A Major Project Report

On

**BSSPD: A Blockchain-Based Security Sharing Scheme for**

**Personal Data with Fine-Grained Access Control**

Submitted to JNTU HYDERABAD

In Partial Fulfilment of the requirements for the Award of Degree of

## BACHELOR OF TECHNOLOGY IN

**INFORMATION TECHNOLOGY**

Submitted By

## Anusha Danamagari (198R1A1210)

**Varshitha Reddy Lattupalli (198R1A1231)**

## Chakravarthi Medicharla (198R1A1237)

## Vighnesh Reddy Bejjanki (198R1A1207)

Under the Esteemed guidance of

## Dr. X. S. Asha Shiny

Professor, Department of IT



# Department of Information Technology

**CMR ENGINEERING COLLEGE**

# (UGC AUTONOMOUS)

(Accredited by NAAC, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad) (Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

**(2022-2023)**

**CMR ENGINEERING COLLEGE**

# (UGC AUTONOMOUS)

(Accredited by NAAC, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad) (Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

**Department of Information Technology**



**CERTIFICATE**

This is to certify that the project entitled **“BSSPD: A Blockchain - Based Security Sharing**

**Scheme for Personal Data with Fine-Grained Access Control”** is a bonafide work carried

out by

## Anusha Danamagari (198R1A1210)

**Varshitha Reddy Lattupalli (198R1A1231)**

## Chakravarthi Medicharla (198R1A1237)

## Vighnesh Reddy Bejjanki (198R1A1207)

in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **INFORMATION TECHNOLOGY** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

|  |  |  |
| --- | --- | --- |
| Internal Guide |  | Head of the Department |
| **Dr. X. S. Asha Shiny** |  | **Dr. Madhavi Pingili** |
| Professor  Department of IT |  | Professor & HOD  Department of IT |
| CMR Engineering College |  | CMR Engineering College |
| Hyderabad |  | Hyderabad |

## DECLARATION

This is to certify that the work reported in the present project entitled **“BSSPD: A Blockchain - Based Security Sharing Scheme for Personal Data with Fine-Grained Access Control”** is a record of bonafide work done by us in the Department of Information Technology, CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

## 

## Anusha Danamagari (198R1A1210)

**Varshitha Reddy Lattupalli (198R1A1231)**

## Chakravarthi Medicharla (198R1A1237)

## Vighnesh Reddy Bejjanki (198R1A1207)

## ACKNOWLEDGEMENT

We are extremely grateful to **Dr.A.Srinivasula Reddy**, Principal and **Dr.Madhavi Pingili**, HOD, **Department of IT, CMR Engineering College** for their constant support.

We are extremely thankful to **Dr. X. S Asha Shiny,** Professor, Internal Guide, Department of IT, for her constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, we are very much thankful to our parents who guided us for every step.

## Anusha Danamagari (198R1A1210)

**Varshitha Reddy Lattupalli (198R1A1231)**

## Chakravarthi Medicharla (198R1A1237)

## Vighnesh Reddy Bejjanki (198R1A1207)

# CONTENTS

|  |  |  |
| --- | --- | --- |
| **TOPIC** | | **PAGE NO** |
| **ABSTRACT** | | **I** |
| **LIST OF FIGURES** | | **II** |
| **LIST OF TABLES** | | **III** |
| **1. INTRODUCTION** | | **1** |
| 1.1. Introduction & Objectives | | 1 |
| 1.2. Project Objectives | | 2 |
| 1.3. Purpose of the project | | 2 |
| 1.4. Existing System with Disadvantages | | 3 |
| 1.5. Proposed System With features | | 4 |
| 1.6. Input and Output Design | | 9 |
| **2. LITERATURE SURVEY** | | **10** |
| **3. SOFTWARE REQUIREMENT ANALYSIS** | | **13** |
| 3.1. Problem Specification | | 13 |
| 3.2. Functional Requirements | | 13 |
| 3.3. Feasibility Study  3.4. Performance Requirements | | 17  18 |
|  | |  |
| **4. SOFTWARE & HARDWARE REQUIREMENTS** | | **19** |
| 4.1. Software requirements | | 19 |
| 4.2. Hardware requirements | | 19 |
| **5. SOFTWARE DESIGN** | | **20** |
| 5.1. System Architecture | | 20 | | |
| 5.2. Dataflow Diagrams | | 21 | | |
| 5.3.UML Diagrams | | 23 | | |

|  |  |
| --- | --- |
| **6. CODING AND IMPLEMENTATION** | **27** |
| 6.1. Source code | 27 |
| 6.2. Implementation | 34 |
| **7. SYSTEM TESTING** | **54** |
| **8. OUTPUT SCREENS** | **58** |
| **9. CONCLUSION** | **63** |
| **10. FUTURE ENHANCEMENTS** | **64** |
| **11. REFERENCES** | **65** |

**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 project, 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 BBSPD 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.

I

# LIST OF FIGURES

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **FIGURE NO** | **DESCRIPTION** | **PAGE NO.** |
| 1 | 5.1 | System Architecture | 20 |
| 2 | 5.2 | Data Flow Diagram | 22 |
| 3 | 5.3.1 | Sequence Diagram | 24 |
| 4 | 5.3.2 | Use Case Diagram | 25 |
| 5 | 5.3.3 | Class Diagram | 26 |
| 6 | 6.2.1 | Smart Contract | 50 |
| 7 | 6.2.2 | Private Keys Generation | 51 |
| 8 | 6.2.3 | Deployment of Blockchain | 51 |
| 9 | 6.2.4 | Smart Contract Functions | 52 |
| 10  11  12  13  14  15  16  17  18  19  20  21 | 8.1  8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9  8.10  8.11  8.12 | IPFS Server  DJANGO Server  BSSPD Screen  User Signup Screen  Signup process completed  User login screen  User login successful  Shared data screen  Selecting users to share messages  Hash code generated  View shared messages  Encryption and Decryption graph | 56  56  57  57  58  58  59  59  60  60  61  62 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

II

**LIST OF TABLES**

III

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **TABLE NO** | **DESCRIPTION** | **PAGE NO** |
| 1 | 7.1 | Test Case | 54 |

**1. INTRODUCTION**

## 1.1. Introduction & Objectives

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. 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.

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 , as a decentralized and self-organized cryptocurrency, its underlying technology blockchain can elegantly help us realize such a data security sharing scheme . In this project, 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.

## 1.2. Project Objectives

By the end of this project, you will be having data sharing management platform which is built with blockchain technology that has fine grained access control over the data, without affecting others data with high cryptography.

## 1.3. Purpose of the Project

## In this project, we combined blockchain, ciphertext-policy attribute-based encryption (CP-ABE) and Inter Planetary File System (IPFS) to address the data sharing security 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.

## 1.4. Existing System with Disadvantages

**Hyperledger Fabric**

Hyperledger Fabric is designed for use in enterprise-level applications, and it is characterized by its modular architecture, permissioned network, and smart contract functionality, known as “chaincode”.

* The platform provides a high degree of security, privacy, and scalability, and it supports the development of custom blockchain solutions for various use cases across industries such as finance, supply chain, and healthcare.
* Hyperledger Fabric operates as a network of nodes, where each node performs a specific function, such as validating transactions, maintaining the ledger, and executing chaincode.
* Transactions are validated and ordered by a consensus mechanism, which ensures the integrity and consistency of the ledger.

Hyperledger fabric is an enterprise-level permission blockchain network. It is made up of various unique organizations or members that interact with each other to serve a specific purpose. For example, these organizations can be a bank, financial institution, or a supply chain network. Each organization is identified and they have a fabric certificate authority. These organizations are called members.Each member of the fabric can set up one or more authorized peers to participate in the network using the fabric certificate authority. All of these peers must be authorized properly.There is a client-sideapplication connected to the network written with the software development kit (SDK) of any particular programming language

### Disadvantages

**Complexity:** Hyperledger Fabric is a complex system that requires a lot of expertise to set up and maintain. This complexity can make it difficult to ensure proper security measures are in place and can increase the likelihood of human error, which can result in security vulnerabilities.

**Limited decentralization:** Hyperledger Fabric's permissioned nature limits the number of nodes that can participate in the network. This can limit the level of decentralization and increase the risk of a single point of failure

**Lack of anonymity:** Hyperledger Fabric's permissioned nature means that all participants in the network are identified and authenticated, which can limit anonymity. While this may be desirable

in some use cases, it can be a disadvantage in others where anonymity is important.

**Smart contract vulnerabilities:** Hyperledger Fabric relies on smart contracts for executing transactions, and like any code, these contracts can contain vulnerabilities that can be exploited by attackers. This can lead to loss of funds or data leaks.

**Limited scalability:** Hyperledger Fabric's scalability is limited by its consensus mechanism, which requires all nodes to validate each transaction. This can lead to network congestion and slow transaction processing times, which can limit its use in large-scale applications

Overall, while Hyperledger Fabric is designed to provide enterprise-grade security features, it still has its own set of disadvantages and limitations. Organizations should carefully evaluate these disadvantages and consider whether Hyperledger Fabric is the best choice for their specific use case.

## 1.5. Proposed System with Features

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 ðA, uid, PkcomÞ 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 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.

Text

Description automatically generated

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

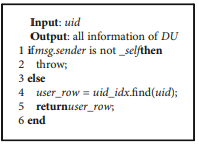
5.Authenticate: When UMContract receives action (UMContract, Authenticate, Auth, (from, method, account, id, args)), this function interface will be triggered to execute. It is used to authenticate the identity of a DU, which is invoked by another contract and returns the result to the invoker.

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

Let PK denote the system public parameters. Let two-tuple ) 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*.



Algorithm 2: GetUserByUid

Text

Description automatically generated with medium confidence

Algorithm 3: Apply.

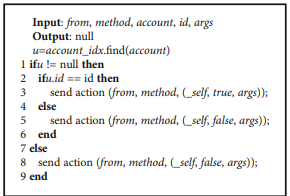
Text

Description automatically generated

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.



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.

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.

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.

**Advantages of proposed system**

## Data coordination& rapid deployment.

## Permissioned networks.

## Network size.

## Scalability and performance.

## 1.6. INPUT AND OUTPUT DESIGN

### INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

### OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

# 2 . LITERATURE SURVEY

* “**Blockchain thinking: the brain as a decentralized autonomous corporation [commentary]”**

**Author:** M. Swan

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.

* **“Decentralizing privacy: using blockchain to protect personal data”**

**Authors:** G. Zyskind, O. Nathan, and A. Pentland

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.

* **“Medrec: using blockchain for medical data access and permission management,”**

**Authors:** Azaria, A. Ekblaw, T. Vieira, and A. Lippman.

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”. This provided them with access to aggregate, anonymized data as mining rewards, in return for sustaining and securing the network via Proof of Work. MedRec thus enabled the emergence of data economics, supplying big data to empower researchers while engaging patients and providers in the choice to release metadata. The purpose of this short paper is to expose, prior to field tests, a working prototype through which we analyze and discuss our approach.

* **“Integrating blockchain for data sharing and collaboration in mobile healthcare applications,”**

**Authors:** X. Liang, J. Zhao, S. Shetty, J. Liu, and D. Li

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. The performance of MeDShare is comparable to current cutting-edge solutions to data sharing among cloud service providers. By implementing MeDShare, cloud service providers and other data guardians will be able to achieve data provenance and auditing while sharing medical data with entities such as research and medical institutions with minimal risk to data privacy.

* **“Blockchain based efficient privacy preserving and data sharing scheme of content-centric network in 5g”**

**Authors:** K. Fan, Y. Ren, Y. Wang, H. Li, and Y. Yang

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.

* **“Secure and trustable electronic medical records sharing using blockchain”**

**Authors:** Dubovitskaya, Z. Xu, S. Ryu, M. Schumacher, and F. Wang

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.

* **“Blockchain-based data sharing system for AI-powered network operations”**

**Authors:** G. Zhang, T. Li, Y. Li, P. Hui, and D. Jin

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.

* **“A blockchain-based file-sharing system for academic paper review”**

**Authors:** I. Zhou, I. Makhdoom, M. Abolhasan, J. Lipman, and N. Shariati

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. Also, the Hyperledger blockchain provided user access control to achieve double blindness in reviews. Along with the Hyperledger blockchain, the IPFS is used to store the paper documents, review documents and open metrics documents to reduce the storage requirement of the blockchain. A broad system architecture is constructed to combine the blockchain and the file storage system. This system architecture distributed the nodes of the system to related parties. Finally, the blockchain network is implemented and tested using the Hyperledger Composer Playground environment.

