole scheme will not be affected. More importantly, the BitTorrent protocol adopted by IPFS can enjoy a high throughput only by requiring paying a small number of fees to incentive storage nodes. Simultaneously, the EOS blockchain is free to users, only the DO needs to mortgage some system tokens in exchange for storage and CPU resources, and these tokens can also be redeemed.

User-Centric: In our proposed scheme, the DO can generate public parameters and the system master key and generate and distribute the private keys for DUs according to their attributes. Moreover, the

DO can formulate access policies arbitrarily to assign and revoke the permission of DUs. All of these are controlled by the DO without any trusted third party. In this manner, the DO has completed control over his shared data.

Identity Authentication: The user generates his identity in the blockchain through an asymmetric encryption algorithm with generating key pairs, whose cost is too low. In our proposed scheme, since the uid is embedded in the ciphertext of CP-ABE as an attribute, the DUs may register many uids and use different uids to search and decrypt the shared data, which increases the burden of the DO. To prevent such attacks, BSSPD requires identity authentication. Before applying for registration, the DU needs to perform a PoW, which is like Bitcoin mining. The DO can adjust the difficulty of PoW according to the total number of DUs in the system. User management and identity authentication are carried out on the blockchain, and only authenticated users can perform operations. These are all executed in Smart Contract ensuring transparency and security.

3.1 WEB3 Python Package

web3.py is a Python library that provides a simple and easy-to-use API for interacting with Ethereum networks using JSON-RPC. It allows developers to easily interact with smart contracts, send transactions, and access blockchain data.

Some of the key features of web3.py include:

Contract interaction: web3.py provides an API for interacting with smart contracts on the Ethereum network. This includes functions for deploying contracts, calling contract functions, and reading contract data.

Transaction management: web3.py makes it easy to send transactions to the Ethereum network, including specifying gas prices and gas limits.

Event listening: web3.py allows developers to listen for events emitted by smart contracts on the Ethereum network, making it easy to build real-time applications that react to blockchain data.

Blockchain data access: web3.py provides functions for accessing blockchain data like account balances, transaction history, and block data.

Integration with popular wallets: web3.py integrates with popular Ethereum wallets like Metamask and Geth, making it easy to manage accounts and interact with the network. Overall, web3.py is a powerful tool for building decentralized applications on the Ethereum network using Python.

3.2 Blockchain

Blockchain is a decentralized, digital ledger technology that is used to record and store data in a secure and transparent manner. It is a distributed ledger, meaning that it is maintained by a network of computers, rather than being controlled by a single entity. Each block in the chain contains a set of transactions, and once a block is added to the chain, it cannot be altered or deleted. This makes blockchain an immutable and tamper-resistant technology that is particularly well-suited for storing and transmitting sensitive data.

Blockchain technology is perhaps best known for its use in cryptocurrencies like Bitcoin and Ethereum, but it has a wide range of other potential applications as well. These include supply chain management, identity verification, voting systems, and more. The decentralized nature of blockchain means that it has the potential to disrupt a variety of industries and business models by enabling trust and transparency in transactions and data exchange.

Concepts

There are several key concepts that are important to understand when it comes to blockchain technology:

Decentralization: Blockchain is a decentralized technology, meaning that it is not controlled by any single entity, but rather maintained by a network of participants. This increases transparency, security, and resilience.

Distributed ledger: Blockchain technology uses a distributed ledger to record and store data. Each block in the chain contains a set of transactions, and once a block is added to the chain, it cannot be altered or deleted.

Cryptography: Blockchain technology uses advanced cryptographic algorithms to secure transactions and data exchange, making it highly resistant to hacking and cyber attacks.

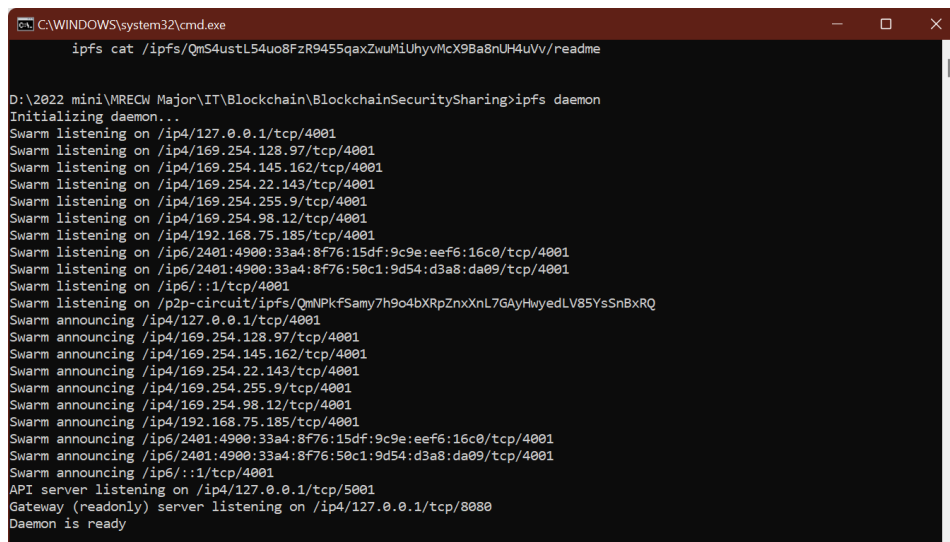
Consensus mechanism: In a blockchain network, participants must agree on the validity of transactions before they are recorded on the blockchain. Different blockchain networks use different consensus mechanisms to achieve this, such as Proof of Work or Proof of Stake.

