

A Project Report on

**“IDENTITY BASED ENCRYPTION: A KEY TO  
DATA PRIVACY IN PERMISSIONED  
BLOCKCHAIN”**

Submitted in partial fulfilment of the requirement for the  
award of the degree of

**BACHELOR OF TECHNOLOGY**  
**in**  
**COMPUTER SCIENCE AND ENGINEERING (Data Science)**

*By*

S. SALIYA	20X51A3255
B. SRAVANI	20X51A3205
S. ASIFA	20X51A3245
U. VAISHNAVI	20X51A3262
T. SHARADA	20X51A3260

*Under the Esteemed Guidance of*  
**Ms.S.RESHMA, M.Tech,**  
**Assistant Professor , Department of CSE (Data Science)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (Data Science)**

**SANTHIRAM ENGINEERING COLLEGE::NANDYAL**  
**(AUTONOMOUS)**

(Approved by AICTE, New Delhi: Affiliated to J.N.T. University, Anantapuramu, A.P.)

Accredited by NAAC (Grade-A), Accredited by NBA(ECE&CSE)

An ISO 9001:2015 Certified Institution, 2(f) and 12(b) recognition by UGC Act, 1956

Web: [www.srecnandyal.edu.in](http://www.srecnandyal.edu.in)

**YEAR: 2020-2024**

# **SANTHIRAM ENGINEERING COLLEGE::NANDYAL**

## **(AUTONOMOUS)**

(Approved by AICTE, New Delhi: Affiliated to J.N.T. University, Anantapuramu, A.P.)

Accredited by NAAC (Grade-A), Accredited by NBA(ECE&CSE)

An ISO 9001:2015 Certified Institution, 2(f) and 12(b) recognition by UGC Act, 1956

Web: [www.srecnandyal.edu.in](http://www.srecnandyal.edu.in)



## **CERTIFICATE**

This is to certify that the dissertation entitled “**IDENTITY BASED ENCRYPTION: A KEY TO DATA PRIVACY IN PERMISSIONED BLOCKCHAIN** ” is being submitted by **S.SALIYA(20X51A3255),B.SRAVANI(20X51A3205),S.ASIFA(20X51A3245),U.VAISHNAVI(20X51A3262), T.SHARADA(20X51A3260)** for partial fulfillment of the award of the Degree of Bachelor of Technology in **COMPUTER SCIENCE AND ENGINEERING (Data Science)** in the SANTHIRAM ENGINEERING COLLEGE Nandyal (Affiliated to J.N.T. University, Anantapur) is a record of bonafied work carried out by them under our guidance and supervision. The results embodied in this project have not been submitted to any other university for the award of any Degree.

**Signature of the Head of the Department:**

**Smt. N. RAMADEVI**, M.Tech , (Ph.D)  
Associate Professor & HOD

**Signature of the Project Guide:**

**Ms. S. RESHMA**, M.Tech  
Assistant Professor

**Signature of the External Examiner:**

**Date of Examination:**

## ACKNOWLEDGEMENT

This will certainly not be complete without acknowledgements paid to all these who have helped us in doing our project work.

We manifest our heartier thankfulness pertaining to our contentment over **Ms.S.Reshma garu** , as project guide with whose adroit concomitance the excellence has been exemplified in bringing out this project work with artistry.

It is a great pleasure to acknowledge our profound sense of gratitude to our Head of Department **Smt.N.Ramadevi garu**, and Coordinator **Mr.P.Subba Rao garu**, for his valuable and inspiring guidance, comments and suggestions and encouragement towards the course of this project.

Involuntarily, we are precious to divulge our sincere gratefulness to our principal, **Dr. M.V. Subramanyam garu**, who has been observed posing valance in abundance forward our individuality to acknowledge our project work tendentiously.

At the outset we thank our Honorable chairman **Dr.M.Santhiramudu garu**, correspondent, Santhiram Engineering College for providing us with good facilities and his moral support throughout the course.

We also express the overall exhilaration and gratitude to all the Teaching and Non- Teaching staff of CSE(DS) Dept., who admitted our project work and accentuated our attitude.

We also extend our heartfelt and sincere gratitude to our beloved parents for their tremendous motivation and moral support.

**S.SALIYA  
B.SRAVANI  
S.ASIFA  
U. VAISHNAVI  
T. SHARADA**

## TABLE OF CONTENTS

<b>S.NO</b>	<b>DESCRIPTION</b>	<b>Page No</b>
1	<b>INTRODUCTION</b>	1-2
	1.1 Introduction	1
	1.2 Objective of the project	2
2	<b>LITERATURE SURVEY</b>	3-5
	2.1 Literature Survey	3
3	<b>SYSTEM ANALYSIS</b>	6-8
	3.1 Existing system	6
	3.2 Proposed system	6
	3.3 Functional requirements	7
	3.4 Non-Functional requirements	7
	3.5 Software Requirements	8
	3.6 Hardware Requirements	8
4	<b>SYSTEM DESIGN</b>	9-18
	4.1 System architecture	9
	4.2 UML diagrams	10
5	<b>IMPLEMENTATION</b>	19-43
	5.1 Blockchain	19
	5.2 Sample code	23
	5.3 Software Environment	27
6	<b>SYSTEM TESTING</b>	44-45
	6.1 Testing strategies	44
	6.2 Test Cases	45
7	<b>RESULTS</b>	46-54
8	<b>CONCLUSION</b>	55

9	<b>FUTURE ENHANCEMENTS</b>	57
10	<b>APPLICATIONS</b>	58-59
11	<b>REFERENCES</b>	60-61
	<b>ANNEXURES</b>	
	i.    Journal Publication	
	ii.   Plagiarism Report of Documentation	

## LIST OF FIGURES

FIG.NO	NAME OF THE FIGURE	PAGENO
4.1.1	System Architecture	9
4.1.2	Data Flow Diagram	10
4.2.1	Use Case Diagram	12
4.2.2	Class Diagram	13
4.2.3	Activity Diagram	14
4.2.4	Sequence Diagram	15
4.2.5	Collaboration Diagram	16
4.2.6	Component Diagram	17
4.2.7	Deployment Diagram	18
5.1.1	Smart Contracts Development	20
5.1.2	Defined Function to Save User Signup Data And Message Data	21
5.1.3	Running Of Blockchain Server	21
5.1.4	Deploying Data Privacy	22
5.1.5	Blockchain Contract Calling Using Python	23
5.2.1	Official Site to Download Python	38
5.2.2	Downloading Python	39
5.2.3	Python Version	39
5.2.5	Opening Python Downloaded File	40
5.2.6	Installing Python	41
5.2.7	Setting Up Python	41
5.2.8	Opening Command Prompt	42
5.2.9	Python Version Checking	42

5.2.10	Opening Python IDLE	43
5.2.11	Saving Python File	43
7.1.1	Output Screens	46

## **ABSTRACT**

Blockchain is an emerging decentralized architecture and distributed public ledger technology underlying Bitcoin, and has recently attracted intensive attention from governments, financial institutions and high-tech enterprises. It is believed that blockchain can improve efficiency, reduce costs and enhance data security, but it is still in the face of serious privacy issues which may hinder the wide application of blockchain. In this paper, we present a practical scheme by adding the Identity-Based encryption system, which effectively improves the data privacy for non-transaction applications. Analyses show that our proposal has a high security level which can prevent both disguise and passive attacks, and is functional, effective and practical in many applications for non-transactional scenarios.



## CHAPTER 1 INTRODUCTION

### 1.1 Introduction:

Blockchain, a distributed public ledger system in a peer-to-peer network with distrust and decentralization, is gaining traction across a range of industries and use cases. A blockchain is, in general, an immutable transaction record that is kept up to date by a dispersed network of peer nodes. Using transactions approved by a consensus protocol and arranged into blocks with a hash connecting each block to the one before it, these nodes independently keep a copy of the ledger. Dispersed ledgers, which keep track of every transaction made on the network, are the central component of blockchain networks.

Blockchain, a distributed public ledger system in a peer-to-peer network with distrust and decentralization, is gaining traction across a range of industries and use cases. A blockchain is, in general, an immutable transaction record that is kept up to date by a dispersed network of peer nodes. By executing transactions that have been approved by a consensus protocol and arranged into blocks with a hash connecting each block to previous ones preceding it, those nodes each keep a copy of the ledger. Dispersed ledgers, which keep track of every transaction made on the network, are the central component of blockchain networks.

Permissionless blockchains usually use a "mined" native cryptocurrency or transaction fees to give an economic incentive to cover the high expenses of engaging in a sort of the byzantine fault tolerant consensus based on "proof of work" in order to ameliorate the lack of trust. In contrast, a blockchain that is administered by a group of verified and known participants is known as a permissioned blockchain. A permissioned blockchain offers a means of safeguarding the communications between a collection of organizations that share a same objective but lack complete mutual trust, like companies that trade goods, money, or information. Permissioned blockchains can employ classic Byzantine-fault tolerant consensus by depending on the peer identities. Our goal in writing this study is to secure data privacy on permissioned blockchains.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## **1.2 Objective of the Project:**

To develop a system to significantly enhance data privacy within permissioned blockchains while ensuring secure and efficient data sharing among authorized users, all while maintaining data integrity. To implement IBE to simplify key management, provide granular access control, and ensure the secure sharing of encrypted data. Utilize Ethereum blockchain's transparency, immutability, and decentralization to enhance data

## CHAPTER 2

### LITERATURE SURVEY

#### 2.1 Literature Survey:

##### **Identity-based cryptosystems and signature schemes:**

**ABSTRACT:** We present a novel kind of cryptographic method in this paper that allows any pair of users to exchange private or public keys, maintain key directories, and verify each other's signatures securely without relying on a third party's services. The plan is predicated on the presence of reliable key generation centers, whose main function is to provide each user with a unique smart card at initial network enrollment. With the use of the information encoded in this card, the user can sign and encrypt messages that he sends as well as decrypt and confirm messages that he receives in a completely autonomous manner, regardless of who the other party is. It is not necessary to update previously issued cards when new users join the network, and the various centers do not have to coordinate their activities or even to keep a user list. The centers can be closed after all the cards are issued, and the network can continue to function in a completely decentralized way for an indefinite period.

##### **Identity-based encryption from the Weil pairing:**

**ABSTRACT:** We provide an identity-based encryption system (IBE) that is completely operational. The system assumes an elliptic curve variation of the computational Diffie-Hellman problem and has selected ciphertext security in the random oracle model. The foundation of our system is the Weil pairing. We provide accurate definitions of safe identity-based encryption techniques together with a number of examples of their uses.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## **Efficient selective-ID secure Identity based encryption without random oracles:**

**ABSTRACT:** We build two effective Identity Based Encryption (IBE) systems that do not require the random oracle model and are selectively identity secure. Compared to the conventional security model for IBE, selective identity secure IBE has a little poorer security model. While in the normal model the adversary is free to select this identity adaptively, in this model the adversary must proactively commit to the identity it plans to attack. Our initial safe IBE system expands to provide a Hierarchical secure IBE that is selective and does not rely on random oracles.