* **“A framework for secure and decentralized sharing of medical imaging data via blockchain consensus”**

**Author:** V. Patel

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. SOFTWARE REQUIREMENTS ANALYSIS**

## 3.1. Problem Specification

## We combined blockchain, ciphertext-policy attribute-based encryption (CP-ABE) and Inter Planetary File System (IPFS) to address the data sharing security 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.

**3.2. Functional Requirements**

**OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization, and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**OUTPUT DEFINITION**

The outputs should be defined in terms of the following points:

* Type of the output
* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**INPUT DESIGN**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT STAGES**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**INPUT TYPES**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**INPUT MEDIA**

At this stage choice has to be made about the input media.To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**ERROR AVOIDANCE**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**ERROR DETECTION**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**DATA VALIDATION**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**USER INTERFACE DESIGN**

It is essential to consult the system users and discuss their needs while designing the user interface:

**USER INTERFACE SYSTEMS CAN BE BROADLY CLASIFIED AS:**

* User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
* Computer initiated interfaces.

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**USER INITIATED INTERGFACES**

User initiated interfaces fall into two approximate classes:

* Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
* Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**COMPUTER-INITIATED INTERFACES**

The following computer – initiated interfaces were used:

The menu system for the user is presented with a list of alternatives and the user chooses one ;

of alternatives.

Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**ERROR MESSAGE DESIGN**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

## 3.3. Feasibility Study

When conducting a feasibility study for a blockchain-based security sharing scheme for personal data with fine-grained access control, several types of feasibility studies can be employed to assess different aspects of the project. The following are the main types of feasibility studies relevant to this scenario:

Technical Feasibility: This study assesses the technical viability of implementing the blockchain-based security sharing scheme. It examines whether the necessary technology infrastructure, hardware, software, and technical expertise are available or can be developed within reasonable constraints. Considerations include evaluating the scalability of the blockchain platform, the performance of the network, and the compatibility with existing systems and protocols.

Economic Feasibility: The economic feasibility study evaluates the financial viability of implementing the proposed scheme. It involves estimating the development costs, operational expenses, and potential returns on investment. Considerations include analyzing the cost of blockchain infrastructure, ongoing maintenance, data storage, and potential cost savings compared to existing systems or alternative solutions.

Legal and Regulatory Feasibility: This study examines the legal and regulatory aspects related to the implementation of the blockchain-based security sharing scheme. It assesses compliance with data protection laws, privacy regulations, and any other relevant legal requirements. Considerations include ensuring adherence to GDPR (General Data Protection Regulation) or other regional data protection regulations, obtaining necessary consents, and managing liability and accountability.

Operational Feasibility: Operational feasibility focuses on whether the proposed blockchain-based scheme can be implemented and integrated into the existing operational framework effectively. It examines the impact on existing processes, workflows, and the capability of users to adapt to the new system. Considerations include assessing the ease of use, training requirements, and potential disruption to business operations during the implementation phase.

Social Feasibility: Social feasibility evaluates the social and ethical implications of implementing the blockchain-based security sharing scheme. It considers the impact on individuals' privacy, consent management, and the level of control users have over their personal data.

Each feasibility study type provides a specific perspective on the viability of the blockchain-based security sharing scheme and helps identify potential challenges and risks. By conducting these studies, stakeholders can make informed decisions about the project's implementation and determine the best course of action.

**3.4. Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system.

This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system
* The existing system is completely dependent on the user to perform all the duties.

# 4. SOFTWARE AND HARDWARE REQUIREMENTS

## 4.1. Software Requirements

**Software Requirements**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

* Python IDLE 3.7 version (or)
* Anaconda 3.7 (or)
* Jupiter (or)
* Google colab

## 4.2. Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

* Operating system : Windows
* Processor : Minimum intel i3
* RAM : Minimum 8 GB
* Hard disk : Minimum 250GB

# 5. SOFTWARE DESIGN

## 5.1. System Architecture

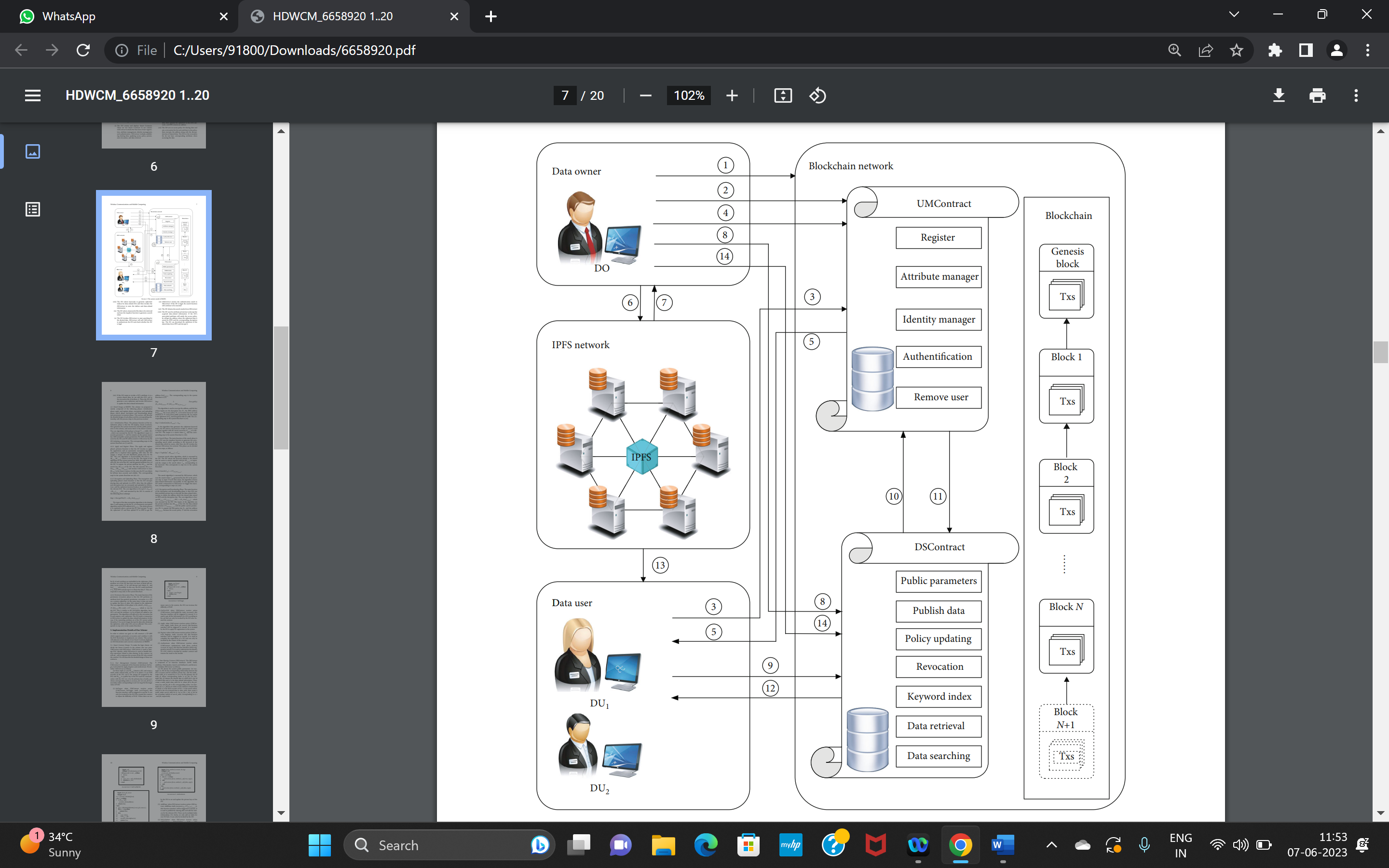


Fig.5.1. System Architecture

1.The DO creates and deploys Smart Contracts. There are two Smart Contracts in our scheme. UMContract includes the functions of user registration, attribute management, identity management, and authentication. DSContract includes publishing sharing data, updating access policy, permission revocation, and data retrieval.

2.The DO generates the system master key and system public key locally and stores the system public key in DSContract.

3.The DU invokes UMContract to apply for registration, and he needs to provide his account of EOS and a public key. The public key is used to communicate with the DO, and the DO uses it to encrypt the message and broadcasts the ciphertext to the blockchain. Only the corresponding DU can decrypt the ciphertext and obtain the message.

4. The DO assigns a unique uid to each DU who applies for, and generates a private attribute key and a secret search key for the DU. After encrypting these two keys with the DU’s communication public key, the DO will save them in the Smart Contract together with the uid.

5.The DU obtains the ciphertext information of the keys and decrypts them with his private communication key.

6.The DO randomly selects a key of the symmetric encryption algorithm, uses it to encrypt the sharing data, then uploads the ciphertext to the IPFS network, and IPFS returns an address.

7.The DO sets an access policy for sharing data and sets a revocation list for each attribute in the policy, then encrypts the address along with the decryption key of shared data. The DUs in the revocation list do not have corresponding attributes when accessing the data.

8.The DO selects keywords to generate ciphertext indices for data-related DUs and then invokes the DSContract to store the indices and data-related information.

9.The DU selects a keyword of the data to be retrieved and uses the trapdoor function to generate a search token.

10.The DU invokes DSContract to start searching for the desired data. DSContract will call UMContract to authenticate the DU and check whether the DU is legal.

11.UMContract returns the authentication result to DSContract. If the DU is legal, the search function will continue to be executed.

12.The DU obtains the search results from DSContract.

13.The DU uses his attribute private key to decrypt the acquired data-related information. If the DU’s unrevoked attributes still satisfy the access policy, he will get the address where the ciphertext data is stored on IPFS and the corresponding decryption key. The DU can download the ciphertext of the shared data from IPFS and decrypt it.

14.If the DO wants to revoke a DU’s attribute A to a certain shared data, he can add this DU’s uid to the revocation list of attribute A. Then, the DO will generate a new ciphertext and invoke DSContract to update the data-related information.

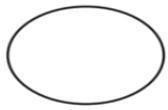
#### 5.2. Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart. The Basic Notation used to create a DFD’s are as follows:

1. Dataflow: Data move in a specific direction from an origin to a destination.



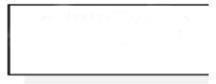
1. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.



1. Source: External sources or destination of data, which may be People, programs.



1. Data Store: Here data are stored or reference by a process in the System.



Diagram

Description automatically generated

**Fig 5.2**: Data flow diagram

## 

## 5.3. UML Diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

The two broadest categories that encompass all other types are:

* Behavioral UML diagram and
* Structural UML diagram.

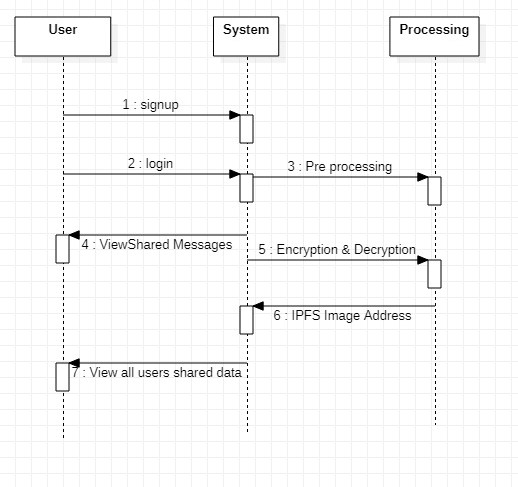
As the name suggests, some UML diagrams try to analyses and depict the structure of a system or process,whereas other describe the behavior of the system, its actors, and its building components.

The different types are broken down as follows:

1. Sequence diagram
2. Use case Diagram
3. Class diagram

### a. Sequence Diagram

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.



**Fig 5.3.1:** Sequence diagram

**List of actions**

### User:

User need to press any of the given three (i.e. upload dataset, accuracy, prediction) then he will get the output accordingly.