Smart contracts: Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They can be used to automate complex transactions and ensure that all parties involved in a transaction adhere to the terms of the contract.

Tokenization: Blockchain technology enables the creation of digital tokens that can be used to represent a variety of assets, such as currencies, commodities, or even real estate.

4. RESULTS AND DISCUSSION

To run project first double click on 'Start_IPFS.bat' file to start IPFS server and get below screen



```
C:\WINDOWS\system32\cmd.exe
ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZiwuMiUhyvMcX9Ba8NU4H4UVv/readme

D:\2022 mini\WRECH Major\IT\Blockchain\BlockchainSecuritySharing>ipfs daemon
Initializing daemon...
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/169.254.128.97/tcp/4001
Swarm listening on /ip4/169.254.145.162/tcp/4001
Swarm listening on /ip4/169.254.22.143/tcp/4001
Swarm listening on /ip4/169.254.255.9/tcp/4001
Swarm listening on /ip4/169.254.98.12/tcp/4001
Swarm listening on /ip4/192.168.75.185/tcp/4001
Swarm listening on /ip6/2401:4900:33a4:8f76:15df:9c9e:eef6:16c0/tcp/4001
Swarm listening on /ip6/2401:4900:33a4:8f76:50c1:9d54:d3a8:da09/tcp/4001
Swarm listening on /ip6::1/tcp/4001
Swarm listening on /p2p-circuit/ipfs/QmNPKfSamy7h9o4bXRpZnxXnL7GAYHwyedLV85YsSnBxRQ
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/169.254.128.97/tcp/4001
Swarm announcing /ip4/169.254.145.162/tcp/4001
Swarm announcing /ip4/169.254.22.143/tcp/4001
Swarm announcing /ip4/169.254.255.9/tcp/4001
Swarm announcing /ip4/169.254.98.12/tcp/4001
Swarm announcing /ip4/192.168.75.185/tcp/4001
Swarm announcing /ip6/2401:4900:33a4:8f76:15df:9c9e:eef6:16c0/tcp/4001
Swarm announcing /ip6/2401:4900:33a4:8f76:50c1:9d54:d3a8:da09/tcp/4001
Swarm announcing /ip6::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

In above screen IPFS server started and now double click on 'runServer.bat' file to start python DJANGO server and get below screen

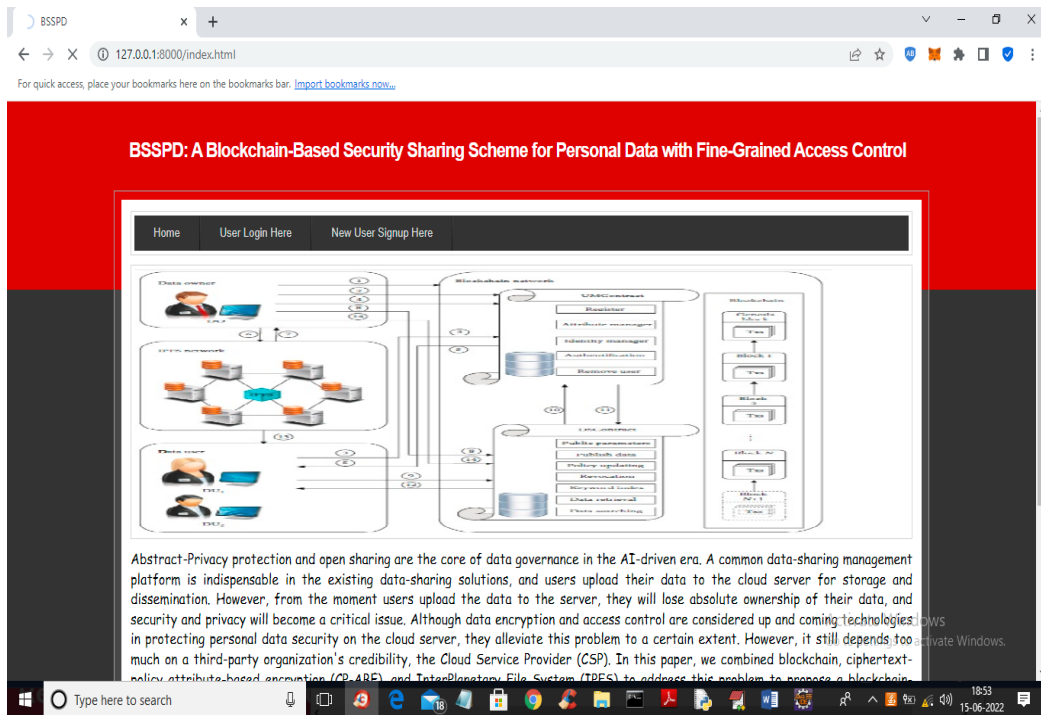
```

C:\WINDOWS\system32\cmd.exe
D:\2022 mini\MRECW Major\IT\Blockchain\BlockchainSecuritySharing>python manage.py runserver
Performing system checks...