## **Secure identity-based encryption without random oracles:**

**ABSTRACT:** Without using the random oracle paradigm, we build two effective Identity Based Encryption (IBE) systems that are selectively identity secure. Compared to the conventional security model for IBE, selective identity secure IBE has a little poorer security model. While in the normal model the adversary is free to select this identity adaptively, in this model the adversary must proactively commit to the identity it plans to attack. Our initial safe IBE system expands to provide a Hierarchical secure IBE that is selective and does not rely on random oracles.

## **Bulletproofs: Efficient range proofs for confidential transactions:**

**ABSTRACT:** We provide Bulletproofs, a revolutionary zero-knowledge proof protocol with very short proofs that doesn't require a trusted setup. The proof size is simply proportional to the size of the witness. Because they allow one to demonstrate that a committed value is within a range with just two  $\log_2(n)+9$  group and field elements—where  $n$  is the bit length of the range—bulletproofs are particularly well suited for effective range proofs on committed values. The times required to generate and verify proofs are linear in  $n$ . The range proofs in existing proposals for confidential transactions in Bitcoin and other cryptocurrencies are linear(in  $n$ ) sized; bulletproofs considerably improve on them.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

Additionally, bulletproofs allow range proofs to be aggregated, meaning a party can demonstrate that  $m$  commitments fall within a certain range with just an additive  $O(\log(m))$  group of items throughout the course of a single demonstration. Through a straightforward multi-party computation (MPC) protocol for creating Bulletproofs, we allow the parties to build a single proof without disclosing their inputs to one another, aggregating proofs from several parties. This MPC protocol uses either a logarithmic number of rounds and logarithmic communication, or a constant number of rounds and linear communication. We demonstrate that, although asymptotically linear, verification time is actually highly efficient. Batch validating 32 aggregated range proofs will cost less than verifying 32 ECDSA signatures, on a marginal cost basis. Building bulletproofs

Bulletproofs expand upon Bootle et al.'s methods (EUROCRYPT 2016). Beyond range proofs, bulletproofs, which only rely on the discrete logarithm assumption, provide succinct zero-knowledge proofs for general arithmetic circuits without the need for a reliable setup. We talk about many applications that would be useful for bulletproofs, mostly related to cryptocurrencies. The dispersed and trustless character of blockchains is especially well suited for Bulletproofs' efficiency. This article can be found in print in its entirety.

## CHAPTER 3

### SYSTEM ANALYSIS

#### 3.1 Existing System:

Several privacy protection technologies used in blockchain have been studied in literatures, such as ring signature, homomorphic encryption and zero-knowledge proofs. As far as we know, most of these works are emphasizing the privacy challenges on the anonymity and confidentiality of transaction data on the decentralized cryptocurrencies. However, for many applications in real economy, such as quality tracking, copyright and supply chain finance, what is needed is only data transfer which means operations in consensus are primarily the consolidation of data.

#### Disadvantages Of Existing System:

Blockchain technology is thought to be able to increase productivity, lower expenses, and improve data security, but there are still significant privacy concerns that could prevent blockchain from being widely used. Reduced safeguards for privacy

#### 3.2 Proposed System:

Our goal in writing this study is to secure data privacy on permissioned blockchains. We describe a strategy that effectively increases the data privacy for non-transaction applications by introducing an Identity-Based encryption mechanism. Our proposal is practical and safe, according to analyses.

#### Advantages Of Proposed System:

Analyses show that our proposal has a high security level which can prevent both disguise and passive attacks, and is functional, effective and practical in many applications for non-transactional scenarios. This scheme provides an inspiring way to achieve delicate confidentiality of the transactions in many applications for non-transactional scenarios

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## 3.3 Functional Requirements:

- New User Signup
- User Login
- Post Private Messages
- View Shared Private Message Blockchain

## 3.4 Non-Functional Requirements:

NFRs (Non-Functional Requirements) outline a software system's quality attribute. They evaluate the software system according to non-functional criteria that are essential to its success, such as portability, security, responsiveness, and usability. An example of a nonfunctional need might be "how quickly can I load the website?" Non-functional requirements provide you the ability to place limitations or restrictions on how the system is designed across different iterations of the agile backlog. As an example, when there are more than 10,000 users online at once, the website should load in three seconds. Just as important as a functional requirement's description is one for non-functional requirements.

- Usability requirement
- Service ability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## 3.5 Software Requirements:

The product viewpoint and features, operating system and environment, graphics requirements, design limitations, and user documentation are all included in the functional requirements, often known as the overall description documents. The project's overall overview, including its strengths and weaknesses and the strategies for addressing them, is provided by the appropriation of requirements and execution limitations.

Python idle 3.7 version(or)

Anaconda 3.7(or)

Jupyter(or)

Google colab

## 3.6 Hardware Requirements:

The minimal hardware requirements vary greatly depending on the specific software that a given Enthought Python, Canopy, or VS Code user is developing. Applications that need to swiftly complete several calculations or operations will need a faster CPU, while those that need to keep big arrays or objects in memory would need more RAM.

Operating system : Windows, Linux

Processor : Minimum intel i3

Ram : Minimum 4 Gb

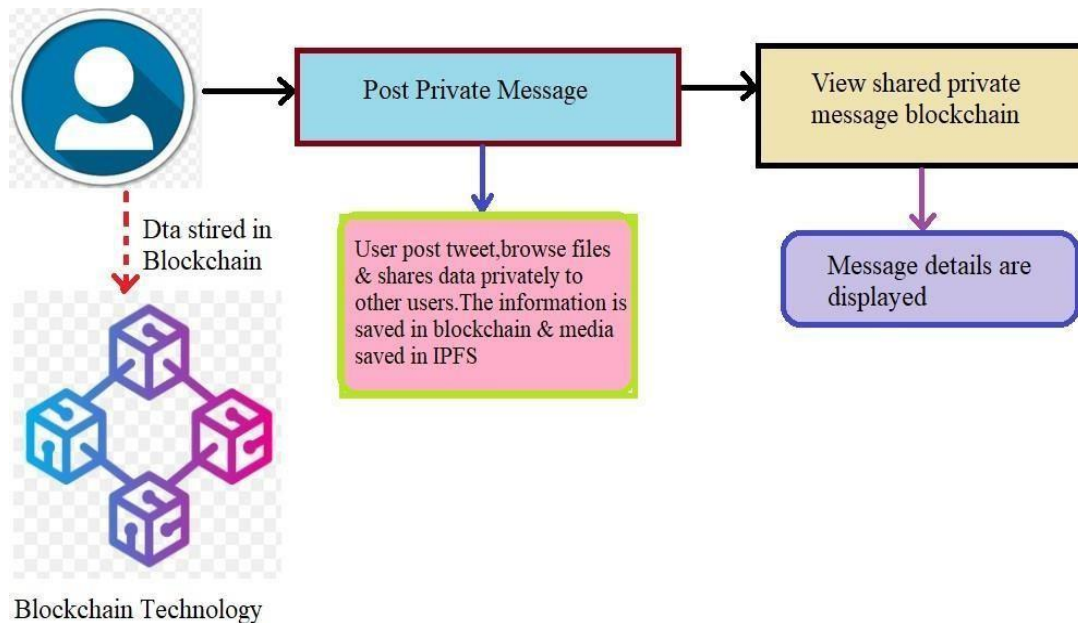
Hard disk : Minimum 250 Gb.



## CHAPTER 4

### SYSTEM DESIGN

#### 4.1 System Architecture:



**Fig.4.1.1 System architecture**

#### Data Flow Diagram:

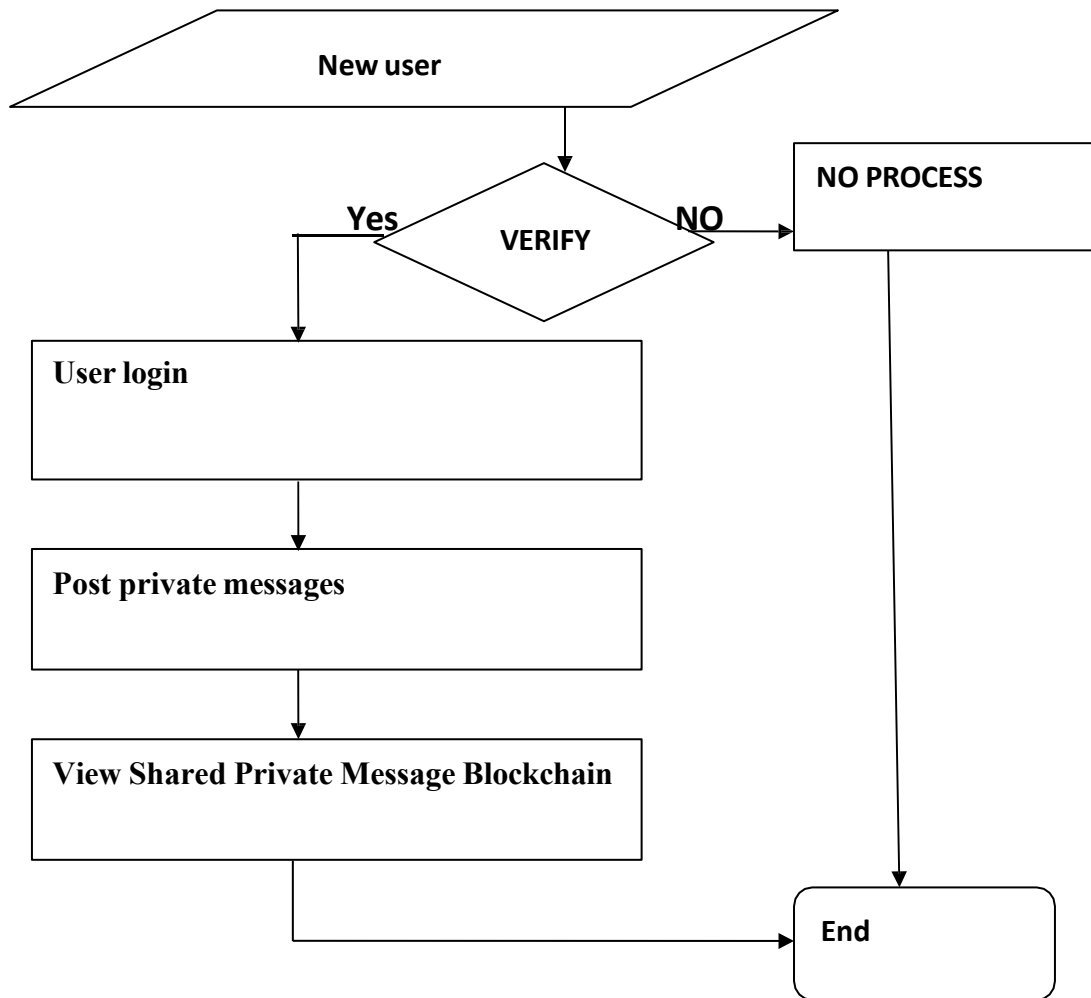
Another name for the DFD is a bubble chart. A system can be represented using this straightforward graphical formalism in terms of the input data it receives, the different operations it performs on that data, and the output data it generates.

- The data flow diagram, or DFD, is a crucial modeling instrument. The components of the system are modeled using it. These elements consist of the system's procedure, the data it uses, an outside party that communicates with it, and the information flows within it.
- DFD illustrates the flow of information through the system and the various changes that alter it. It's a visual method that shows how information moves and changes that are applied as data moves from input to output.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.