### System:

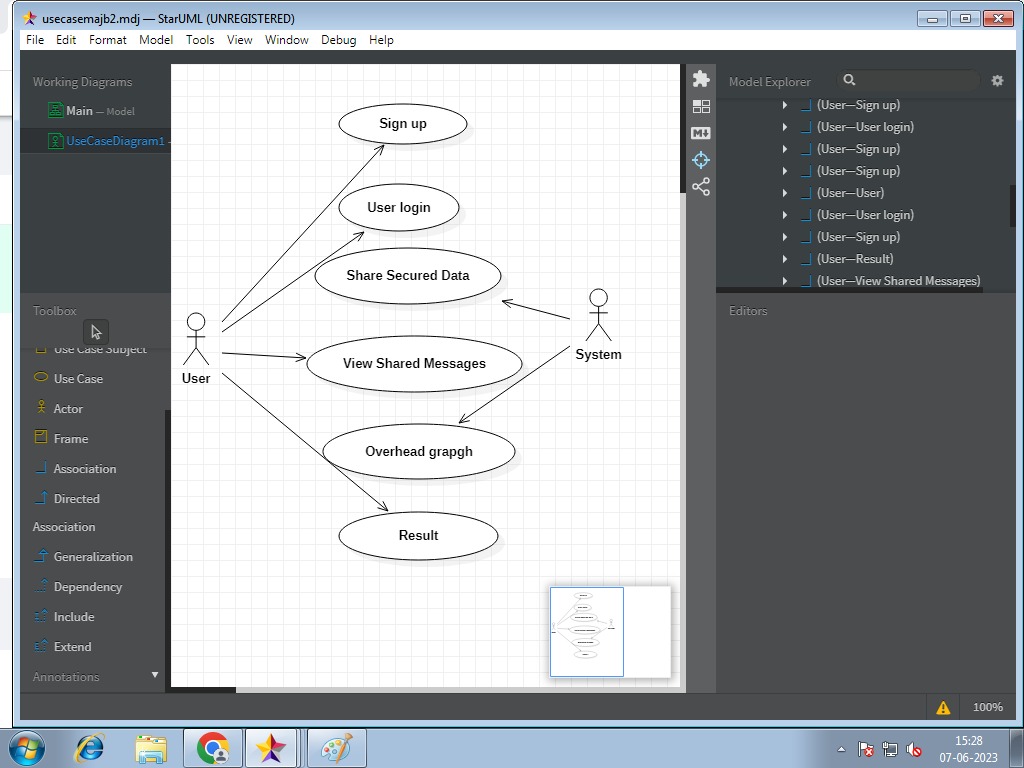
System will give the output as he enters according to the given data.

### Result:

As per user enters the data it will gives the suitable jobs.

**b. Use Case Diagram**

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use case in which the user is involved. A use case diagram is used to structure of the behavior thing in a model. The use cases are represented by either circles or ellipses.

****

\

**Fig 5.3.2:** Use case diagram

### Actor:

* User
* System

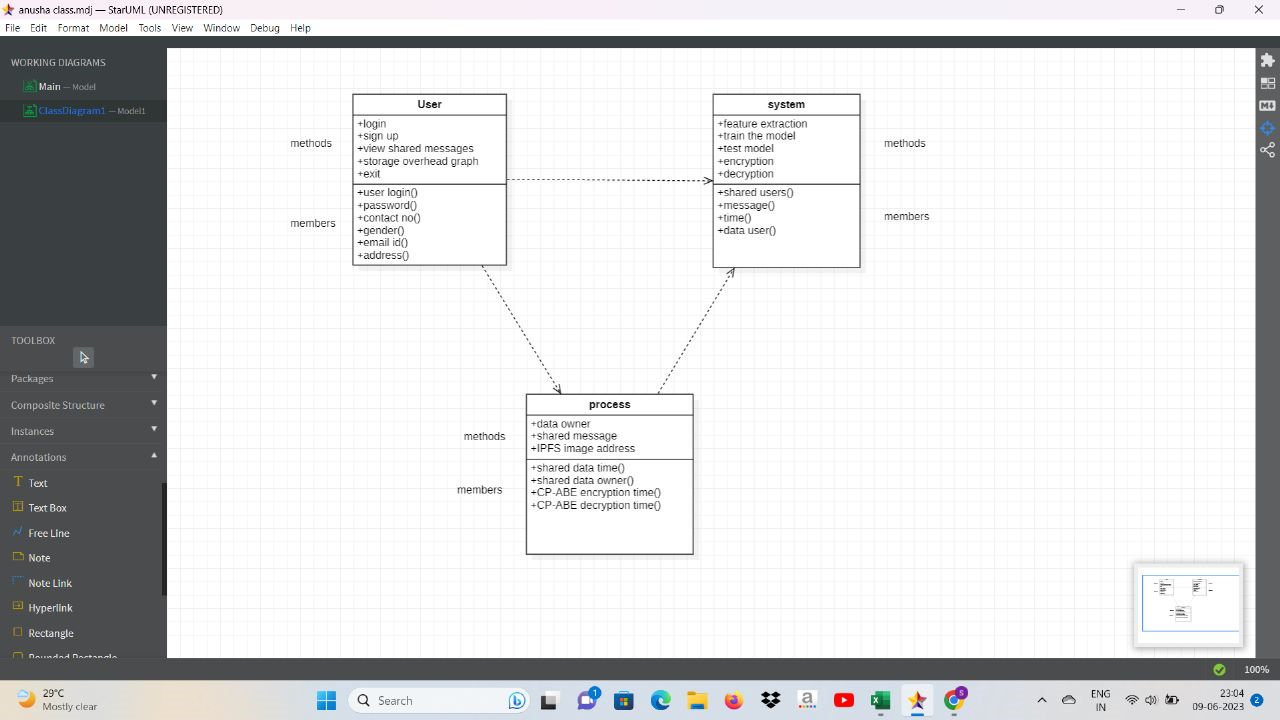
### Use case:

* User need to give the data. Then the System will give the results.
* System Consist of data set that it will preprocess the data and the split the data

and apply the train and test the data then it will predict the result.

### c. Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



**Fig 5.3.3:** Class Diagram

# 

# 6. CODING AND ITS IMPLEMENTATION

## 6.1. Source Code

from django.shortcuts import render

from django.template import RequestContext

from django.contrib import messages

from django.http import HttpResponse

from django.contrib import admin

from django.apps import AppConfig

import datetime

import ipfsApi

import os

import json

from web3 import Web3, HTTPProvider

from django.core.files.storage import FileSystemStorage

import pickle

from ecies.utils import generate\_eth\_key, generate\_key

from ecies import encrypt, decrypt

import time

import matplotlib.pyplot as plt

import numpy as np

api = ipfsApi.Client(host='http://127.0.0.1', port=5001)

global details, username

global enc\_time, dec\_time

#function to generate public and private keys for CP-ABE algorithm

def CPABEgenerateKeys():

if os.path.exists("pvt.key"):

with open("pvt.key", 'rb') as f:

private\_key = f.read()

f.close()

with open("pri.key", 'rb') as f:

public\_key = f.read()

f.close()

private\_key = private\_key.decode()

public\_key = public\_key.decode()

else:

secret\_key = generate\_eth\_key()

private\_key = secret\_key.to\_hex() # hex string

public\_key = secret\_key.public\_key.to\_hex()

with open("pvt.key", 'wb') as f:

f.write(private\_key.encode())

f.close()

with open("pri.key", 'wb') as f:

f.write(public\_key.encode())

f.close()

return private\_key, public\_key

#CP-ABE will encrypt data using plain text adn public key

def CPABEEncrypt(plainText, public\_key):

cpabe\_encrypt = encrypt(public\_key, plainText)

return cpabe\_encrypt

#CP-ABE will decrypt data using private key and encrypted text

def CPABEDecrypt(encrypt, private\_key):

cpabe\_decrypt = decrypt(private\_key, encrypt)

return cpabe\_decrypt

def readDetails(contract\_type):

global details

details = ""

print(contract\_type+"======================")

blockchain\_address = 'http://127.0.0.1:9545' #Blokchain connection IP

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'BlockchainSecureSharing.json' #Blockchain Secure Shared Data contract code

deployed\_contract\_address = '0xB67e5756062a767e8BfC30F8ca3732ca196c5E24' #hash address to access Shared Data contract

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract = web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi) #now calling contract to access data

if contract\_type == 'signup':

details = contract.functions.getSignup().call()

if contract\_type == 'attribute':

details = contract.functions.getAttribute().call()

print(details)

def saveDataBlockChain(currentData, contract\_type):

global details

global contract

details = ""

blockchain\_address = 'http://127.0.0.1:9545'

web3 = Web3(HTTPProvider(blockchain\_address))

web3.eth.defaultAccount = web3.eth.accounts[0]

compiled\_contract\_path = 'BlockchainSecureSharing.json' #Blockchain contract file

deployed\_contract\_address = '0xB67e5756062a767e8BfC30F8ca3732ca196c5E24' #contract address

with open(compiled\_contract\_path) as file:

contract\_json = json.load(file) # load contract info as JSON

contract\_abi = contract\_json['abi'] # fetch contract's abi - necessary to call its functions

file.close()

contract = web3.eth.contract(address=deployed\_contract\_address, abi=contract\_abi)

readDetails(contract\_type)

if contract\_type == 'signup':

details+=currentData

msg = contract.functions.setSignup(details).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

if contract\_type == 'attribute':

details+=currentData

msg = contract.functions.setAttribute(details).transact()

tx\_receipt = web3.eth.waitForTransactionReceipt(msg)

def index(request):

if request.method == 'GET':

return render(request, 'index.html', {})

def Login(request):

if request.method == 'GET':

return render(request, 'Login.html', {})

def Signup(request):

if request.method == 'GET':

return render(request, 'Signup.html', {})

def SharedData(request):

if request.method == 'GET':

global username

readDetails('signup')

arr = details.split("\n")

status = '<tr><td><font size="" color="black">Choose&nbsp;Shared&nbsp;Users</b></td><td><select name="t3" multiple>'

for i in range(len(arr)-1):

array = arr[i].split("#")

if array[1] != username:

status += '<option value="'+array[1]+'">'+array[1]+'</option>'

status += "</select></td></tr>"

context= {'data1':status}

return render(request, 'SharedData.html', context)

def ViewSharedMessages(request):

if request.method == 'GET':

global enc\_time, dec\_time, username

dec\_time = 0

strdata = '<table border=1 align=center width=100%><tr><th><font size="" color="black">Data Owner</th><th><font size="" color="black">Shared Message</th>'

strdata+='<th><font size="" color="black">IPFS Image Address</th><th><font size="" color="black">Shared Image</th>'

strdata+='<th><font size="" color="black">Shared Date Time</th><th><font size="" color="black">Shared Data Users</th></tr>'

for root, dirs, directory in os.walk('static/tweetimages'):

for j in range(len(directory)):

os.remove('static/tweetimages/'+directory[j])

readDetails('attribute')

arr = details.split("\n")

start\_times = time.time()

for i in range(len(arr)-1):

array = arr[i].split("#")

share\_user = array[6].split(",")

if array[0] == 'post' and (username in share\_user or username == array[1]):

content = api.get\_pyobj(array[3])

private\_key, public\_key = CPABEgenerateKeys()

decrypted = CPABEDecrypt(content, private\_key)

content = pickle.loads(decrypted)

with open("BlockchainSecurityApp/static/shareimages/"+array[5], "wb") as file:

file.write(content)

file.close()

strdata+='<tr><td><font size="" color="black">'+str(array[1])+'</td><td><font size=""

color="black">'+array[2]+'</td><td><font size="" color="black">'+str(array[3])+'</td>'

strdata+='<td><img src=static/shareimages/'+array[5]+' width=200 height=200></img></td>'

strdata+='<td><font size="" color="black">'+str(array[4])+'</td>'

strdata+='<td><font size="" color="black">'+str(array[6])+'</td>'

end\_times = time.time()

dec\_time = end\_times - start\_times

context= {'data':strdata}

return render(request, 'ViewSharedMessages.html', context)

def LoginAction(request):

if request.method == 'POST':

global username

username = request.POST.get('t1', False)

password = request.POST.get('t2', False)

readDetails('signup')

arr = details.split("\n")

status = "none"

for i in range(len(arr)-1):

array = arr[i].split("#")

if array[1] == username and password == array[2]:

status = "Welcome "+username

break

if status != 'none':

file = open('session.txt','w')

file.write(username)

file.close()

context= {'data':status}

return render(request, 'UserScreen.html', context)

else:

context= {'data':'login failed'}

return render(request, 'Login.html', context)

def SharedDataAction(request):

if request.method == 'POST':

global enc\_time, username

post\_message = request.POST.get('t1', False)

share = request.POST.getlist('t3')