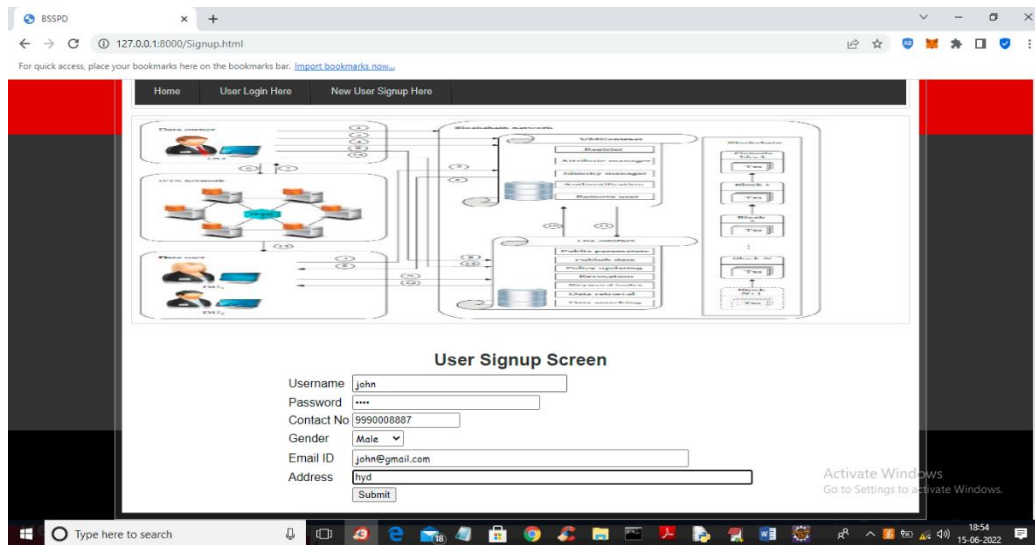
System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
October 05, 2022 - 01:06:40
Django version 2.1.7, using settings 'BlockchainSecurity.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
    