**Fig.4.1.2 Dataflow diagrams**

## 4.2 UML Diagrams:

Unified Modeling Language is known as UML. An industry-standard general-purpose modeling language used in object-oriented software engineering is called UML. The Object Management Group developed and oversees the standard. The intention is for UML to spread as a standard language for modeling object-oriented software. The two main parts of UML as

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

it exists now are a notation and a meta-model. In the future, UML may also include other processes or methods that are connected to it.

- Business modeling and other non-software systems can also be modelled using the Unified Modeling Language, a standard language for defining, visualizing, building, and documenting software system artifacts.
- Developing object-oriented software and software in the development process both heavily rely on the UML, which is a collection of best engineering practices that have been successful in modeling big and complex systems. The UML primarily expresses software project design through graphical notations.

## **Goals:**

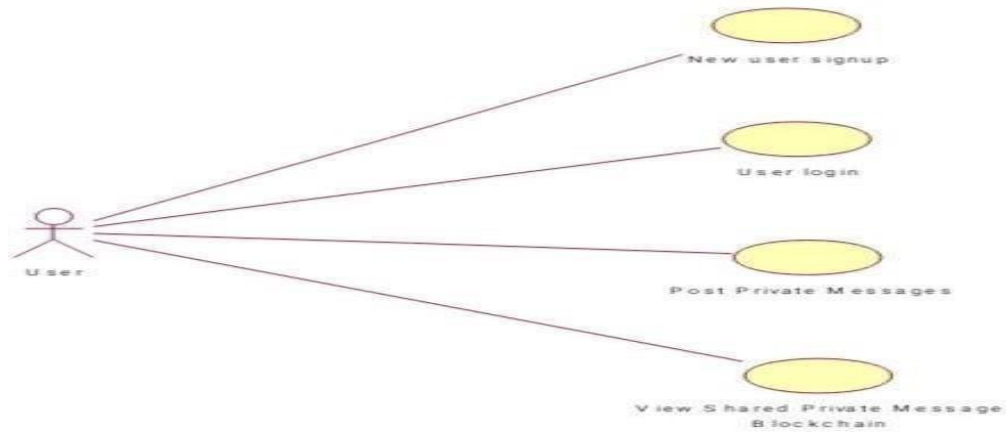
- The Primary goals in the design of the UML are as follows:
- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.

## **USE CASE DIAGRAM:**

According to the Unified Modeling Language (UML), a use case diagram is a particular kind of behavioral diagram that is produced from and defined by a use case study. Its objective is to provide a graphical summary of the functionality that a system offers in terms of actors, use cases (representations of their goals), and any interdependencies among those use cases. A use case diagram's primary goal is to display which actors receive which system functionalities. It is possible to illustrate the actor roles in the UML diagram which will be helpful to showcase.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---



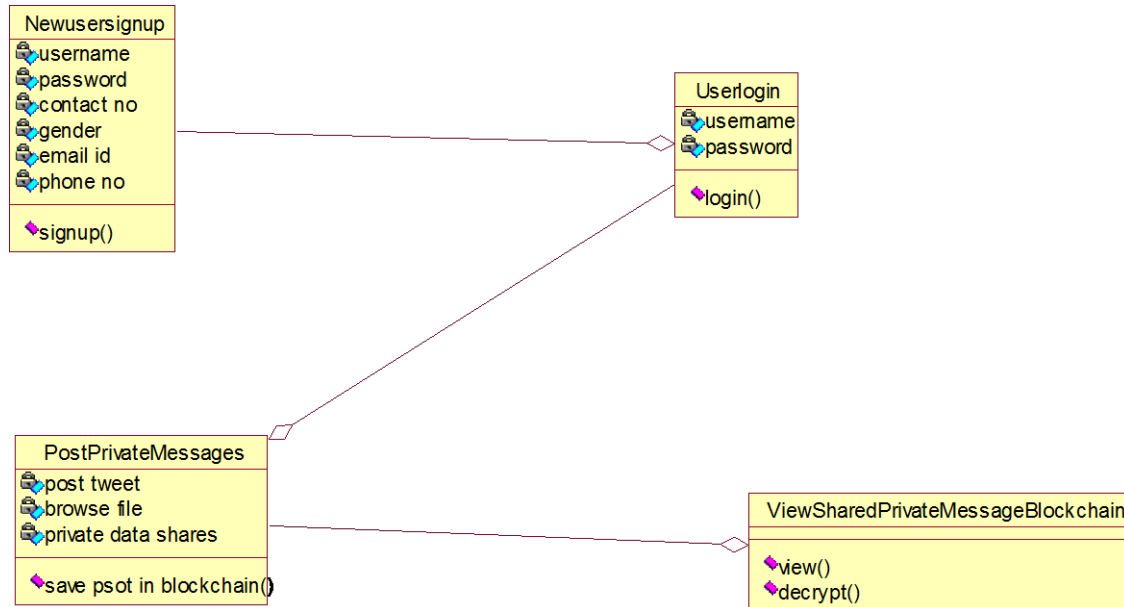
**Fig.4.2.1 Use case diagram**

## **CLASS DIAGRAM:**

The use case diagram is improved and a more thorough system design is defined by the class diagram. The use case diagram's actors are categorized into a number of related classes by the class diagram. It is possible for there to be a "has-a" or "is-a" relationship or association between the classes. There may be particular features that each class in the class diagram is capable of offering. The term "methods" refers to the functions that the class offers. Aside from this, any class could have some "attributes" that make the class distinct.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

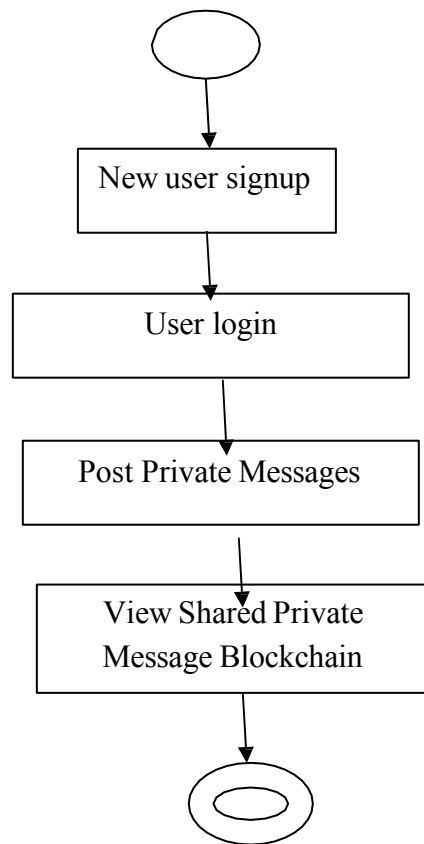
---



**Fig.4.2.2 Class diagram**

## ACTIVITY DIAGRAM:

The activity diagram shows how the system's processes are organized. An activity diagram has the same elements as a state diagram: activities, actions, guard conditions, initial and final states, and transitions. An additional crucial behavioral diagram in the UML diagram for describing the system's dynamic elements is the activity diagram. An activity diagram is a more complex form of a flow chart that illustrates how an activity flows from one to the next. Activity Diagrams show how various levels of abstraction can be used to depict how activities are coordinated to create a service. Generally, an operation needs to accomplish multiple goals that need to be coordinated, or it needs to determine how the events in a single use case relate to each other, especially in use cases where there may be overlap in the activities that need to be coordinated. It can also be used to simulate the coordination of a set of use cases to depict business workflows.



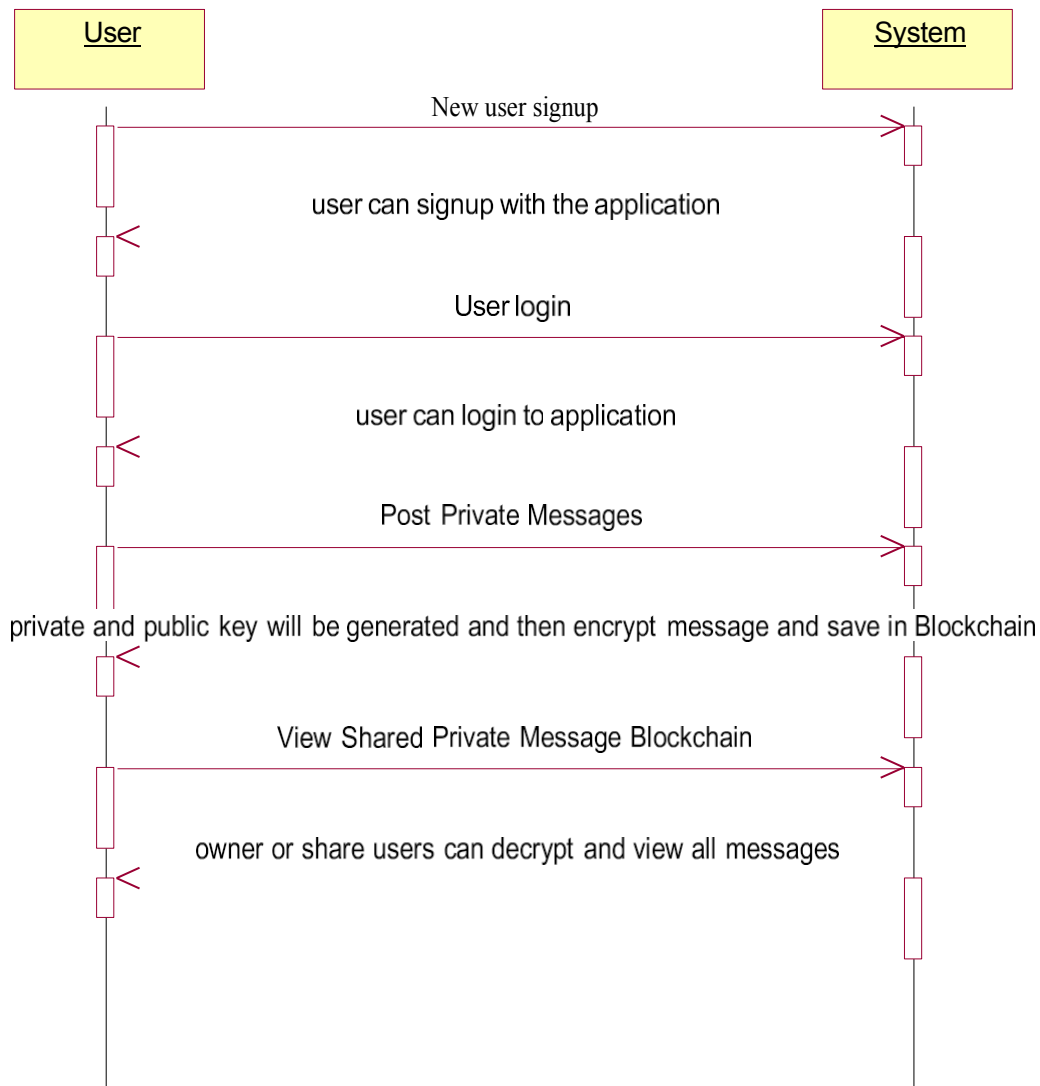
**Fig.4.2.3 Activity diagram**

## **SEQUENCE DIAGRAM:**

The way various system items interact with one another is depicted in a sequence diagram. A sequence diagram's time-ordering is one of its key features. This indicates that a step-by-step representation of the precise order in which the items interacted is provided. In the sequence diagram, various objects communicate with one another by sending "messages". In a UML diagram, an actor symbolizes a certain kind of role that it plays in interacting with the system and its objects. Here, it's critical to remember that an actor is never inside the boundaries of the system that the UML diagram is intended to represent.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---