share = ','.join(share)

filename = request.FILES['t2'].name

start = time.time()

myfile = request.FILES['t2'].read()

myfile = pickle.dumps(myfile)

private\_key, public\_key = CPABEgenerateKeys()

cpabe\_encrypt = CPABEEncrypt(myfile, public\_key)

now = datetime.datetime.now()

current\_time = now.strftime("%Y-%m-%d %H:%M:%S")

user = username

hashcode = api.add\_pyobj(cpabe\_encrypt)

data =

"post#"+user+"#"+post\_message+"#"+str(hashcode)+"#"+str(current\_time)+"#"+filename+"#"+share+"\n"

end = time.time()

enc\_time = end - start

saveDataBlockChain(data,"attribute")

output = 'Shared Data saved in Blockchain with below hashcodes & Image file saved in IPFS.<br/>'+str(hashcode)

context= {'data':output}

return render(request, 'SharedData.html', context)

def SignupAction(request):

if request.method == 'POST':

global details

username = request.POST.get('t1', False)

password = request.POST.get('t2', False)

contact = request.POST.get('t3', False)

gender = request.POST.get('t4', False)

email = request.POST.get('t5', False)

address = request.POST.get('t6', False)

output = "Username already exists"

readDetails('signup')

arr = details.split("\n")

status = "none"

for i in range(len(arr)-1):

array = arr[i].split("#")

if array[1] == username:

status = username+" already exists"

break

if status == "none":

details = ""

data = "signup#"+username+"#"+password+"#"+contact+"#"+gender+"#"+email+"#"+address+"\n"

saveDataBlockChain(data,"signup")

context = {"data":"Signup process completed and record saved in Blockchain"}

return render(request, 'Signup.html', context)

else:

context = {"data":status}

return render(request, 'Signup.html', context)

def Graph(request):

if request.method == 'GET':

global username

global enc\_time, dec\_time

height = [enc\_time, dec\_time]

bars = ('CP-ABE Encryption Time', 'CP-ABE Decryption Time')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.title("Uploading, Encryption & Decryption Overhead Graph")

plt.show()

context = {"data":"Welcome "+username}

return render(request, 'UserScreen.html', context)

## 

## 6.2. Implementation

**Python Development Steps**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.

Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

Print is now a function.

* Views and iterators instead of lists
* The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
* There is only one integer type left, i.e., int. long is int as well.

The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.

* Text Vs. Data Instead of Unicode Vs. 8-bit

**Purpose**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**History of Python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

### Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about

how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors.

This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All

its tools have been quick to implement, saved a lot of time, and several of them have later been

patched and updated by people with no Python background - without breaking.

**Modules Used in Project**

1. **TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

1. **NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

A powerful N-dimensional array object

Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

1. **Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and

preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

1. **Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code.

For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**e. Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

**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.

**Advantages of Python**

Let’s see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Install Python Step-by-Step in Windows and Mac

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but

this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

A screenshot of a computer

Description automatically generated with medium confidence

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Graphical user interface, application

Description automatically generated

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Graphical user interface, application

Description automatically generated

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Graphical user interface, text

Description automatically generated

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

**Installation of Python**

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

Graphical user interface, text, application

Description automatically generated

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Graphical user interface, text, application, chat or text message

Description automatically generatedStep 3: Click on Install NOW After the installation is successful. Click on Close.

Graphical user interface, text, application, chat or text message

Description automatically generated

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Graphical user interface, application

Description automatically generated

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

A screenshot of a computer

Description automatically generated with medium confidence

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

Application

Description automatically generated with low confidence

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

Graphical user interface, text, application, email

Description automatically generated

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

Graphical user interface, text, application, email

Description automatically generated

You will see that the command given is launched. With this, we end our tutorial on how to install

Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

**Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

2. Weak in Mobile Computing and Browsers

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

3. Design Restrictions

As you know, Python is dynamically typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

4. Underdeveloped Database Access Layers

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

5. Simple

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

**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.

**Applications of Blockchain**

Blockchain technology has a wide range of potential applications across various industries. Some examples of how blockchain is currently being used, or has the potential to be used, include:

Cryptocurrencies: Blockchain technology is the foundation of cryptocurrencies like Bitcoin and Ethereum, which use blockchain to enable peer-to-peer transactions without the need for a centralized intermediary.

Supply chain management: Blockchain technology can be used to create transparent and secure supply chain systems, allowing participants to track and verify the origin and authenticity of products.

Identity verification: Blockchain technology can be used to create secure and tamper-proof digital identity systems, allowing individuals to prove their identity without the need for a centralized authority.

Voting systems: Blockchain technology can be used to create secure and transparent voting systems, ensuring the accuracy and legitimacy of election results.

Healthcare: Blockchain technology can be used to create secure and transparent healthcare systems, enabling secure sharing of patient data and facilitating drug traceability.

Finance: Blockchain technology can be used to create more efficient and secure financial systems, allowing for faster and cheaper transactions while reducing the risk of fraud and corruption.

Real estate: Blockchain technology can be used to create more transparent and secure real estate transactions, allowing for faster and more efficient transfer of ownership.

These are just a few examples of how blockchain technology is being used, and there are many other potential applications that are currently being explored.

**Advantages of Block Chain**

Here are several advantages to using blockchain technology:

**Decentralization:** The decentralized nature of blockchain technology means that it is not controlled by any single entity, which increases transparency and security.

**Immutability:** Once data has been recorded on a blockchain, it cannot be altered or deleted, which ensures that it is tamper-proof and provides a high degree of data integrity.

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

**Transparency:** Blockchain technology provides a high degree of transparency, as all participants in the network have access to the same information, making it easier to verify and track transactions.

**Efficiency:** Blockchain technology can reduce the need for intermediaries in transactions, reducing the time and cost associated with processing and verifying transactions.

**Trust:** The security and transparency provided by blockchain technology can increase trust among participants in a network, leading to more efficient and secure transactions.

Overall, the advantages of blockchain technology make it a promising technology for a wide range of applications in various industries.

To store data in Blockchain we need to build SMART CONTRACT using solidity code and this contract will contains function to store user registration and attribute management for data sharing. Below code showing smart contact code

## 

## Fig.6.2.1.smart contract

In above smart contract we have functions to store and get user registration data and other details. To deploy above contract in Blockchain we need to follow below steps.

First go inside ‘hello-eth/node-modules/.bin’ folder and then double click on ‘runBlockchain.bat’ file to get below screen.

## Text Description automatically generated

Fig.6.2.2.Private keys generation

In above screen Blockchain Ethereum generated some default accounts and private keys and now type ‘truffle migrate’ command and press enter key to deploy contract in Blockchain and get below output

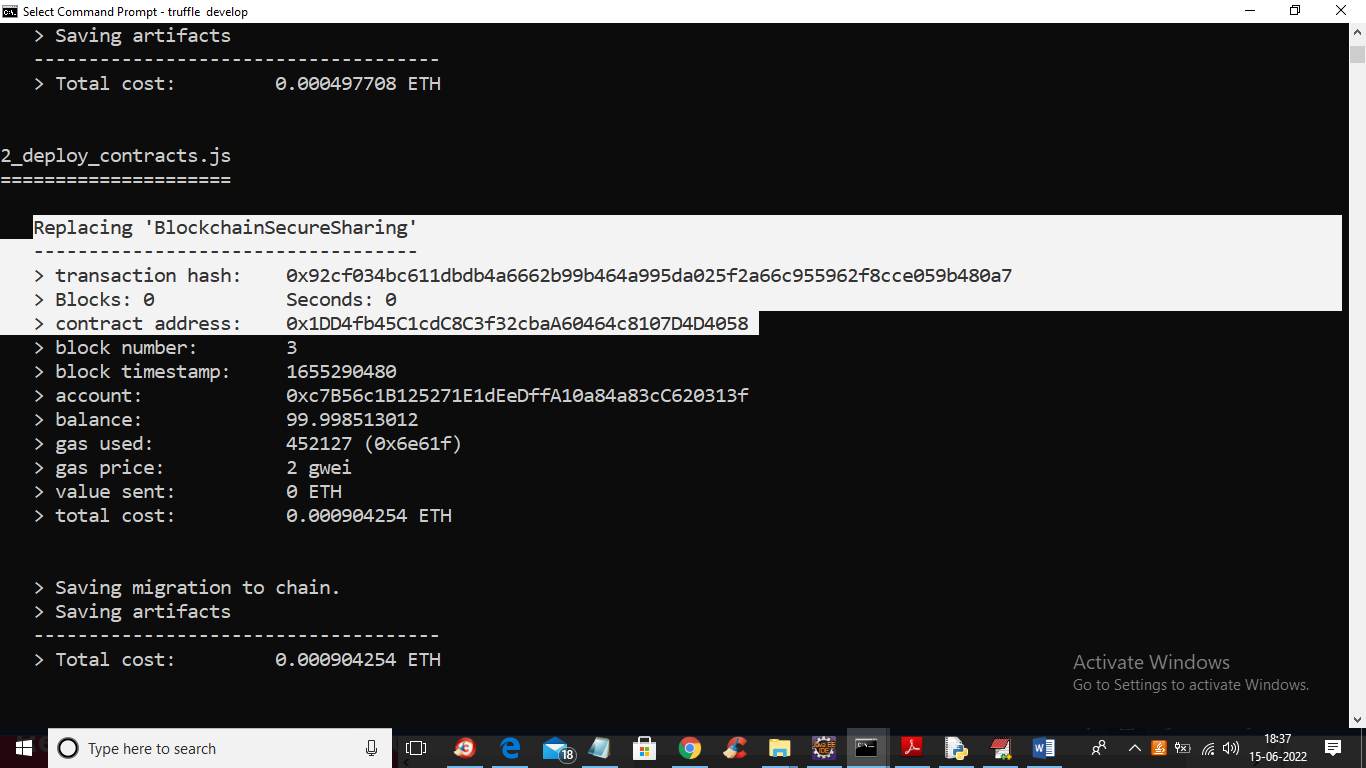


Fig.6.2.3.deployment of Blockchain

In above screen in white colour text we can see ‘BlockchainSecureSharing’ contract deployed and we got contract address also and this address we can specify in python code to store called Smart

Contract functions to store or retrieve data. Let above Blockchain console running and in below python code screen you can see we are calling contract by specifying its address.