```

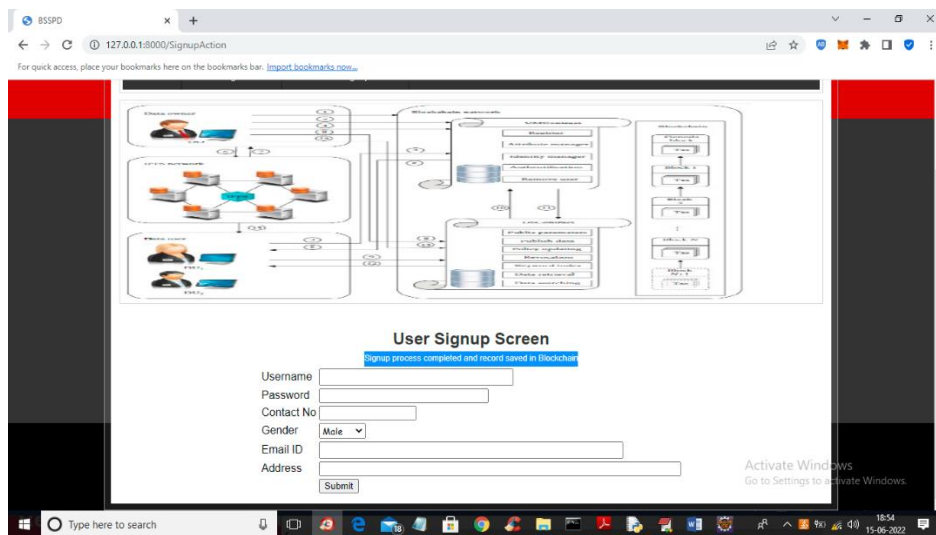
In above screen python DJANGO server started and now open browser and enter URL as ‘http://127.0.0.1:8000/index.html’ and press enter key to get below screen



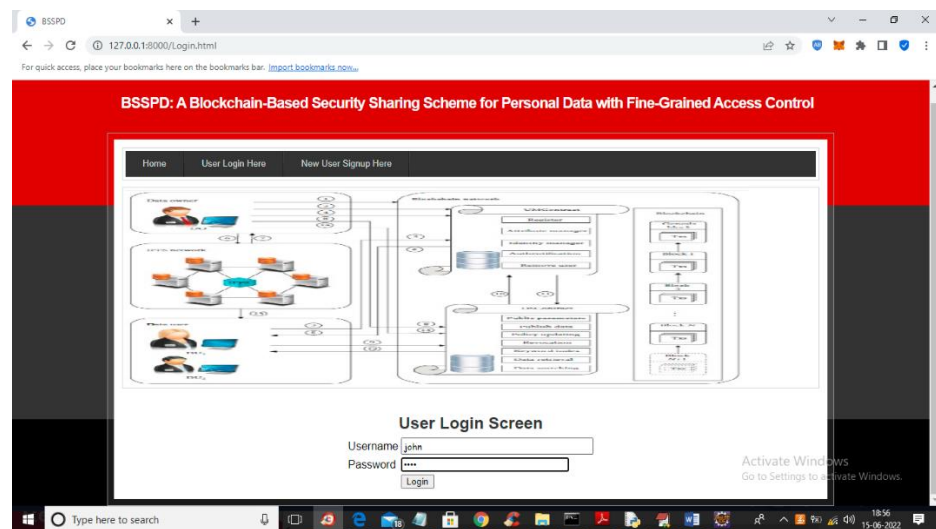
In above screen click on ‘New User Signup Here’ link to add new user to Blockchain



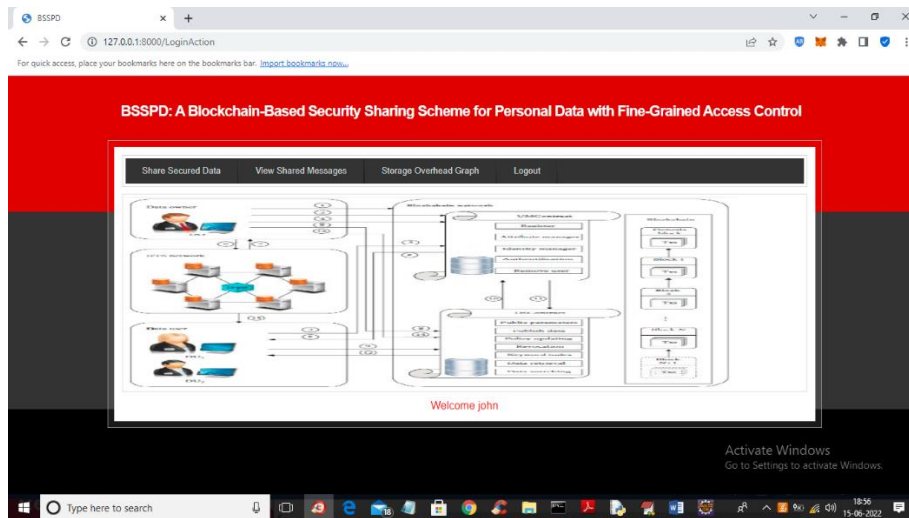
In above screen user is signup and press button to get below output



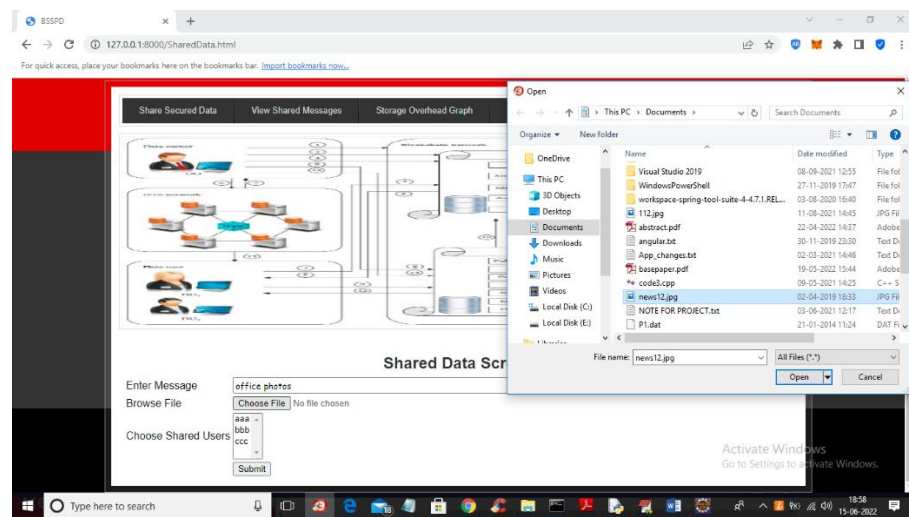
In above screen user signup process completed and similarly you can add any number of users and now click on 'User Login Here' link to get below login screen



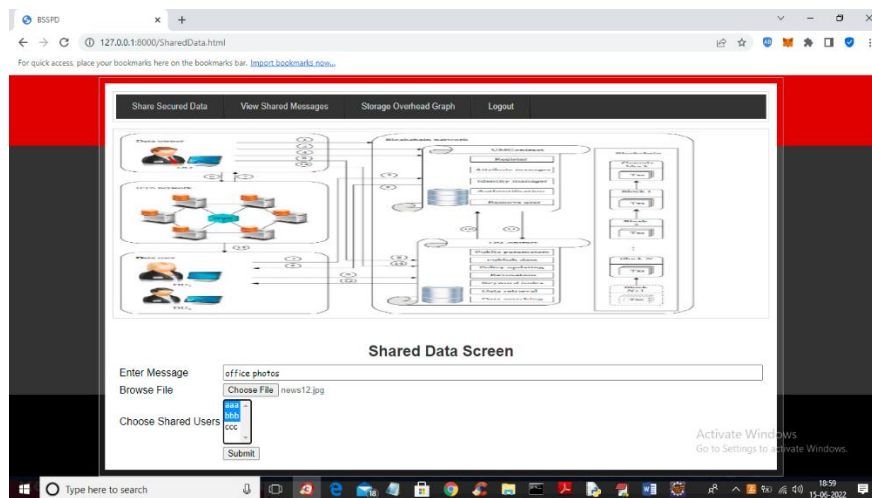
In above screen user is login and press button to get below output



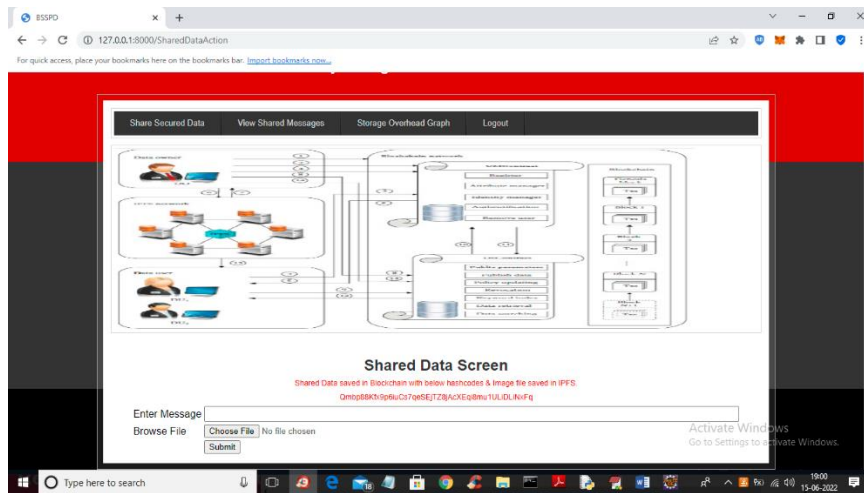
In above screen user logged in successfully and now click on ‘Share Secured Data’ link to share data with other users



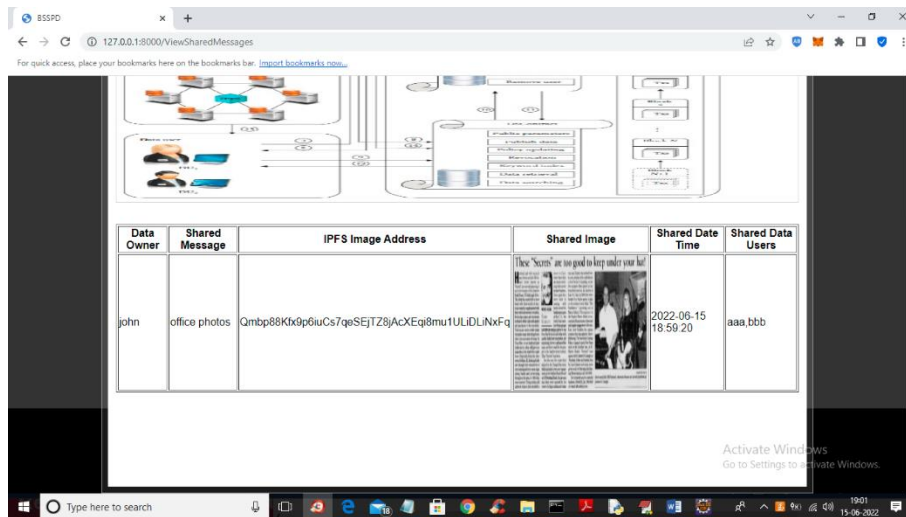
In above screen user can enter some message and then upload image and by holding CTRL KEY you can select names of users with whom you want to share this data and press button to get below output



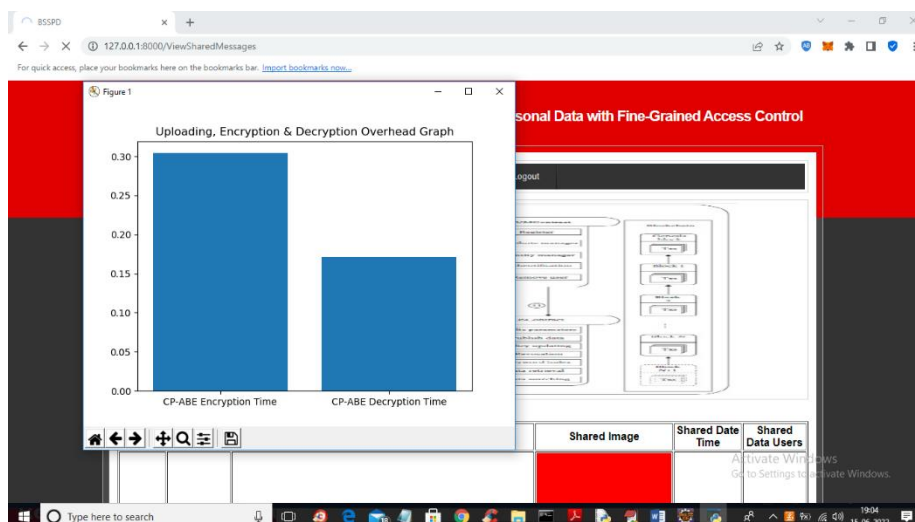
In above screen ‘John’ is sharing data with user ‘aaa’ and ‘bbb’ and both users can decrypt and view data but user ‘ccc’ cannot view it.



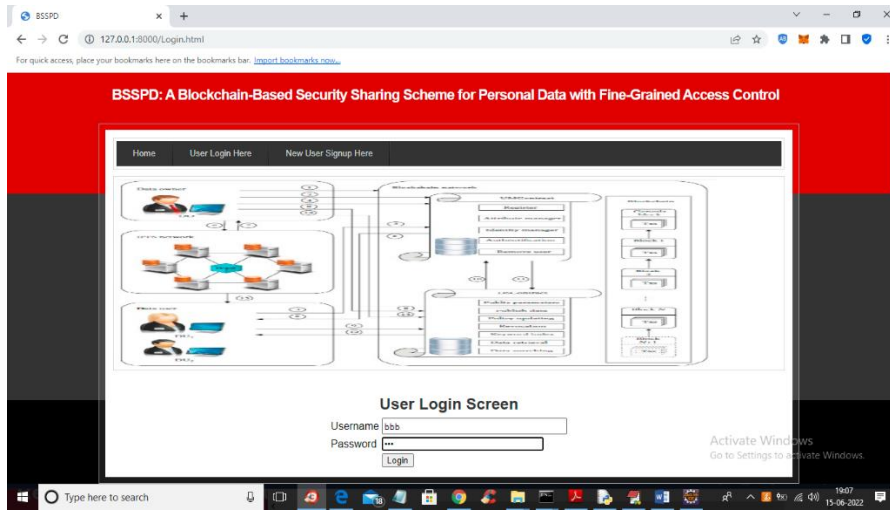
In above screen we can see sharing attributes stored at Blockchain and images and decryption keys stored at IPFS and now click on ‘View Shared Messages’ link to view own messages and other users shared messages so ‘John’ is the data owner so he can view his own upload and others shared data.