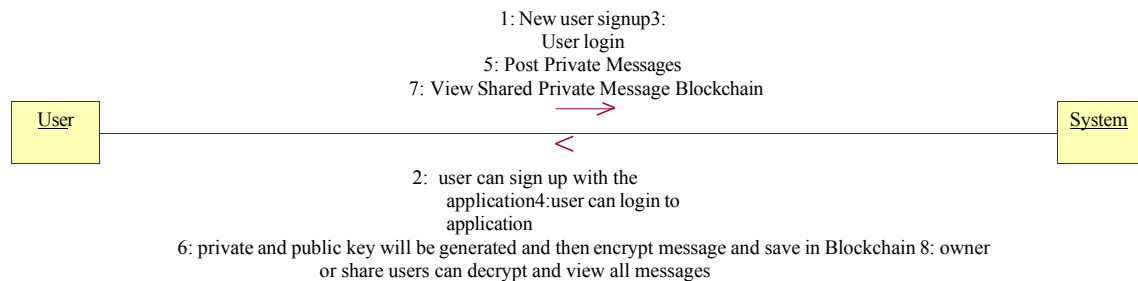
**Fig.4.2.4 Sequence diagram**

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## COLLABORATION DIAGRAM:

A cooperation diagram combines the ways in which various things interact with one another. To make it easier to follow the order of the encounters, they are listed as numbered interactions. All potential interactions between each object and other objects are identified with the aid of the cooperation diagram.



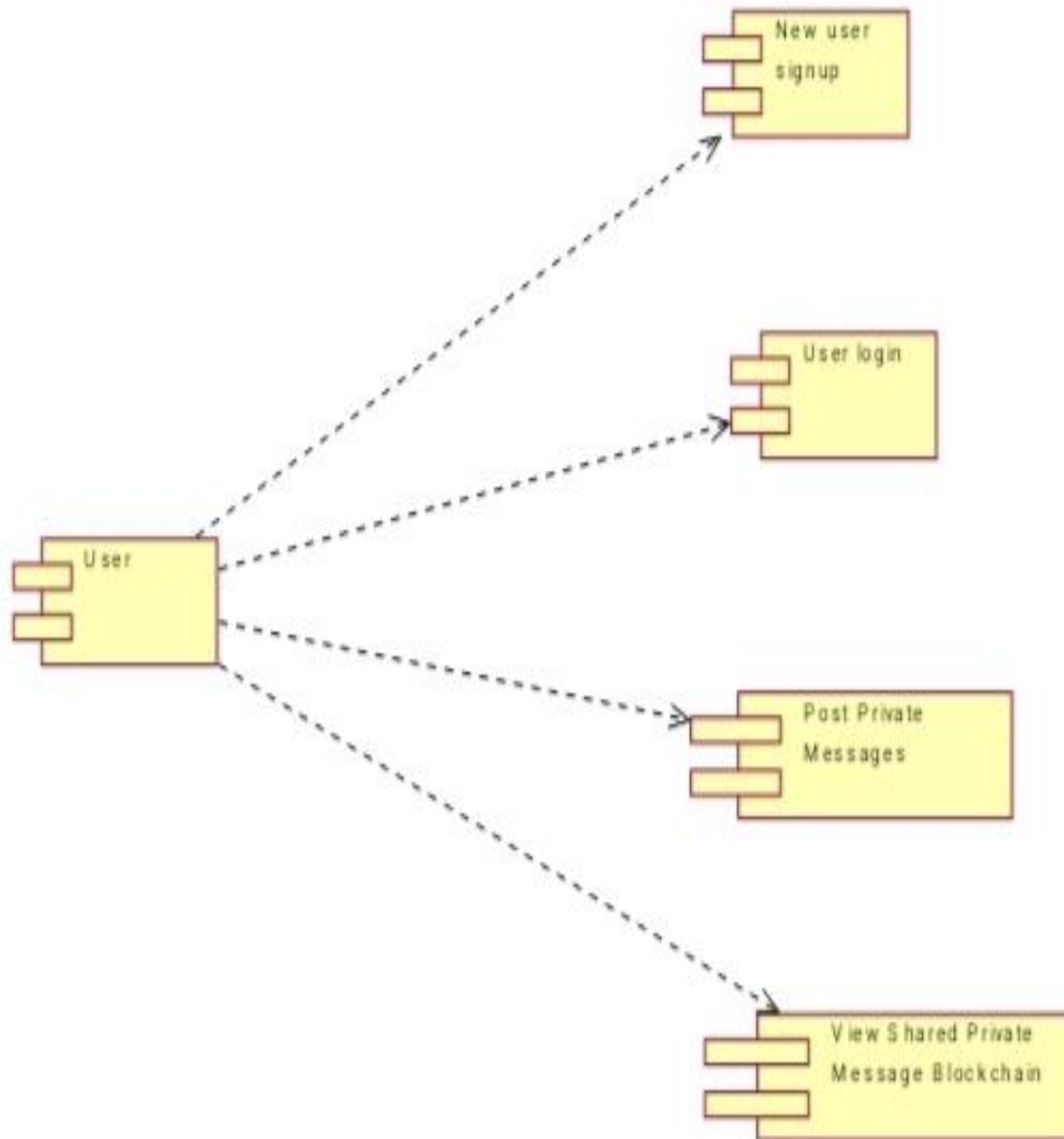
**Fig.4.2.5 Collaboration diagram**

## COMPONENT DIAGRAM:

The high-level components that comprise the system are represented in the component diagram. A high-level representation of the system's components and their relationships is shown in this diagram. The parts removed from the system after it has completed the development or manufacturing stage are shown in a component diagram.

Component Diagrams in Unified Modeling Language (UML) are used to display code modules of a system. Subsystems are typically modeled using them. It depicts how every single part behaves when a system program is being executed and operated. They are also employed to illustrate and symbolize the arrangement and structure of each component. Application applications, ActiveX controls, Java Beans, backend databases, and certain ASP programs are examples of these code modules. The component diagrams show how view models are implemented. The purpose of component diagrams is to show dependencies and interfaces within software architecture. Modules of a class that typically represent an independent subsystem are simply referred to as components.





**Fig.4.2.6 Component diagram**

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## DEPLOYMENT DIAGRAM:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.



**Fig 4.2.7: Deployment diagram**

## CHAPTER 5

### IMPLEMENTATION

#### 5.1 Blockchain:

Blockchain is a distributed technology where it will store data at multiple nodes in a network and has inbuilt support for data verification and encryption. Blockchain store data as block/transaction and associate each block with unique hash code and before storing new block or data then Blockchain will verify all blocks hash code and if data not tamper then result into same hash code and verification will be successful otherwise verification failed and due to this verification Blockchain consider as immutable which means data cannot be alter in any manner.

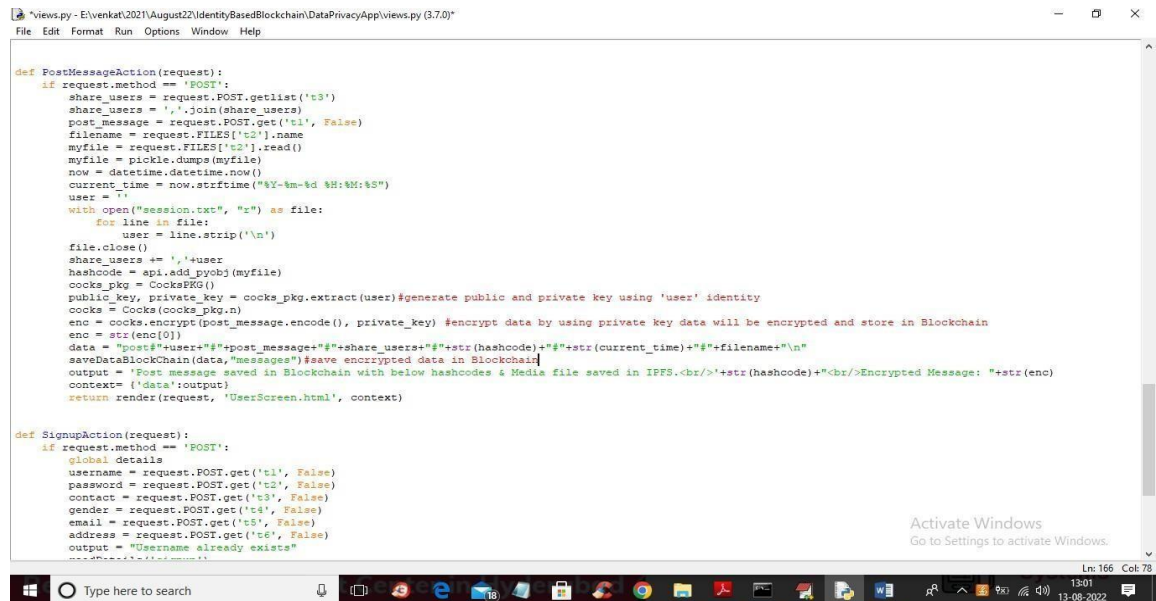
Blockchain is of two types Permissioned and Permission less, any person can join in permission less Blockchain but only authorized companies are allowed to access Permissioned based Blockchain and all data stored in Permissioned based Blockchain are visible to all authorized users and to provide privacy author of this paper employing IBE (Identity based encryption) algorithm to encrypt data stored in Permissioned Based Blockchain.

IBE encrypt data using person identity such as Mobile number or username, first it will generate private and public key by using person identity and then by using private key it will encrypt data and by using public key it will decrypt data.