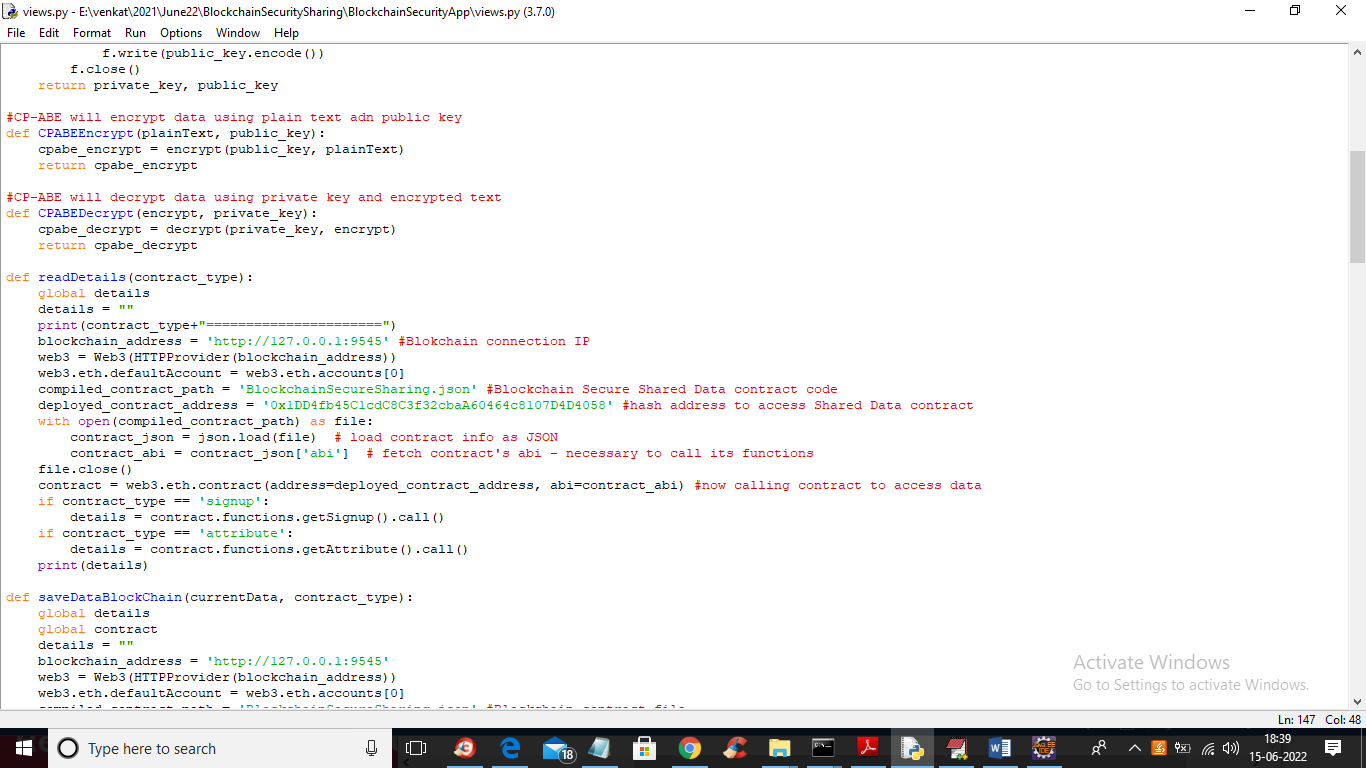


Fig.6.2.4.Smart contract functions

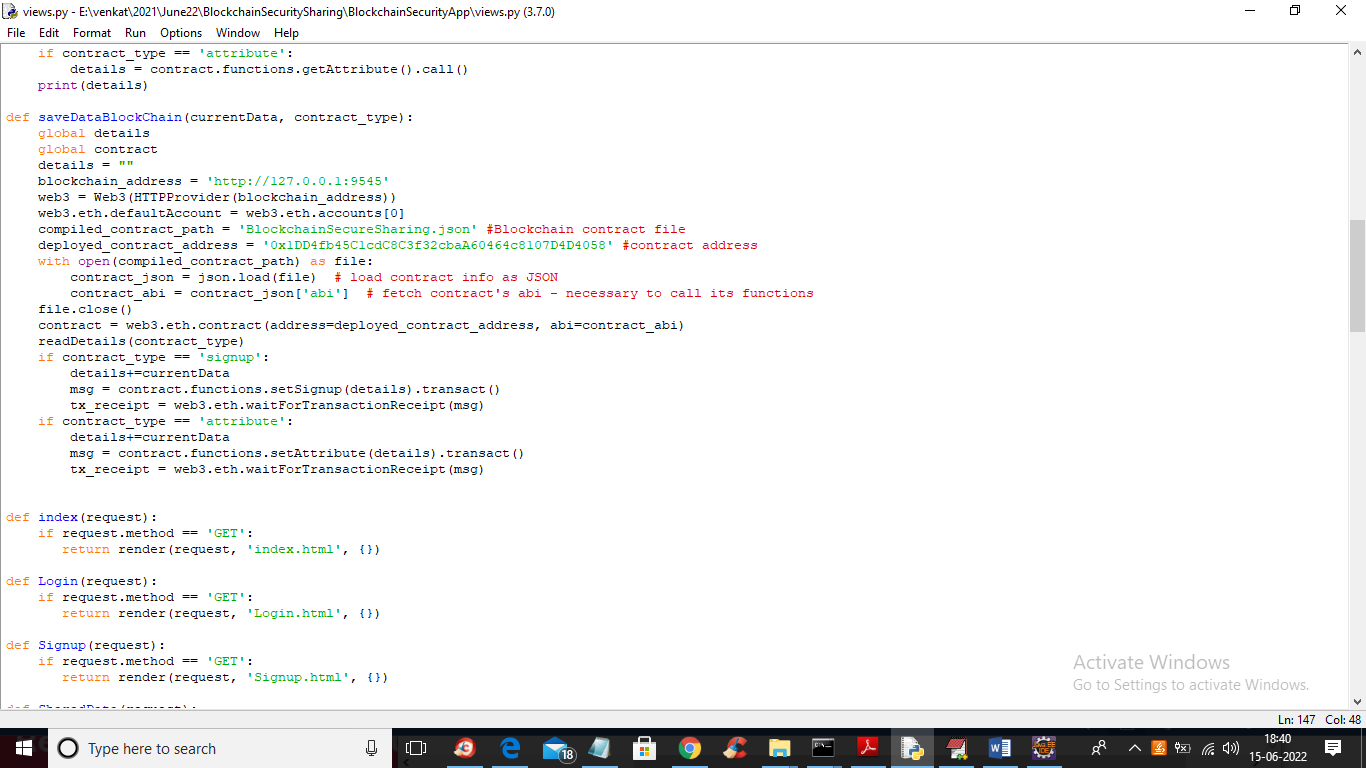


Fig.6.2.5.Smart contract functions

In above two screen read red colour comments to know about contract calling using python code.

To implement this project we have designed following modules

1. New User Signup: using this module user can get signup with the application and all this details saved in Blockchain.
2. User Login: using this module user can login to application.
3. Share Secured Data: using this module data owner can upload messages and images and then select sharing usernames and then encrypt data and then encrypt decryption keys by using sharing user details and only those users can decrypt data who has their names in decryption keys.
4. View Shared Messages: using this modules data users can view all messages shared by data owner and this message will display and get decrypted if user has access permission in decryption keys.
5. Storage Overhead Graph: using this module we will display encryption and decryption time overhead graph.

# 7. SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

### 7.1. Test cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **TEST CASES** | **EXPECTED RESULT** | **RESULT** | **REMARKS**  **(IF FAILS)** |
| 1. | Initializing SMART CONTRACT | Uploading the user registration data | PASS | If data file is not available, project process will not respond |
| 2. | Deploy contract in Blockchain | Ethereum and private keys generated | PASS | Contract is not deployed in Blockchain |
| 3. | Start IPFS Server | IPFS Server is started and the user details are stored here | PASS | Data storing is not possible |
| 4. | RUN server | Python DJANGO server is started | PASS | If the server is not started ,project process will not respond |
| 5. | New User singup | New user is added to blockchain | PASS | No data is added to blockchain |
| 6. | User login | User logged in successfully | PASS | User login failed |
| 7. | Shared Secured Data | Secure messages and images are uploaded | PASS | Data is not shared |
| 8. | View shared messages | Sent and received messages are visible along with IPFS address | PASS | Messages that are sent are not visible and IPFS address is not generated |

**Table.7.1:**Test cases

**Test Results:**

All the test cases mentioned above passed successfully. No defects encountered.

**8. OUTPUT SCREEN**

To run project first double click on ‘Start\_IPFS.bat’ file to start IPFS server and get below screen