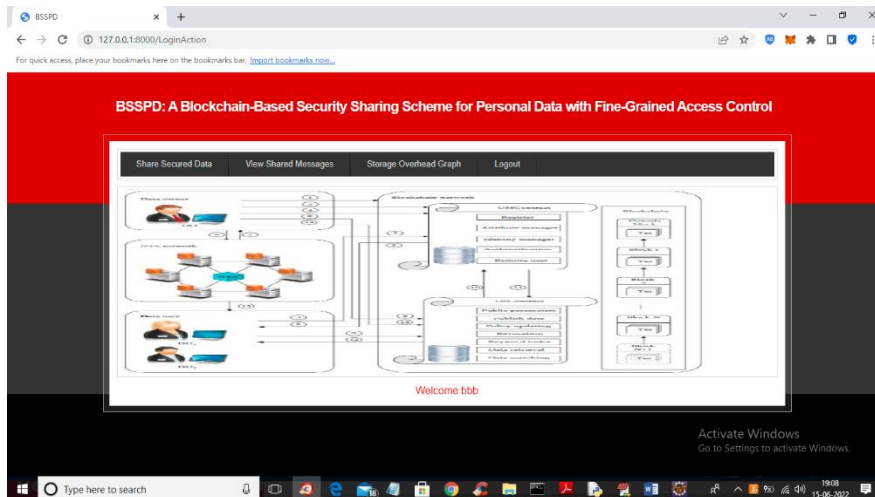
In above screen we can see data owner name, shared messages with IPFS address and we can see names of shared users list and now we can check weather aaa or bbb can view this data or not and now click on ‘Storage Overhead Graph’ link to view encryption and decryption time overhead



In above screen x-axis represents encryption and decryption and y-axis represents time overhead and now logout and login as ‘bbb’ user to view shared data.



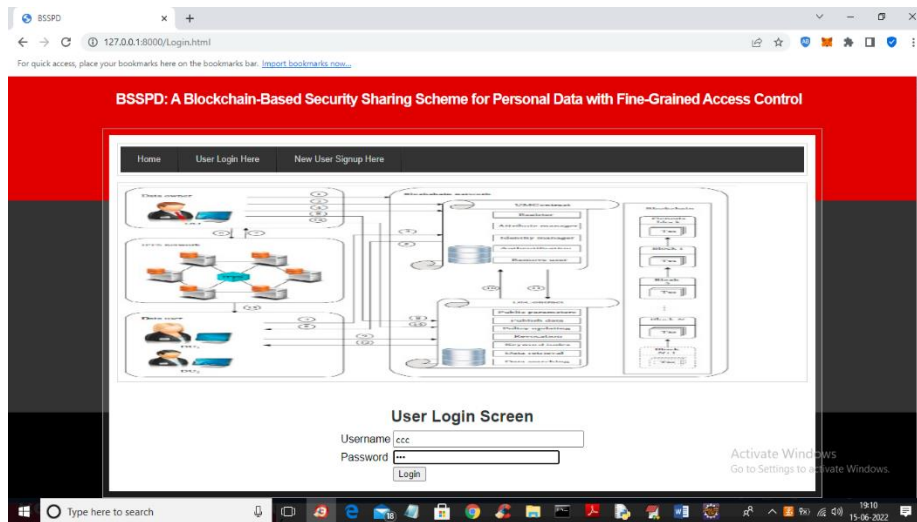
In above screen shared user ‘bbb’ is login and after login will get below output



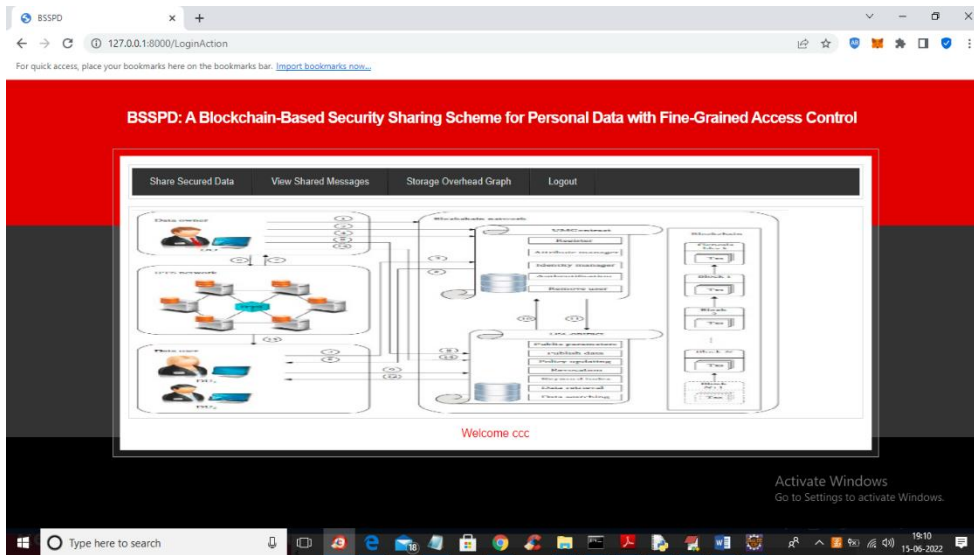
Now in above screen ‘bbb’ can click on ‘View Shared Messages’ link to view all users shared data

| Data Owner | Shared Message | IPFS Image Address | Shared Image | Shared Date Time | Shared Data Users |
|------------|----------------|---|--------------|---------------------|-------------------|
| aaa | school photos | QmYVDEdDhQqPwXVok638LJvesHy5Xkp3yK3PB2uwr52TuL | [Redacted] | 2022-06-15 17:28:15 | bbb |
| john | office photos | Qmbp88Ktk9p6iuCs7qeSEJtZ8JAcXEai8mu1ULiDLiNxFq | | 2022-06-15 18:59:20 | aaa,bbb |
| aaa | test data | QmYVBwQio8Lmb5Le1fBCDCwYtYQqvqV3XCPBDatgySs7wIB | | 2022-06-15 19:04:25 | bbb |

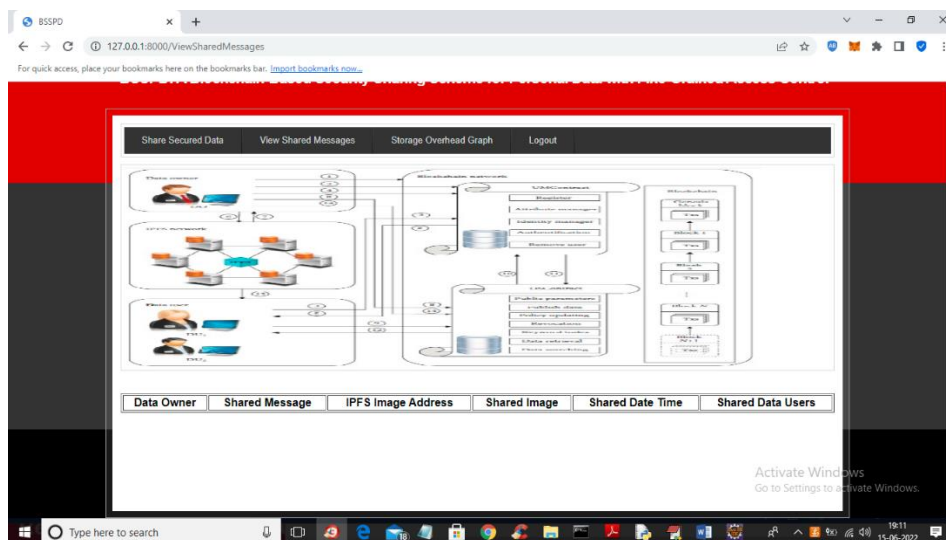
In above screen ‘bbb’ can view shared data from aaa and john and now logout and login as ‘ccc’ and nobody shared data with ‘ccc’ so he cannot access any data



In above screen user 'ccc' is login and after login will get below screen



Now in above screen 'ccc' can click on 'View Shared Messages' link to get below output