Data owner can share encrypted data to other member by generating keys on their identity and only share users can decrypt and view data and non-share users cannot decrypt and view data. So only allowed users can access data and achieve data privacy

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

In below screen we are showing code for IBE to generate keys and encrypt data



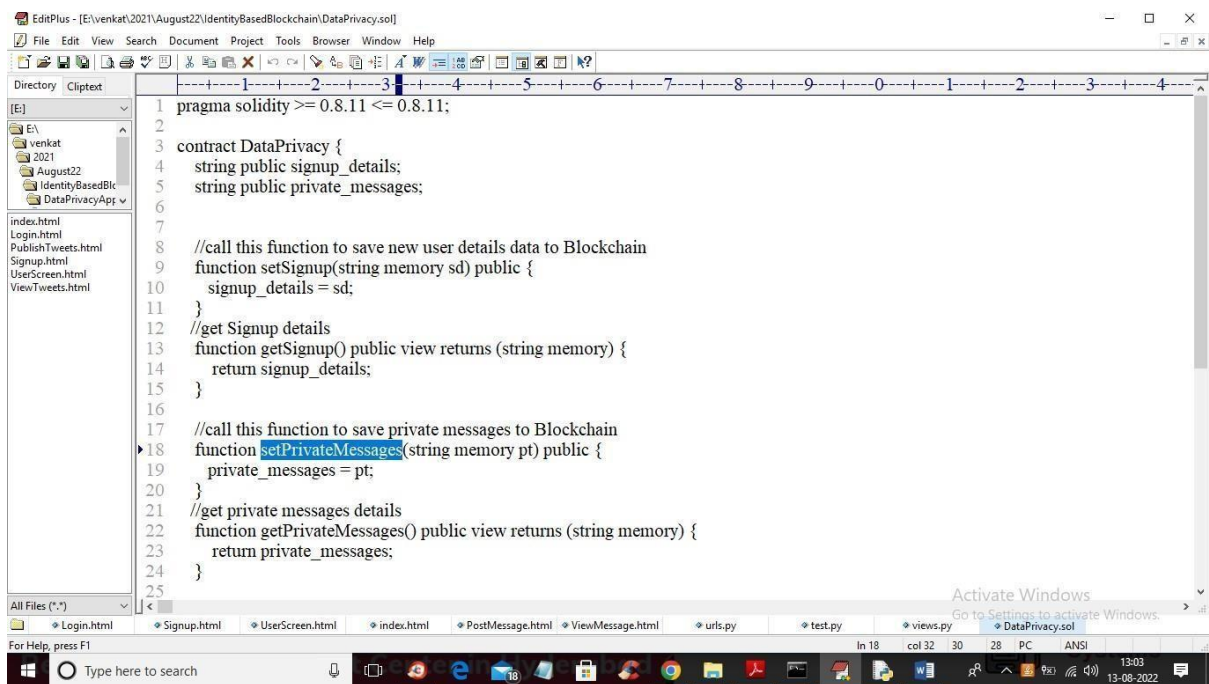
```
def PostMessageAction(request):
    if request.method == 'POST':
        share_users = request.POST.getlist('t3')
        share_users = ','.join(share_users)
        post_message = request.POST.get('t1', False)
        filename = request.FILES['t2'].name
        myfile = request.FILES['t2'].read()
        myfile = pickle.dumps(myfile)
        now = datetime.datetime.now()
        current_time = now.strftime("%Y-%m-%d %H:%M:%S")
        user = ''
        with open("session.txt", "a") as file:
            for line in file:
                user = line.strip('\n')
            file.close()
        share_users += ',' + user
        hashcode = api.add_pyobj(myfile)
        coocks_pkg = Cocks2PKG()
        public_key, private_key = coocks_pkg.extract(user) #generate public and private key using 'user' identity
        coocks = Cocks(coocks_pkg.n)
        enc = coocks.encrypt(post_message.encode(), private_key) #encrypt data by using private key data will be encrypted and store in Blockchain
        enc = str(enc[0])
        data = "post#" + user + "#" + post_message + "#" + share_users + "#" + str(hashcode) + "#" + str(current_time) + "#" + filename + "\n"
        saveDataBlockchain(data, "messages") #save encrypted data in Blockchain
        output = 'Post message saved in Blockchain with below hashcodes & Media file saved in IPFS.<br/>' + str(hashcode) + "<br/>Encrypted Message: " + str(enc)
        context = {'data': output}
        return render(request, 'UserScreen.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"
```

**Fig 5.1.1: Smart contracts Development**

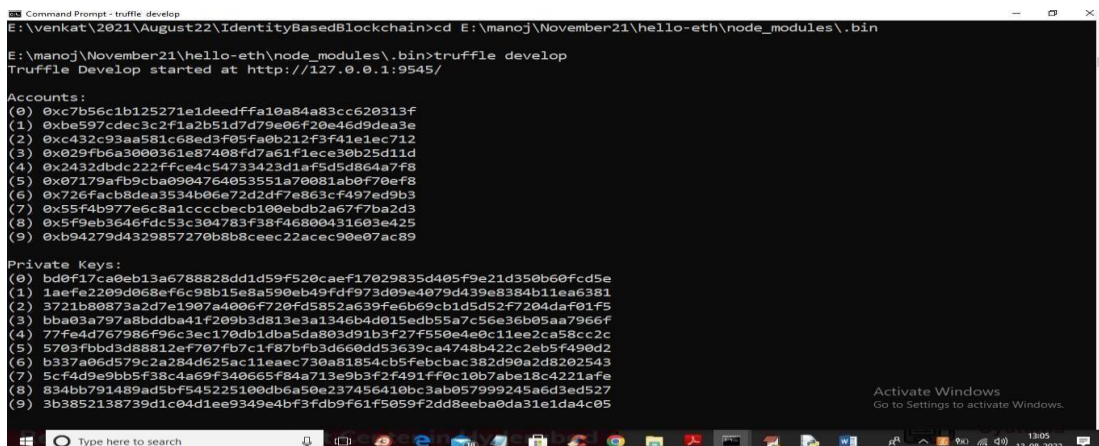
In above screen read red color comments to know about data encryption using IBE and now to store data in Permissioned Blockchain we need developed smart contract which will contains functions to store and retrieve data. In below screen we are showing Smart Contract code for Data Privacy

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain



**Fig 5.1.2: Defined function to save USER signup data and Message data**

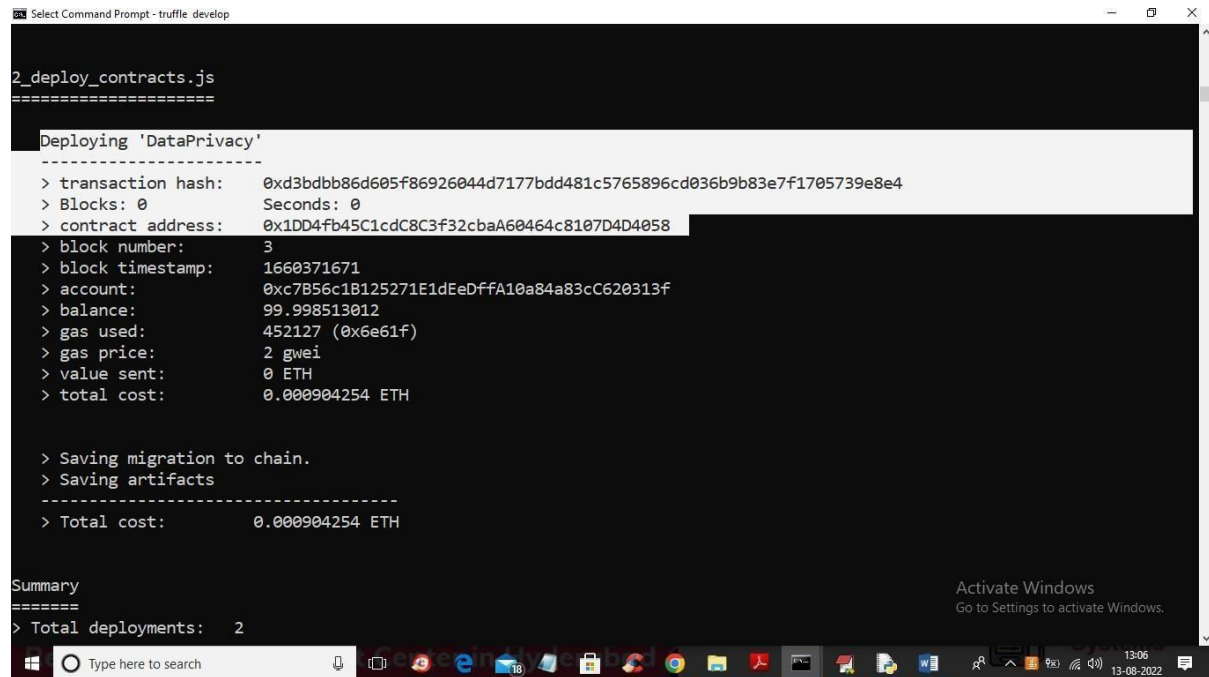
In above smart contract solidity code, we have defined functions to save USER signup data and Messages data. Now we need to deploy above contract in Blockchain by using below steps  
Go inside 'hello-eth/node-modules/bin' folder and then double click on 'runBlockchain.bat' file to start Blockchain server and get below screen



**Fig 5.1.3: Running of Blockchain Server**

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

In above screen Blockchain generated some default keys and accounts and in above screen type command as 'truffle migrate' and press enter key to deploy contract and get below



```
Select Command Prompt - truffle develop

2_deploy_contracts.js
=====
Deploying 'DataPrivacy'
-----
> transaction hash: 0xd3bdbb86d605f86926044d7177bdd481c5765896cd036b9b83e7f1705739e8e4
> Blocks: 0 Seconds: 0
> contract address: 0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058
> block number: 3
> block timestamp: 1660371671
> account: 0xc7B56c1B125271E1dEeDffA10a84a83cC620313f
> balance: 99.998513012
> gas used: 452127 (0x6e61f)
> gas price: 2 gwei
> value sent: 0 ETH
> total cost: 0.000904254 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.000904254 ETH

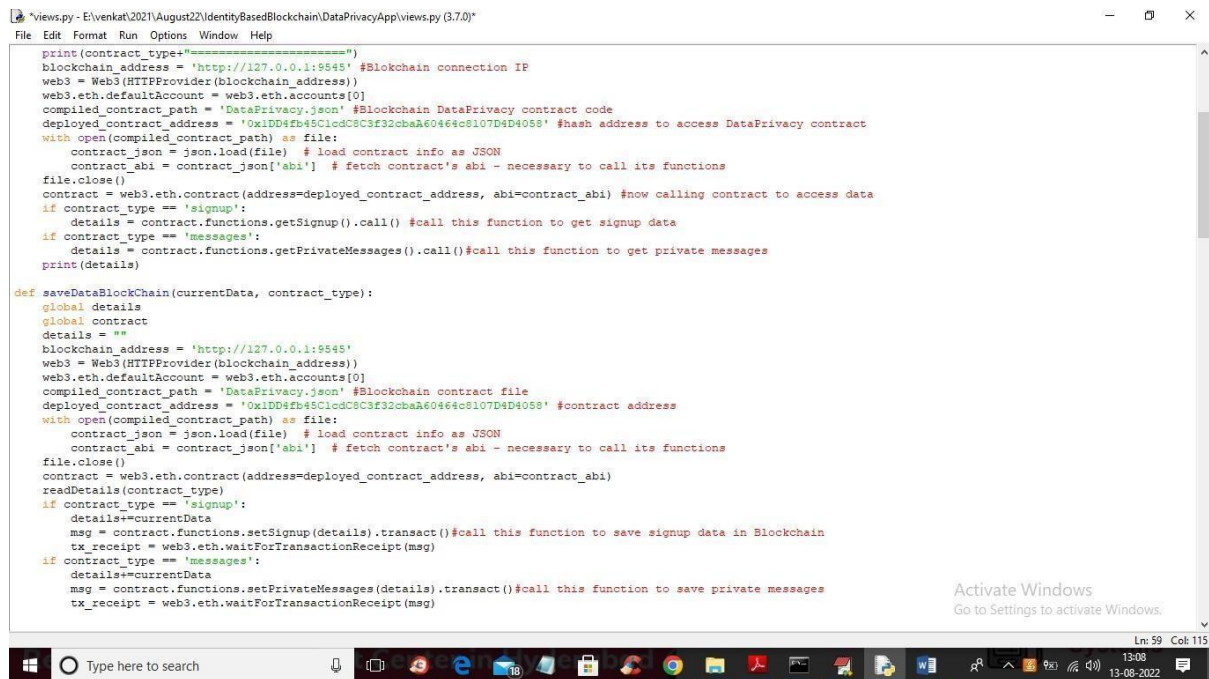
Summary
=====
> Total deployments: 2
```

output

**Fig 5.1.4: Deploying Data Privacy**

In above screen in white color text, we can see 'Data Privacy' Contract deployed (let above screen running) and we got contract address also and this address we need to specify in python program to allow it call Blockchain to store and retrieve data and in below screen we can see python code calling Blockchain functions