Text

Description automatically generated

Fig.8.1.IPFS server

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

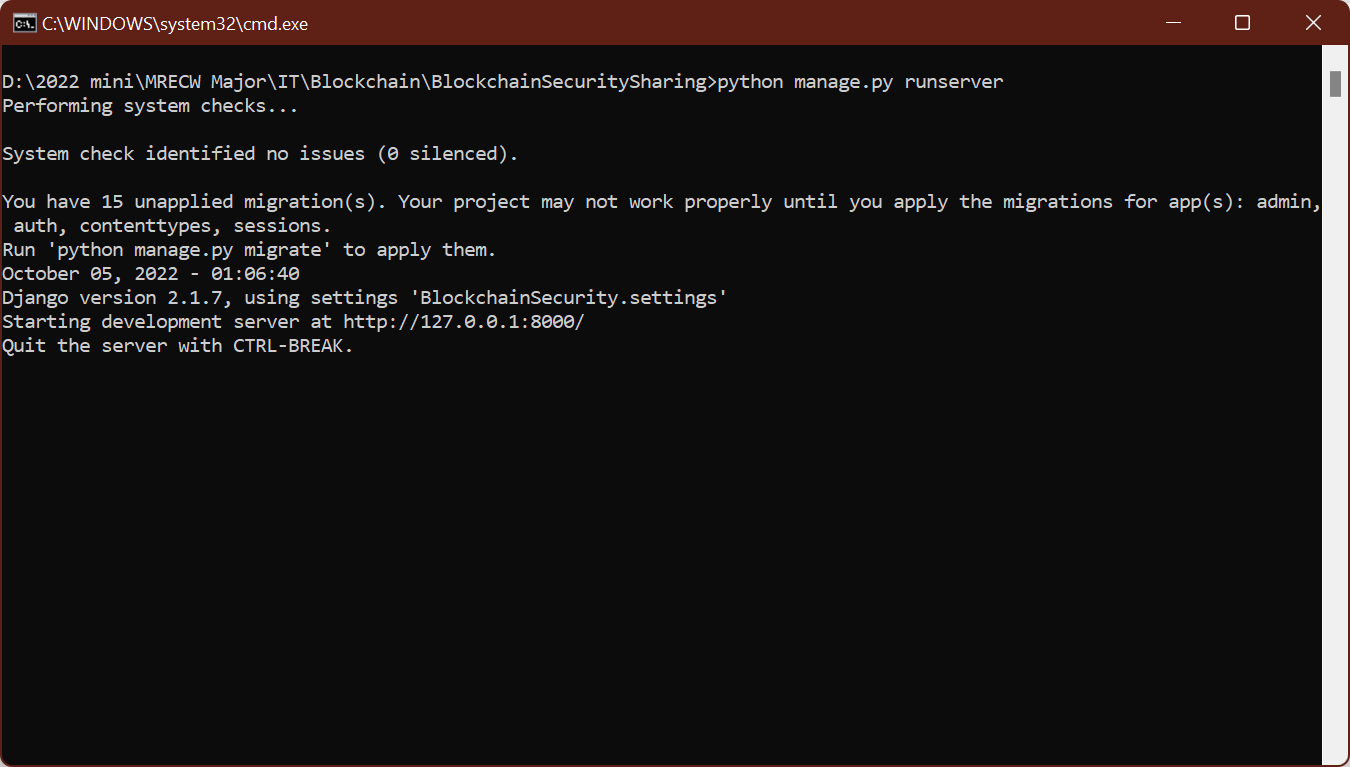


Fig.8.2.DJANGO server

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

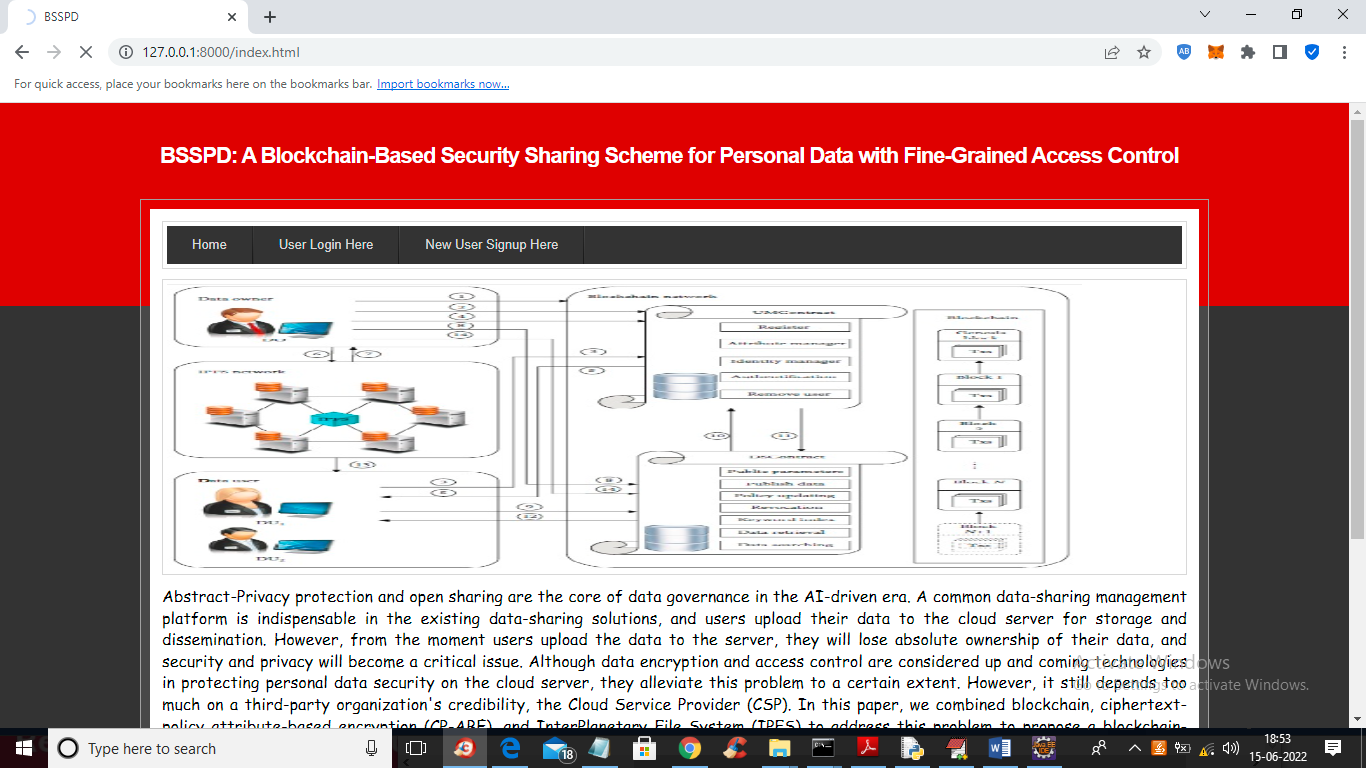


Fig.8.3.BSSPD Screen

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

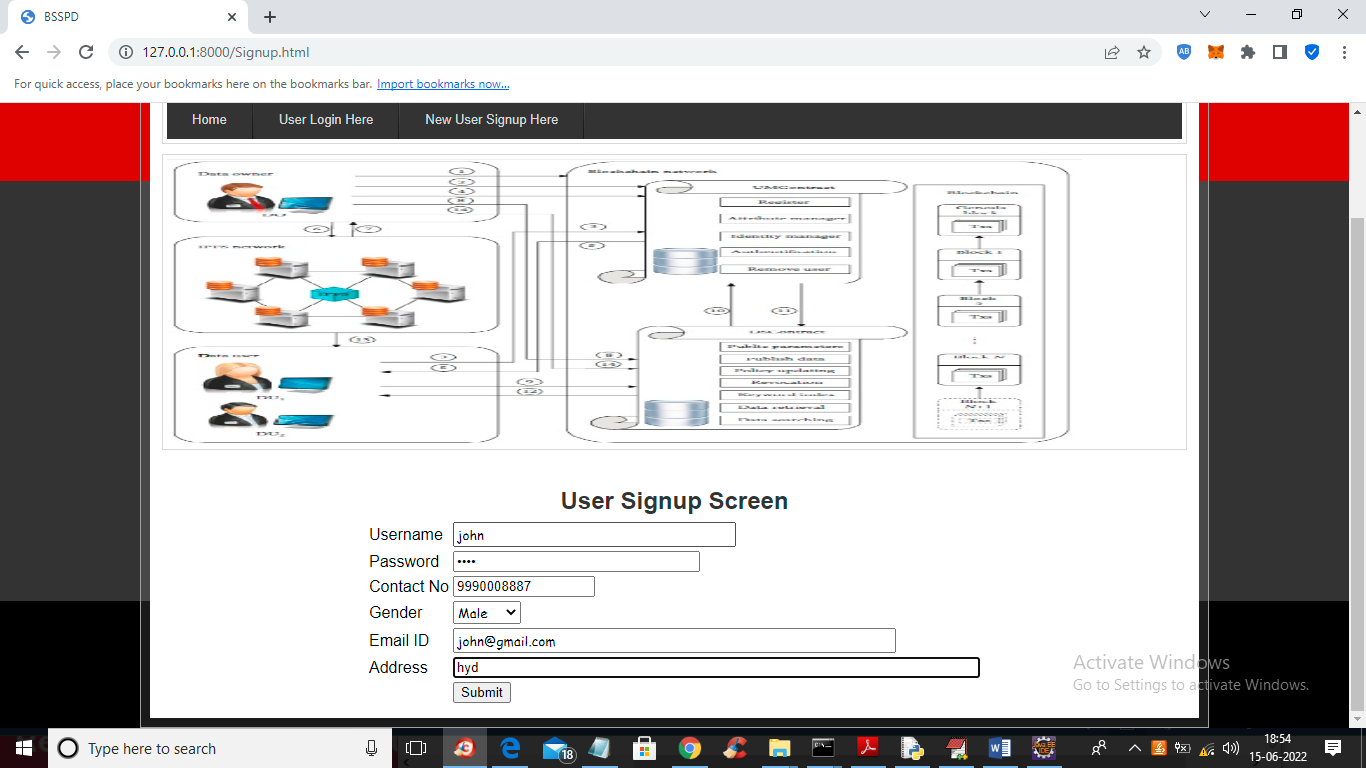


Fig.8.4.User Signup Screen

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

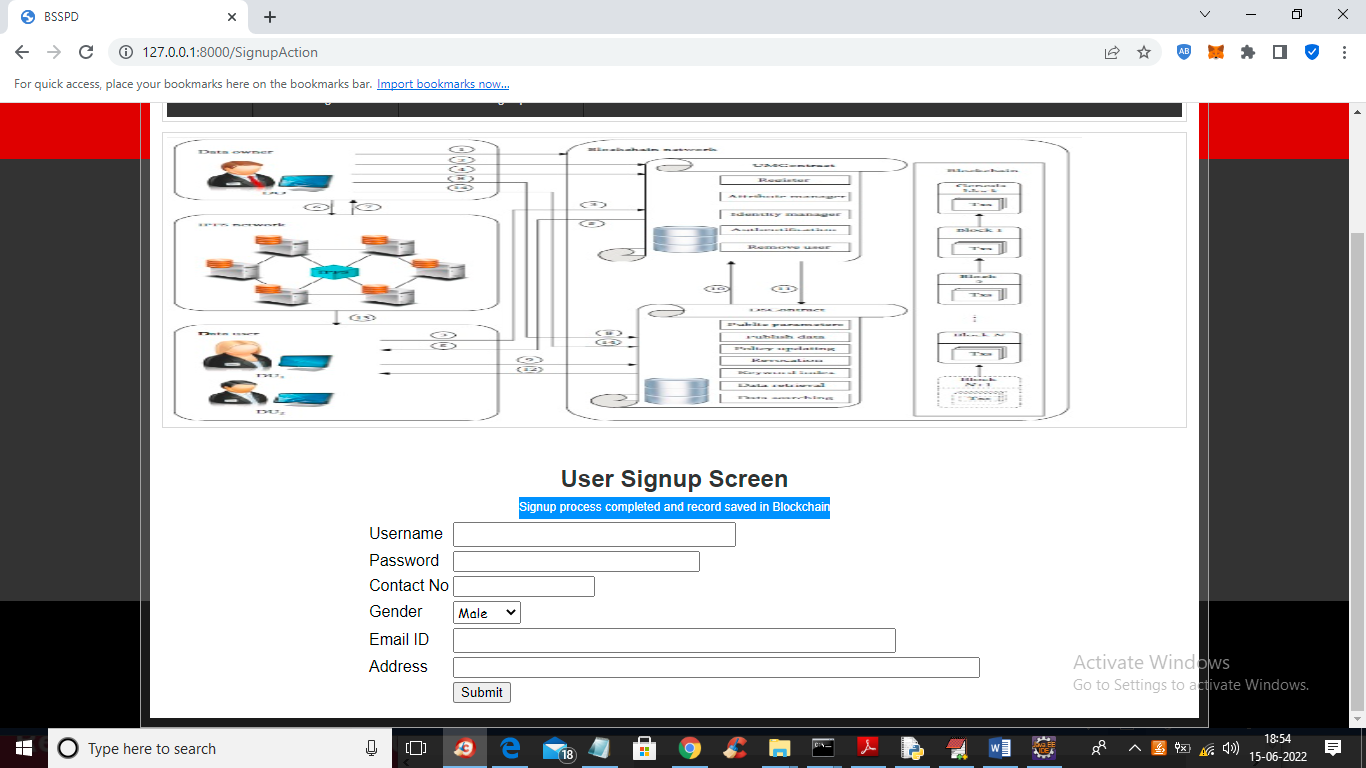


Fig.8.5.user signup process completed

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

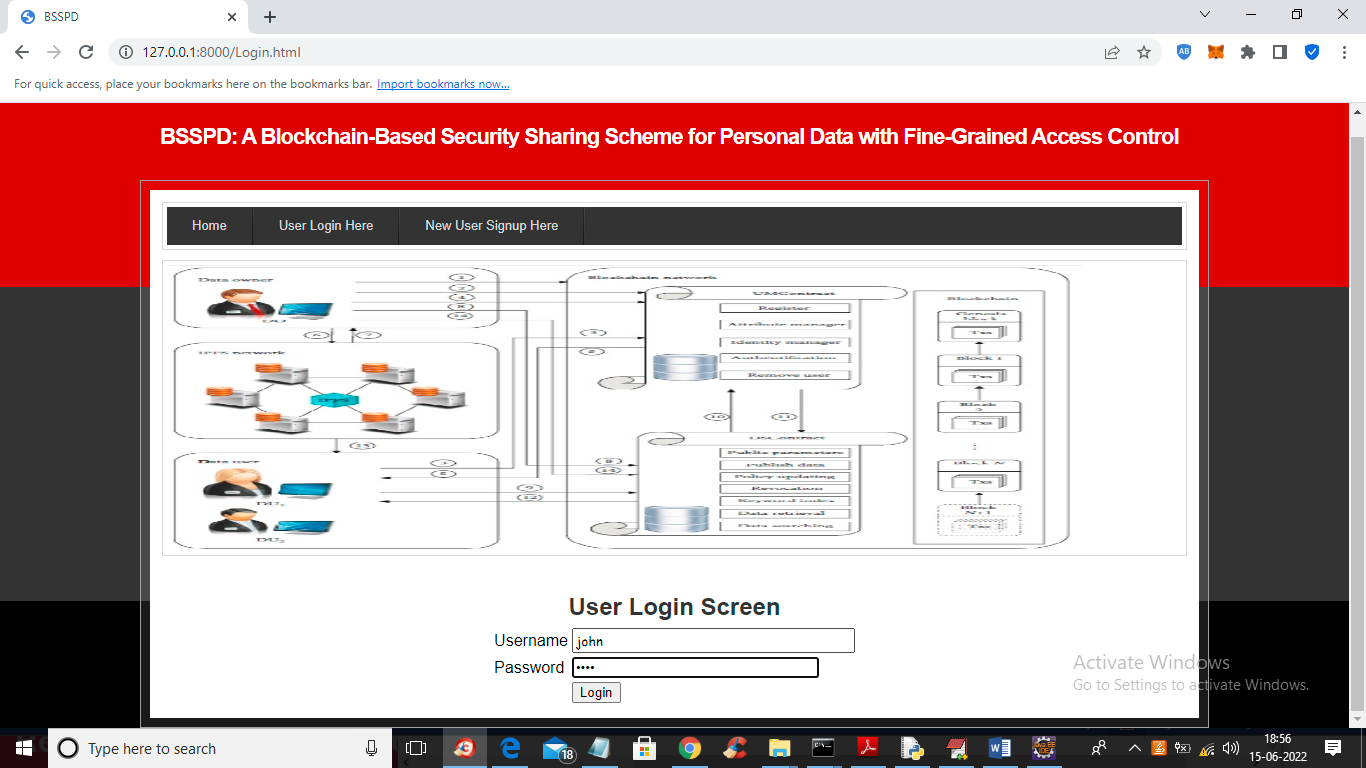