In above screen 'ccc' can get empty table as nobody shared data with him. Similarly, any number of user can sign up and share data

5. CONCLUSION

In the AI-driven era, a user-centered sharing model is proposed to open data while ensuring data privacy. We combined blockchain, CP-ABE, and IPFS to propose a blockchain-based security data-sharing scheme with fine grained access control and permission revocation. In our proposed scheme, the DO encrypts his data and uploads it to IPFS, then encrypts the returned address and decryption key by CP-ABE. Only DUs whose attributes satisfy the access policy can decrypt and obtain the data. There is no centralized node in the scheme, and the DO has completed control over his shared data, which promises privacy and security. To achieve the goal, we have implemented our scheme on the EOS blockchain. The security and performance analysis proves that our scheme is feasible and practical and has a good performance. We can also add a cryptocurrency to introduce an economic system for data sharing and further enrich our scheme's functions. At the same time, there are many shortcomings in our scheme. For example, the CPABE we designed with permission revocable does not have the best performance. There are also many types of research on CP-ABE. We can use a CP-ABE with better performance to improve our scheme. Besides, for the searchable encryption algorithm used in our scheme, the DO needs to distribute a secret key for each DU and store it on-chain. It also needs to maintain large amounts of indices for each shared data, which can be further optimized. At present, some researchers have proposed using blockchain to solve the fairness problem in searchable encryption algorithm.