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain



```
*views.py - E:\venkat\2021\August22\IdentityBasedBlockchain\DataPrivacyApp\views.py (3.7.0)*
File Edit Format Run Options Window Help

print(contract_type+"=====")
blockchain_address = 'http://127.0.0.1:9545' #Blockchain connection IP
web3 = Web3(HTTPProvider(blockchain_address))
web3.eth.defaultAccount = web3.eth.accounts[0]
compiled_contract_path = 'DataPrivacy.json' #Blockchain DataPrivacy contract code
deployed_contract_address = '0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058' #hash address to access DataPrivacy 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() #call this function to get signup data
if contract_type == 'messages':
    details = contract.functions.getPrivateMessages().call() #call this function to get private messages
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 = 'DataPrivacy.json' #Blockchain contract file
    deployed_contract_address = '0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058' #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() #call this function to save signup data in Blockchain
        tx_receipt = web3.eth.waitForTransactionReceipt(msg)
    if contract_type == 'messages':
        details=currentData
        msg = contract.functions.setPrivateMessages(details).transact() #call this function to save private messages
        tx_receipt = web3.eth.waitForTransactionReceipt(msg)
```

**Fig 5.1.5: Blockchain Contract calling using python**

In above screen read red color comments to know about Blockchain contract calling using python.

In base paper author is sending messages to phone for verification but we don't have any SMS facility or email facility as GMAIL block all email accessing from python or any other code. So, we cannot verify with phone or email but we are verifying users with IBE public and private keys. Only genuine user will have valid key and they only can decrypt data. In this paper author implemented this project as NON-TRANSACTION based data sharing application

To implements this project, we have designed following modules

- New User Signup: using this module user can sign up with the application
- User Login: using this module user can login to application
- Post Private Messages: using this module user can post messages and using USER'S 'username or id' private and public key will be generated and then encrypt message and save in Blockchain. User can share this message with multiple users

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

- View Shared Private Message Blockchain: using this module message owner or share users can decrypt and view all messages and non-sharing users cannot view or decrypt message.

## 5.2 SAMPLE CODE:

```
import bonaci

import Json

import base64

import base58

import nacl.hash

import nacl.signing

import nacl.secret

import nacl.utils

def to_b64(barray):

    return base64.b64encode(barray).decode('utf8')

def from_b64(string):

    return base64.b64decode(string)

def to_bytes(obj):

    if isinstance(obj, bytes):

        return obj

    if isinstance(obj, dict):

        obj = json.dumps(obj, sort_keys=True, separators=(',', ':'))
```



# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

```
    return obj.encode('utf8')

def sign(data, sign_key):

    return to_b64(sign_key.sign(to_bytes(data)))

def random(size=nacl.secret.SecretBox.KEY_SIZE):

    return nacl.utils.random(nacl.secret.SecretBox.KEY_SIZE)

def hash(data):

    return nacl.hash.sha256(to_bytes(data)).decode('utf8')

def pkencrypt(data, sender_sk, receiver_pk):

    sender_sk = nacl.public.PrivateKey(base58.b58decode(sender_sk))

    receiver_pk = nacl.public.PublicKey(base58.b58decode(receiver_pk))

    box = nacl.public.Box(sender_sk, receiver_pk)

    nonce = nacl.utils.random(nacl.public.Box.NONCE_SIZE)

    encrypted = box.encrypt(to_bytes(data), nonce)

    return to_b64(encrypted)

def pkdecrypt(data, sender_pk, receiver_sk):

    sender_pk = nacl.public.PublicKey(base58.b58decode(sender_pk))

    receiver_sk = nacl.public.PrivateKey(base58.b58decode(receiver_sk))

    box = nacl.public.Box(receiver_sk, sender_pk)

    return box.decrypt(from_b64(data))
```

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

```
def encrypt(data, key):

    box = nacl.secret.SecretBox(key)

    nonce = nacl.utils.random(nacl.secret.SecretBox.NONCE_SIZE)

    cipher = box.encrypt(to_bytes(data), nonce)

    return to_b64(cipher)

def decrypt(cipher, key):

    box = nacl.secret.SecretBox(key)

    decrypted = box.decrypt(cipher)

    return json.loads(decrypted.decode('utf8'))

def keypair(seed=None):

    if not seed:

        seed = nacl.utils.random(32)

    signing_key = nacl.signing.SigningKey(seed=seed)

    private_key = signing_key.to_curve25519_private_key()

    return {'sign': signing_key,

            'sign_b58': base58.b58encode(signing_key.encode()),

            'verify': signing_key.verify_key,

            'verify_b58': base58.b58encode(signing_key.verify_key.encode()),

            'private': private_key,
```

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

```
'private_b58': base58.b58encode(private_key.encode()),

'public': private_key.public_key,

'public_b58': base58.b58encode(private_key.public_key.encode()),

'seed': seed}

def create_keypair(name):

    filename = '{}.b58db_seed'.format(name)

    seed = nacl.utils.random(32)

    with open(filename, 'wb') as fh:

        fh.write(seed)

def load_keypair(name):

    filename = '{}.b58db_seed'.format(name)

    with open(filename, 'rb') as fh:

        seed = fh.read()

    return keypair(seed)

def resolve(name):

    try:

        return load_keypair(name)['verify_b58']

    except FileNotFoundError:

        return name
```

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## 5.3 SOFTWARE ENVIRONMENT:

### What is Python:

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python programming supports both procedural and object-oriented paradigms. Compared to other programming languages like Java, Python programs are typically smaller.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

Python's largest asset is its vast library of standard libraries, which can be used for the following.

- GUI Applications (like Kivy, Tkinter, PyQtetc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

### FEATURES OF PYTHON:

#### Extensive Libraries

A vast collection of code for a variety of uses, including regular expressions, documentation creation, unit testing, web browsers, databases, CGI, email, image manipulation, and more, can be downloaded in Python. Therefore, we may avoid writing the entire code by hand for that.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

As seen previously, Python is extensible to other languages. Some of your code can be written in languages like C++ or C. This is useful, particularly for projects.

## **Integrable**

In addition to being extensible, Python may also be embedded. It is possible to include Python code within the source code of another language, such as C++. This enables us to give our code written in the other language scripting capabilities.

## **Enhanced Output**

Compared to languages like Java and C++, programmers are more productive with this language because of its ease of use and large library. In addition, you should write less and do more tasks.

## **Opportunities for IOT**

Python has a bright future for the Internet of Things since it serves as the foundation for emerging platforms like Raspberry Pi.

## **Simple and Easy**

It could be necessary to construct a class when dealing with Java in order to print "Hello World." In Python, however, a simple print statement suffices. It is also very simple to learn, comprehend, and write code for. This is the reason why people who learn Python find it difficult to transition to more verbose languages like Java.

## **Readable**

Reading Python is similar to reading English since the two languages are not as verbose. That's why learning, comprehending, and coding it is so simple. Curly brackets are not required to define blocks, and indentation is required. This improves the code's readability much further.

## **Object-Oriented**

The object-oriented and procedural programming paradigms are supported by this language. Classes and objects allow us to model the actual world, whereas functions aid in the reusability of code. It is possible to combine functions and data into a single class.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## **Free and Open-Source**

As we previously stated, Python is open source. Not only is Python available for free download, but you may also obtain its source code, modify it, and even share it. It comes with a large library of downloads to assist you in your work. Transportable If you wish to execute your C++ project on a different platform, you might need to make some changes to its code. With Python, though, things are different. You only have to write the code once, and it may be executed anywhere. Write Once Run Anywhere (WORA) is the term for this. You must take care, though, to ensure that no system-dependent elements are included.

## **Advantages of Python Over Other Languages:**

When the identical activity is done in another language, almost all tasks done in Python require less coding. Additionally, Python offers fantastic built-in library support, so you won't need to look for any other resources to complete your task. For this reason, a lot of people. Since Python is free, anyone may use it to create apps, be it an individual, a small business, or a large enterprise. Python offers you greater community support because it is extensively used. According to the 2019 GitHub annual survey

## **Python is for Everyone**

Python code is compatible with Mac, Windows, and Linux operating systems. Programmers need to learn multiple languages for different tasks, but Python allows you to create games and complex visualizations in addition to professionally building web apps, automating tasks, performing machine learning and data analysis, and web scraping. This programming language is versatile.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## **Disadvantages of Python:**

We've shown thus far why Python is a fantastic option for your project. But you should also be conscious of the implications if you decide to go that route. Now let's look at the drawbacks of selecting Python over other languages.

### **Speed Limitations**

Python code is run line by line, as we have seen. However, Python's interpreted nature frequently leads to sluggish execution. But this isn't an issue unless the project's main focus is speed. To put it another way, Python's advantages outweigh its speed constraints until very high speeds are required.

### **Weak in Mobile Computing and Browsers**

Python is a rather uncommon client-side language, while being a great server-side language. Aside from that, implementing smartphone-based applications is a rare occurrence. The Carbon Nelle application is one example of this. Despite Bryton's presence, it isn't as well-known because it isn't very safe.

### **Design Restrictions**

Python, as you are aware, uses dynamic typing. This implies that when writing the code, you are not need to define the type of variable. It typesets with ducks. But hold on—what's that? It simply says that something must be a duck if it appears to be one. Although this makes coding simpler for the programmers, run-time mistakes may result from it.

### **Underdeveloped Database Access Layers**

Python's database access layers are a little less advanced than those of more popular technologies like JDBC (Java Database Connectivity) and ODBC (Open Database Connectivity). As such, it is used less frequently in large corporations.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## Easy

We are not joking. The simplicity of Python can provide a challenge. Consider my illustration. My preference is for Python over Java. Its grammar seems so basic to me that I don't think the verbosity of Java code is necessary. This article focused solely on the benefits and drawbacks of the Python programming language.

## History of Python:

What is the connection between Python programming language and the alphabet? Yes, both begin with ABC. Clearly, the programming language ABC is meant if we discuss ABC within the context of Python. ABC is a general-purpose programming language and environment that was created at the CWI (Centrum Wiskunde & Informatica) in Amsterdam, the Netherlands. The biggest contribution ABC made to Python's design was its influence. The idea for Python originated in the late 1980s. During that period, Guido van Rossum was employed by the CWI on a distributed operating system project named Amoeba. Guido van Rossum stated in an interview with Bill Venners<sup>1</sup>: "I worked as an implementer at Centrum voor Wiskunde en Informatica in the early 1980s on a team that I'm not sure how widely acknowledged ABC's impact on Python is. I make an effort to acknowledge the impact of ABC as I owe a lot to the individuals who worked on the project and to everything I learnt from it." Guido van Rossum added later in the same interview: "I recalled everything I had experienced, including some of my frustrations with ABC. I made the decision to attempt creating a basic scripting language with some of ABC's better features but none of its drawbacks. I then began to type. I made a basic runtime, a basic parser, and a basic virtual machine. I took my favorite portions from ABC and made my own rendition of them. I made a simple syntax, grouped statements using indentation rather than curly brackets or begin-end blocks.