Fig.8.6.User Login Screen

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

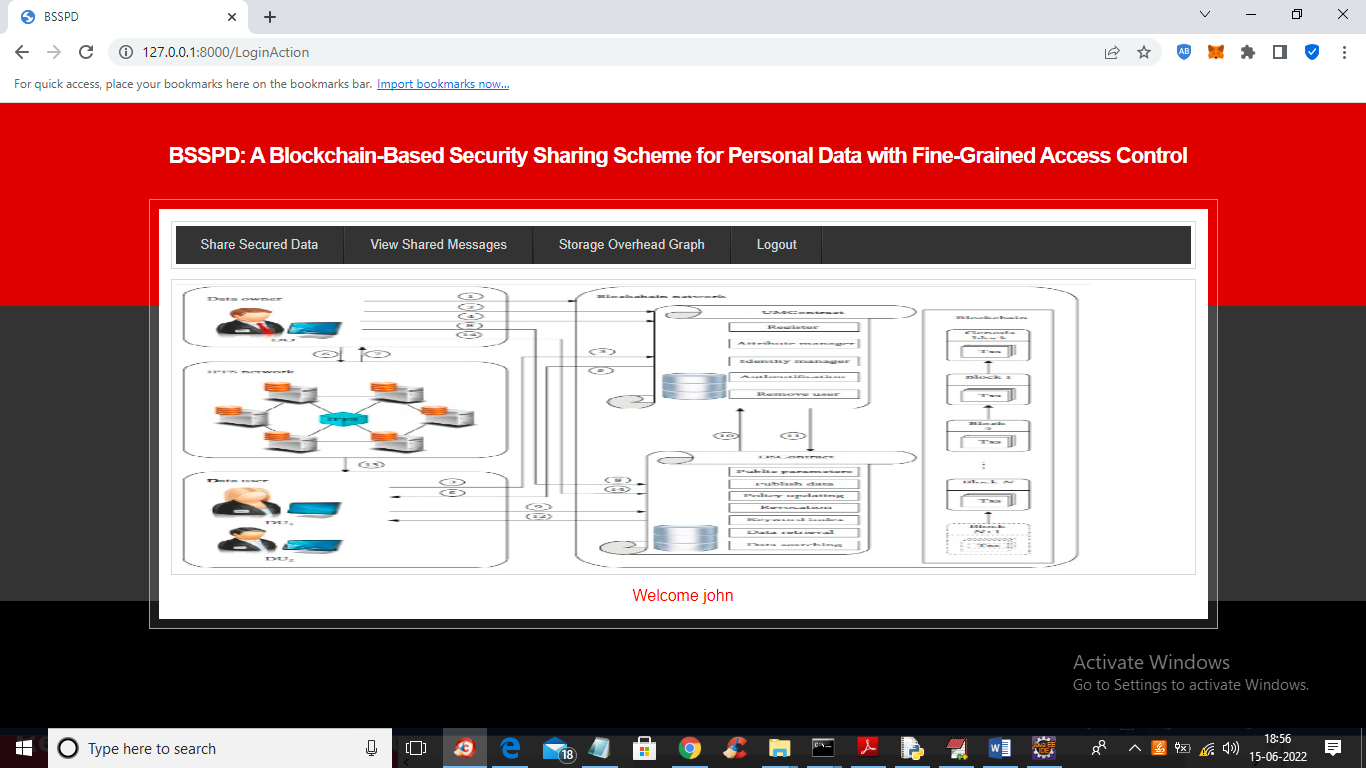


Fig.8.7.User Login Successful

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

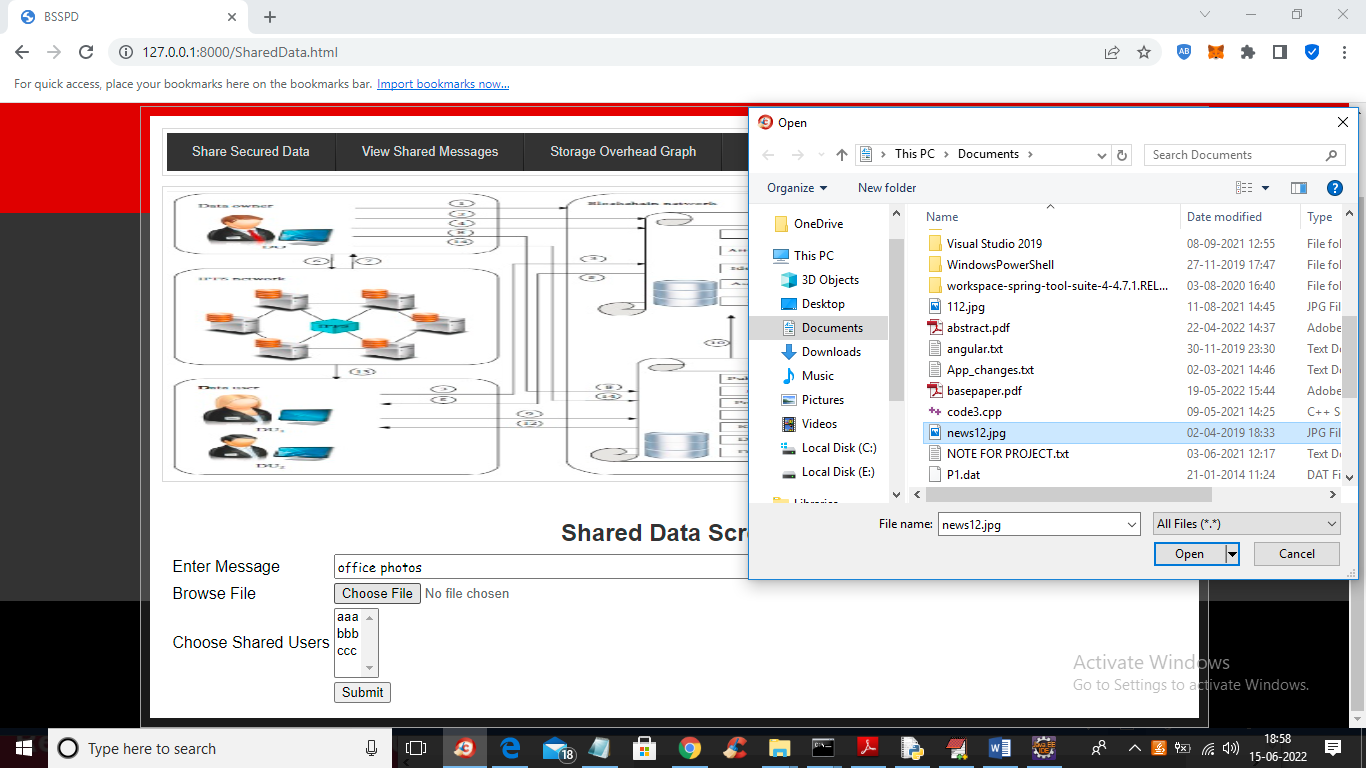


Fig.8.8.shared data screen

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

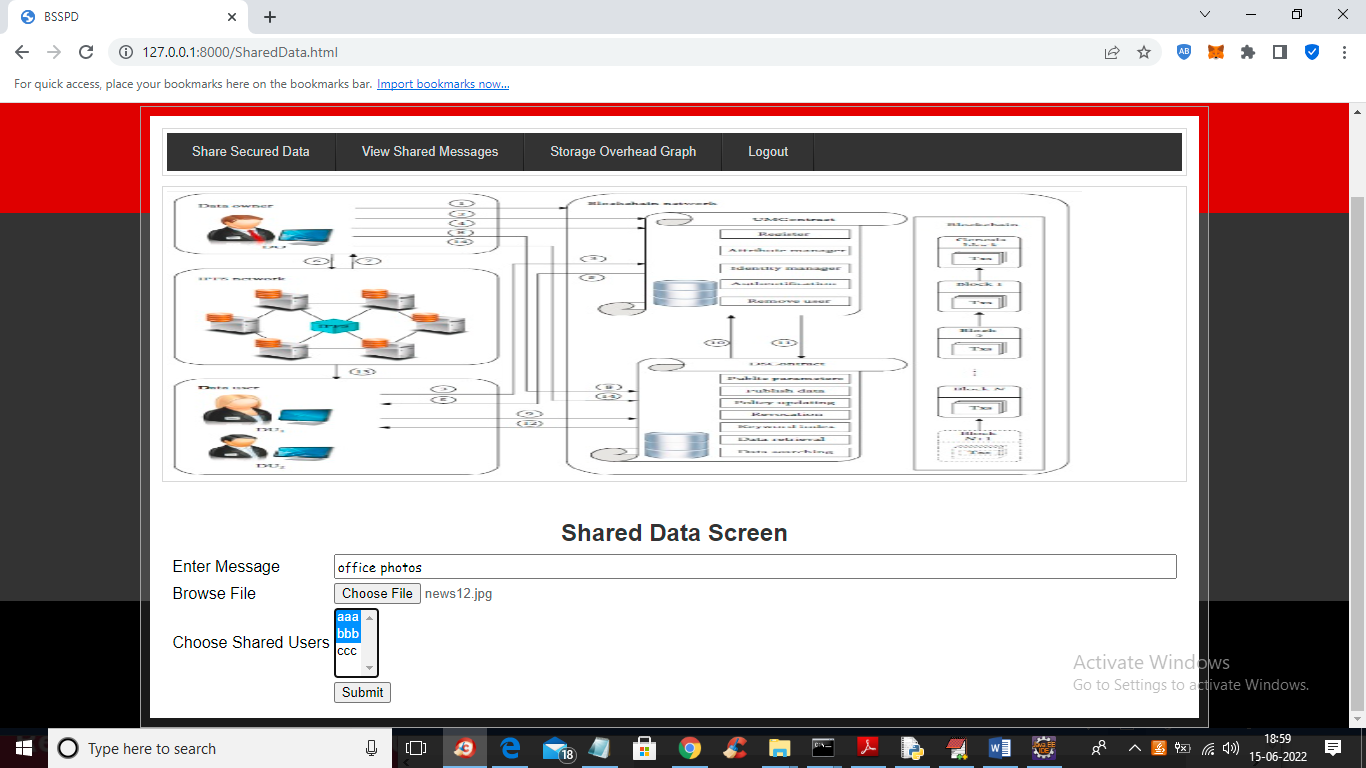


Fig.8.9.selecting users to share messages

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

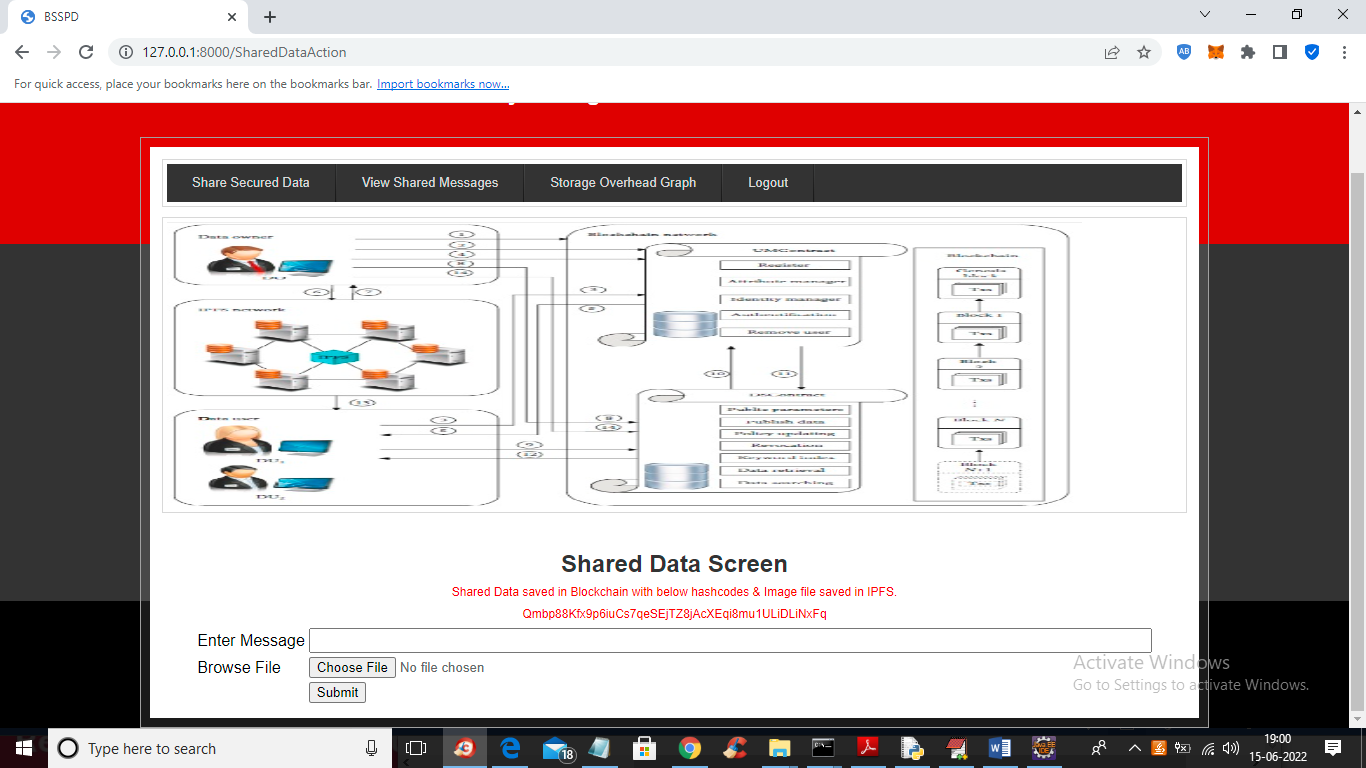


Fig.8.10.hash code generated and image saved in IPFS

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.