In the future, we will study and discuss the endowment of a better ciphertext searchable algorithm to further optimize our scheme. Simultaneously, to make our scheme more practical, we can combine some studies with ours and put forward a data governance scheme that is more in line with the practical application.

REFERENCES

- [1] J. Li, Y. Zhang, X. Chen, and Y. Xiang, "Secure attribute-based data sharing for resource-limited users in cloud computing," *Computers & Security*, vol. 72, pp. 1–12, 2018.
- [2] S. Sundareswaran, A. Squicciarini, and D. Lin, "Ensuring distributed accountability for data sharing in the cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 4, pp. 556–568, 2012.
- [3] Cheng-Kang Chu, S. S. M. Chow, Wen-Guey Tzeng, Jianying Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 468–477, 2014.
- [4] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system," 2008, <https://bitcoin.org/bitcoin.pdf>.
- [5] Y. Xu, C. Zhang, G. Wang, Z. Qin, and Q. Zeng, "A blockchain-enabled duplicatable data auditing mechanism for network storage services," *IEEE Transactions on Emerging Topics in Computing*, 2020.
- [6] M. Swan, "Blockchain thinking: the brain as a decentralized autonomous corporation [commentary]," *IEEE Technology and Society Magazine*, vol. 34, no. 4, pp. 41–52, 2015.
- [7] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: using blockchain to protect personal data," in *2015 IEEE Security and Privacy Workshops*, pp. 180–184, San Jose, CA, 2015.