## Python Development Steps:

In February 1991, Guido Van Rossum released version 0.9.0 of the Python code at Altisource. Functions, exception handling, and the fundamental data types like lists, dict,



# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

str, and others were already present in this release. It featured a module system and was object oriented as well. January 1994 saw the introduction of Python version 1.0. The functional programming tools lambda, map, filter, and reduce—which Guido Van Rossum never liked—were the main new features added to this edition. October 2000, six and a half years later, saw the release of Python 2.0.

List comprehensions, a comprehensive garbage collector, and Unicode support were all included in this edition. Python had further success in versions 2.x for a further 8 years, until the next major release, Python 3.0 (sometimes referred to as "Python 3000" and "Py3K"), was released. Python 2.x and Python 3 are not backwards compatible. The goal of Python 3 was to eliminate redundant programming techniques and modules, which partially or fully complies with the 13th law of the Zen of Python, which states that "there should be one and preferably only one obvious way to do it." A few adjustments to Python 7.3.

- Views and iterators instead of lists
- Ordering rules for comparisons have been streamlined. For example, since each element in a list needs to be comparable to every other element, a heterogeneous list cannot be sorted.
- There is only one more integer type, which is int. Long is also int.
- The division of two integers returns a float instead of an integer. "/" can be used to have the "old" behavior.
- Text Vs. Data Instead of Unicode Vs. 8-bit

## **Purpose:**

We showed that, with the help of the ANIS feature, our method allows successful intra-retinal layer segmentation even with low-quality pictures that have speckle noise, poor contrast, and varying intensity ranges throughout.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## **Python:**

An interpreted, high-level language for general-purpose use is called Python. Python was created by Guido van Rossum and released for use in 1991. . Its design philosophy prioritizes code readability and makes extensive use of whitespace.

Python has automated memory management and a dynamic typing system. It features a sizable and extensive standard library and supports a variety of programming paradigms, such as imperative, functional, procedural, and object-oriented programming.

Interpretation of Python: Python is runtime processed by the interpreter. You can run your software without first having to assemble it. This makes me think about PERL and PHP.

Python is interactive, meaning that you can write programs by simply interacting with the interpreter while seated at a Python prompt. Python also recognizes the significance of development pace.

This includes having access to strong constructs that prevent boring code duplication as well as readable and concise code. Maintainability, which is closely related to this statistic, may be almost meaningless, but it does provide insight into the amount of code that has to be scanned, read, and/or comprehended in order to diagnose issues or adjust behavior. Another area in which Python shines is in the speed at which it is developing, the simplicity with which programmers from other languages can learn the fundamentals of Python, and the size of the standard library. All of its tools were easy to use, saved a ton of time, and many of them could be upgraded and fixed later on by non-Python experts without breaking

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## **Modules Used in Project:**

### **TensorFlow**

A free and open-source software library called TensorFlow is used for differentiable programming and dataflow in a variety of applications. It is a symbolic math library that is also utilized in neural network and other machine learning applications. At Google, it's utilized for both research and manufacturing. The Google Brain team created TensorFlow primarily for internal use at Google. On November 9, 2015, it was made available to the public under the Apache 2.0 license.

### **NumPy**

A general-purpose array processing package is called NumPy. It offers tools for manipulating these arrays as well as a high-performance multidimensional array object. This is the core Python module for scientific computing. It has a number of characteristics, some of which are significant:

- An effective N-dimensional array entity Advanced broadcasting functionalities Integrating C/C++ and Fortran code tools Practical functions for linear algebra, Fourier transform, and random numbers

In addition to its apparent applications in science, NumPy can be used as a productive multi-dimensional data container for general purposes. NumPy can create any data-types, which makes it possible to quickly and easily connect NumPy with a large range of databases.

### **Pandas**

Using its robust data structures, Pandas is an open-source Python library that offers high-performance data manipulation and analysis capabilities. Python was mostly utilized for preprocessing and data munging. It didn't really contribute much to the examination of data. Pandas figured out the solution to this. Regardless of the source of the data, we may use Pandas to complete five common phases in data processing and analysis: load, prepare,

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

modify, model, and analyze. Numerous academic and professional subjects, including finance, economics, statistics, analytics, and other areas, employ Python with Pandas.

## **Matplotlib**

Matplotlib is a Python 2D plotting package that generates figures of publishing quality in a range of hardcopy formats and cross-platform interactive settings. Web application servers, Jupyter Notebooks, Python scripts, the Python and I Python shells, and four graphical user interface toolkits can all use Matplotlib. Matplotlib aims to make difficult things feasible and easy things easy. With just a few lines of code, you can create plots, histograms, power spectra, bar charts, error charts, scatter plots, and more. See the thumbnail gallery and sample graphs for examples. The pilot module offers a MATLAB-like interface for basic charting, especially when paired with Python. With the power user, you may fully customize axes attributes, line styles, font characteristics, and etch through an object-oriented interface or a collection of functions that are common to MATLAB users.

## **Scikit – learn**

Scikit-learn provides a standardized Python interface along with a range of supervised and unsupervised learning methods. It is licensed under a permissive simplified BSD license that supports both commercial and academic use, and it is available under numerous Linux versions.

## **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 JPython enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## **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. Python's object-oriented methodology and linguistic structure help programmers create rational and understandable code for projects. There is no pre-packaged version of this software for Windows.

## **Using Python on a Mac or Windows computer:**

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. you should know about the system requirements

Based on your system type i.e. operating system and based processor, you must download the python version. The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## 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>



**Fig 5.2.1: Official site to download Python**

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

**Step 2:** Click on the Download Tab.

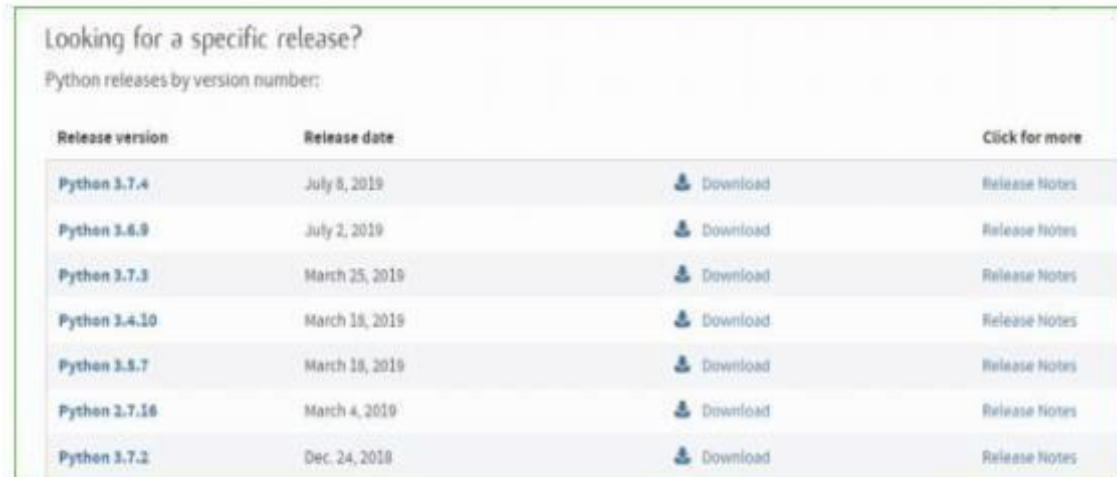


**Fig 5.2.2: Downloading Python**








# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

**Step 3:** You have two options: either scroll down and click on the download button that corresponds to the version, or select the Python for Windows 3.7.4 button that is highlighted in yellow. Here, we're downloading the most recent Windows 3.7.4 Python version.



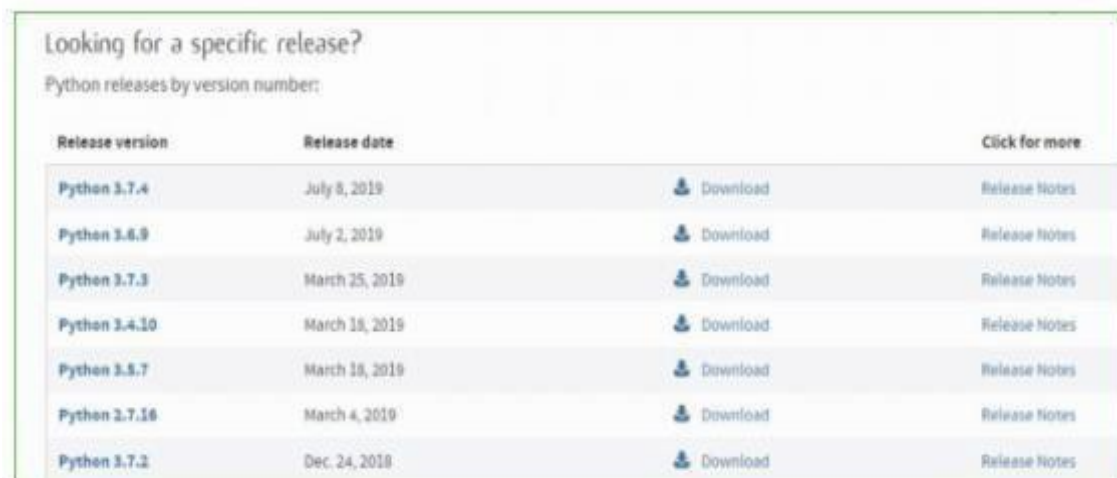
Looking for a specific release?  
Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	 Download	<a href="#">Release Notes</a>
Python 3.6.9	July 2, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.3	March 25, 2019	 Download	<a href="#">Release Notes</a>
Python 3.4.10	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 3.5.7	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 2.7.16	March 4, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.2	Dec. 24, 2018	 Download	<a href="#">Release Notes</a>








**Fig 5.2.3: Python Versions**

**Step 4:** Find the Files option by scrolling down the page.

**Step 5:** This displays an alternate Python version in addition to the OS.



Looking for a specific release?  
Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	 Download	<a href="#">Release Notes</a>
Python 3.6.9	July 2, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.3	March 25, 2019	 Download	<a href="#">Release Notes</a>
Python 3.4.10	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 3.5.7	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 2.7.16	March 4, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.2	Dec. 24, 2018	 Download	<a href="#">Release Notes</a>

**Fig 5.2.4: Python Versions**

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

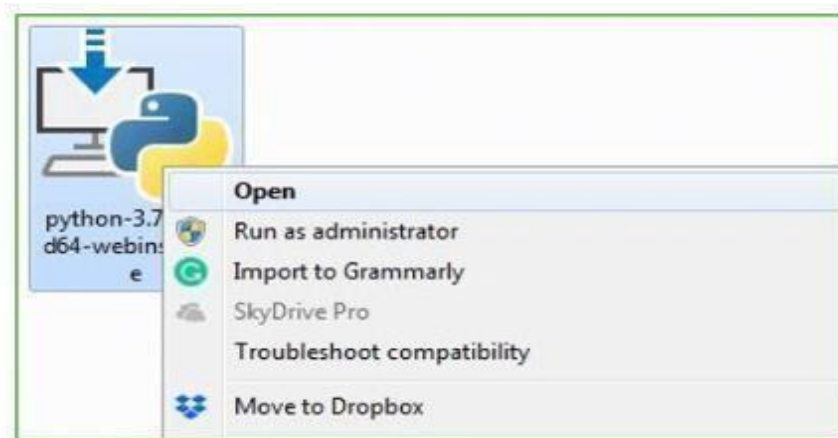
---

- 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 x86web-based installer.
- You have three alternatives to download Windows 64-bit Python: Windows x86-64 web-based installer, Windows x86-64 executable installer, or Windows x86-64 embeddable zip file.