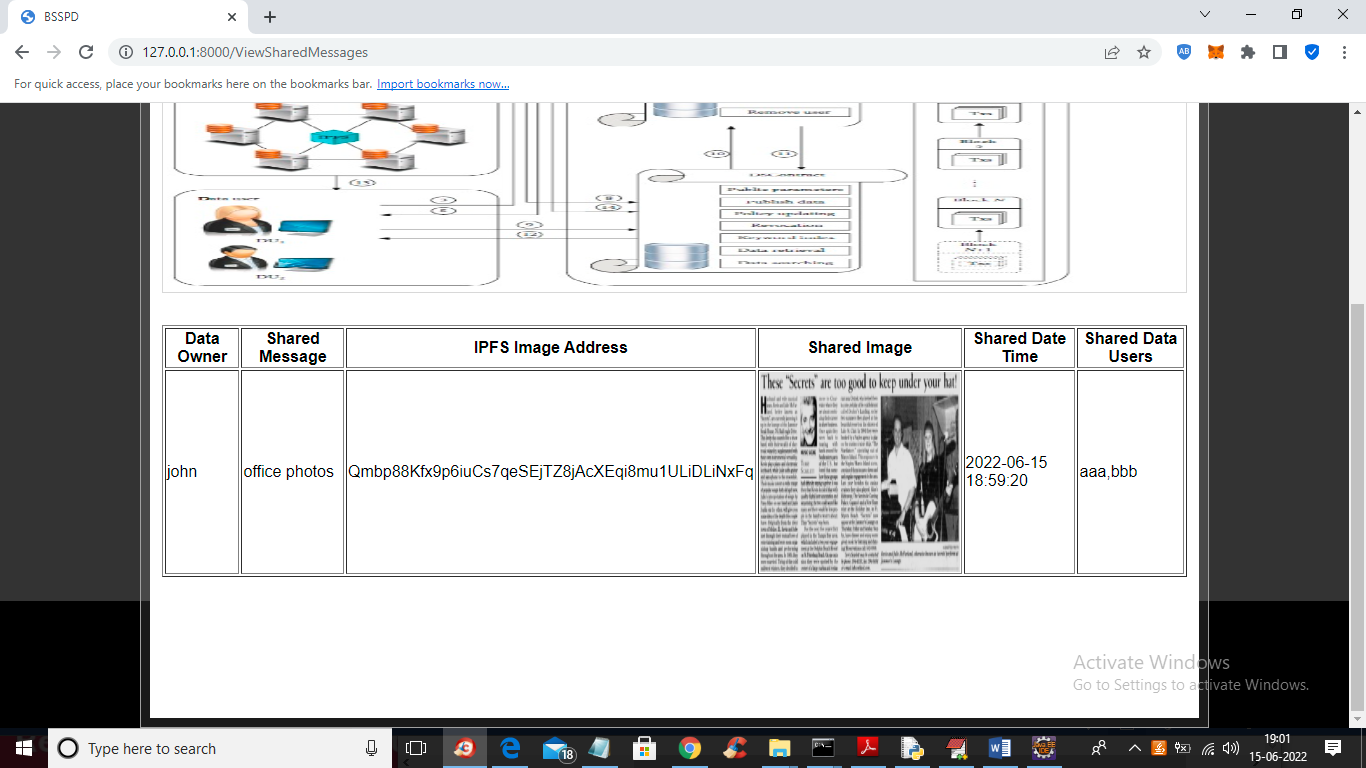


Fig.8.11.view shared messages

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.

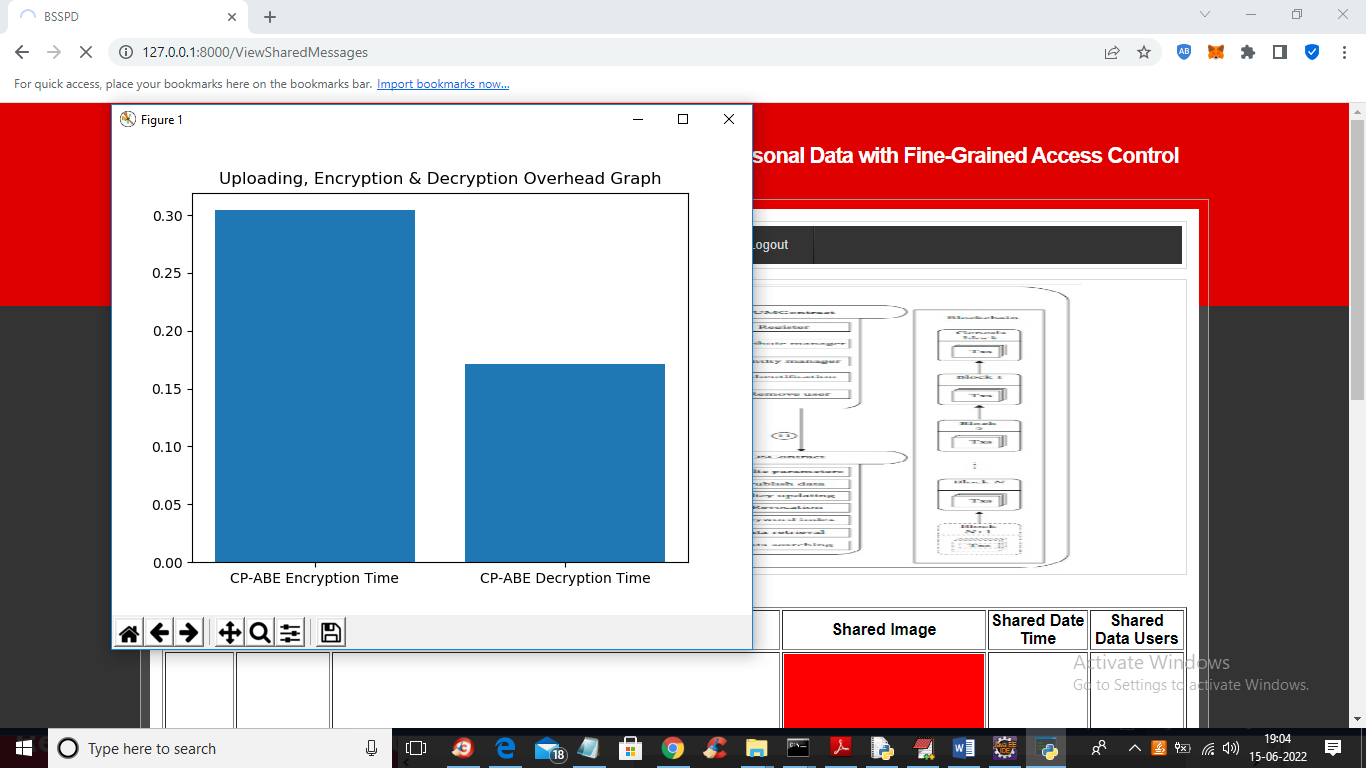


Fig.8.12.Encryption and Decryption graph

In above screen x-axis represents encryption and decryption and y-axis represents time overhead and now logout .

# 9. 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.

# 10. FUTURE ENHANCEMENTS

# Block chain technology reduces the chances of an attack on data as it verifies data and encrypts it using cryptography technology. Thus, this technology is immensely used in the field of cyber security. Blockchain technology helps to document a transaction as an everlasting distributes recorded and hence, supervises the transactions more transparently .Moreover, it helps in monitoring costs, employment, and releases at every point of the supply chain.

# 11. REFERENCES

1. X. Zhou, W. Liang, K. I. Wang, and L. T. Yang, “Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations,” IEEE Transactions on Computational Social Systems, vol. 8, no. 1, pp. 171–178, 2021.

2. X. Zhou, Y. Hu, W. Liang, J. Ma, and Q. Jin, “Variational LSTM enhanced anomaly detection for industrial big data,” IEEE Transactions on Industrial Informatics, 2020.

3.X. Yan, Y. Xu, B. Cui, S. Zhang, T. Guo, and C. Li, “Learning URL embedding for malicious website detection,” IEEE Transactions on Industrial Informatics, vol. 16, no. 10, pp. 6673– 6681, 2020.

4. X. Zhou, W. Liang, K. I. Wang, H. Wang, L. T. Yang, and Q. Jin, “Deep-learning-enhanced human activity recognition for Internet of healthcare things,” IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6429–6438, 2020.

5. G. Yu, X. Zha, X. Wang et al., “Enabling attribute revocation for fine-grained access control in blockchain-IoT systems,” IEEE Transactions on Engineering Management, vol. 67, no. 4, pp. 1213–1230, 2020.

6. Y. Xu, Q. Zeng, G. Wang, C. Zhang, J. Ren, and Y. Zhang, “An efficient privacy-enhanced attribute-based access control mechanism,” Concurrency and Computation: Practice and Experience, vol. 32, no. 5, article e5556, 2020.

7. X. Sun, S. Yao, S. Wang, and Y. Wu, “Blockchain-based secure storage and access scheme for electronic medical records in ipfs,” IEEE Access, vol. 8, pp. 59389–59401, 2020.

8. L. Tan, N. Shi, C. Yang, and K. Yu, “A blockchain-based access control framework for cyber-physical-social system big data,” IEEE Access, vol. 8, pp. 77215–77226, 2020.

9. Y. Xu, C. Zhang, Q. Zeng, G. Wang, J. Ren, and Y. Zhang, “Blockchain-enabled accountability mechanism against information leakage in vertical industry services,” IEEE Transactions on Network Science and Engineering, 2020.

10. Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, “Blockchain empowered arbitrable data auditing scheme for network storage as a service,” IEEE Transactions on Services Computing, vol. 13, no. 2, pp. 289–300, 2020.

11. Z. Cai, X. Zheng, and J. Yu, “A differential-private framework for urban traffic flows estimation via taxi companies,” IEEE Transactions on Industrial Informatics, vol. 15, no. 12, pp. 6492–6499, 2019.

1. Y. Xu, J. Ren, G. Wang, C. Zhang, J. Yang, and Y. Zhang, “A blockchain-based nonrepudiation network computing service scheme for industrial IoT,” IEEE Transactions on Industrial Informatics, vol. 15, no. 6, pp. 3632– 3641, 2019.
2. I. 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.
3. H. Gao, Z. Ma, S. Luo, and Z. Wang, “Bfr-mpc: a blockchain based fair and robust multi-party computation scheme,” IEEE Access, vol. 7, pp. 110439–110450, 2019.
4. Z. Cai and Z. He, “Trading private range counting over big IoT data,” in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 144–153, Dallas, TX, USA, 2019
5. L. Chen, W. K. Lee, C. C. Chang, K. K. R. Choo, and N. Zhang, “Blockchain based searchable encryption for electronic health record sharing,” Future Generation Computer Systems, vol. 95, pp. 420–429, 2019.
6. X. Yan, B. Cui, Y. Xu, P. Shi, and Z. Wang, “A method of information protection for collaborative deep learning under GAN model attack,” IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2019.
7. Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, “Deep learning-based inference of private information using embedded sensors in smart devices,” IEEE Network, vol. 32, no. 4, pp. 8–14, 2018.
8. S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, “Searching an encrypted cloud meets blockchain: a decentralized, reliable and fair realization,” in IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, pp. 792–800, Honolulu, HI, 2018.
9. Y. Xu, G. Wang, J. Yang, J. Ren, Y. Zhang, and C. Zhang, “Towards secure network computing services for lightweight clients using blockchain,” Wireless Communications and Mobile Computing, vol. 2018, 12 pages, 2018.
10. J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, “User collision avoidance CP-ABE with efficient attribute revocation for cloud storage,” IEEE Systems Journal, vol. 12, no. 2, pp. 1767–1777, 2018.
11. S. Wang, Y. Zhang, and Y. Zhang, “A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems,” IEEE Access, vol. 6, pp. 38437–38450, 2018.
12. M. Jemel and A. Serhrouchni, “Decentralized access control mechanism with temporal dimension based on blockchain,” in 2017 IEEE 14th International Conference on e-Business Engineering (ICEBE), pp. 177–182, Shanghai, 2017.