- [8] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "Medrec: using blockchain for medical data access and permission management," in *2016 2nd International Conference on Open and Big Data (OBD)*, pp. 25–30, Vienna, 2016.
- [9] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "Medshare: trust-fewer medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14757–14767, 2017.

- [10] Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang, "Secure and trustable electronic medical records sharing using blockchain," *AMIA Annual Symposium Proceedings*, vol. 2017, pp. 650–659, 2017.
- [11] X. Liang, J. Zhao, S. Shetty, J. Liu, and D. Li, "Integrating blockchain for data sharing and collaboration in mobile healthcare applications," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–5, Montreal, QC, 2017.
- [12] K. Fan, Y. Ren, Y. Wang, H. Li, and Y. Yang, "Blockchain based efficient privacy preserving and data sharing scheme of content-centric network in 5g," *IET Communications*, vol. 12, no. 5, pp. 527–532, 2017.
- [13] G. Zhang, T. Li, Y. Li, P. Hui, and D. Jin, "Blockchain-based data sharing system for AI-powered network operations," *Journal of Communications and Information Networks*, vol. 3, no. 3, pp. 1–8, 2018.
- [14] Zhou, I. Makhdoom, M. Abolhasan, J. Lipman, and N. Shariati, "A blockchain-based file-sharing system for academic paper review," in *2019 13th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pp. 1–10, Gold Coast, Australia, 2019.
- [15] V. Patel, "A framework for secure and decentralized sharing of medical imaging data via blockchain consensus," *Health informatics journal*, vol. 25, no. 4, pp. 1398–1411, 2018.