**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:** To begin the installation procedure, click Download and open the Python version that has been downloaded.



**Fig 5.2.5: Opening Python downloaded file**

**Step 2:** Before you click on Install Now, make sure to put a tick on Add Python 3.7



# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---



**Fig 5 .2.6: Installing Python**

**Step 3:** Click on Install NOW After the installation is successful. Click on Close.



**Fig 5.2.7:Setting UP Python**

You have successfully and correctly installed Python by following the preceding three installation procedures. It's time to check the installation now.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

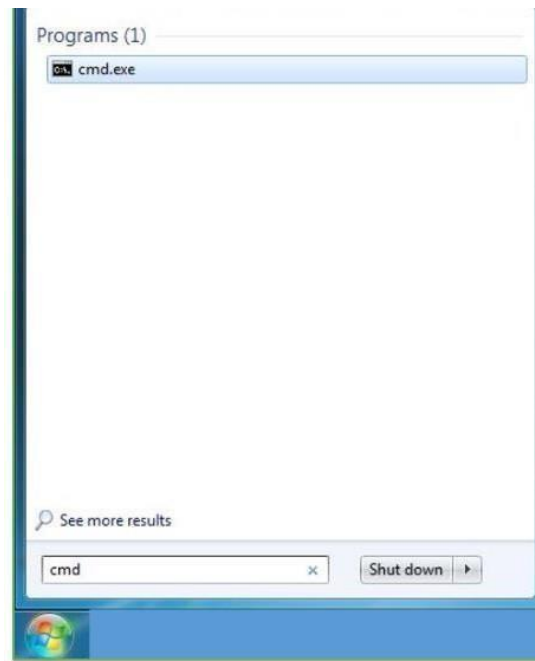
---

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

Verify the Python Installation

**Step 1:** Click on Start

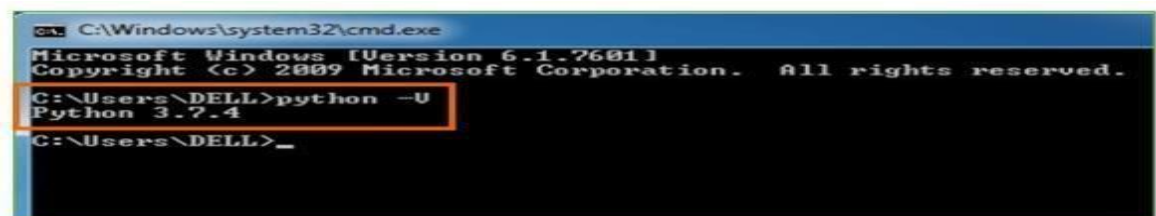
**Step 2:** In the Windows Run Command, type “cmd”.



**Fig 5.2.8: Opening command prompt**

**Step 3:** Open the Command prompt option.

**Step 4:** Let's check to see if Python has been installed correctly. Enter "python -V" after typing it.



**Fig 5.2.9: Python version checking**

**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.

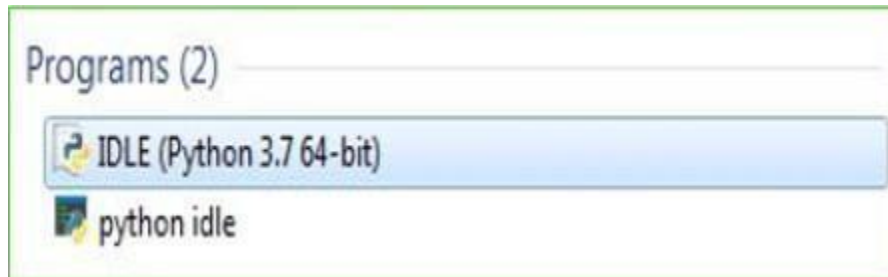
# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

Check how the Python IDLE works

**Step 1:** Click on Start

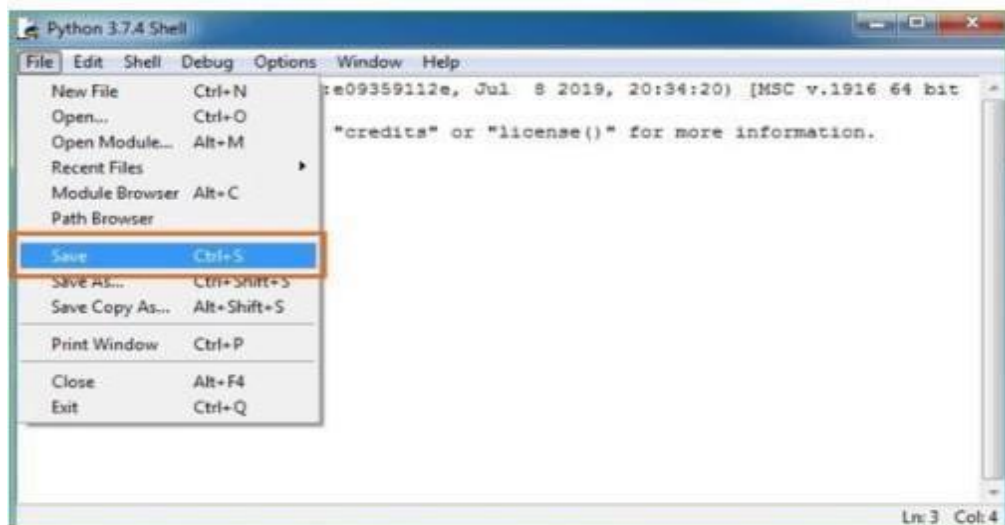
**Step 2:** In the Windows Run command, type “python idle”.



**Fig 5.2.10: Opening python idle**

**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**



**Fig 5.2.11: Saving python file**

**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.**

## CHAPTER 6

### TESTING

#### 6.1 TESTING STRATEGIES:

##### **UNIT TESTING:**

Unit testing is a type of testing where the developer tests individual modules to see if there are any problems. It is focused on the standalone modules' functional accuracy. The primary goal is to separate every component of the system in order to locate, examine, and correct. Black box testing is a technique used to test the input, output, and user interface. White Box Testing: This technique is used to evaluate the behavior of each of those functions.

##### **DATA FLOW TESTING:**

A group of testing techniques known as data flow testing are focused on choosing routes through the program's control flow to investigate the series of events connected to the status of variables or data objects. The points in time at which variable acquire and the points at which these values are used are the primary concerns of dataflow testing

##### **INTEGRATION TESTING:**

Unit testing is followed by integration testing since the units or modules need to be integrated, which raises the need for integration testing. Verifying the functionality, performance, and dependability of the integrated modules is the aim of integration test

##### **BIG BANG INTEGRATION TESTING:**

Big Bang Integration Testing is an approach to integration testing in which every unit is connected simultaneously to form a whole system. Adopting this kind of testing technique makes it harder to pinpoint any flaws that are discovered because the interfaces.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

## TESTING USER INTERFACES:

A product or piece of software is put through user interface testing, which uses a graphical user interface (GUI) to detect any flaws.

### 6.1 TEST CASES:

S.NO	INPUT	If available	If not available
1	New user signup	user can sign up with the application	There is no process
2	User login	user can login to application	There is no process
3	Post Private Messages	private and public key will be generated and then encrypt message and save in Blockchain	There is no process
4	View Shared Private Message Blockchain	owner or share users can decrypt and view all messages	There is no process

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

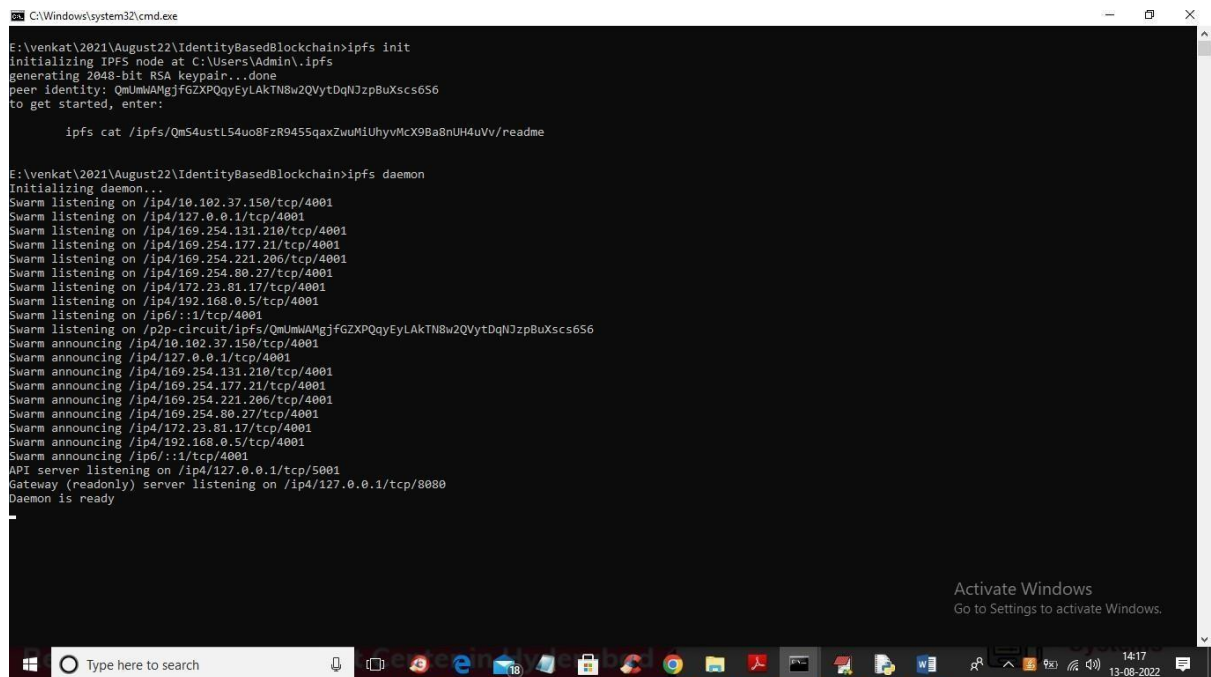
## CHAPTER 7

### RESULTS

#### Output screens:

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

output



```
E:\venkat\2021\August22\IdentityBasedBlockchain>ipfs init
initializing IPFS node at C:\Users\Admin\.ipfs
generating 2048-bit RSA keypair...done
peer identity: QmUmwAMgjFGZXPQqyEYLAKTN8w2QVytDqNjzpBuXscs6S6
to get started, enter:

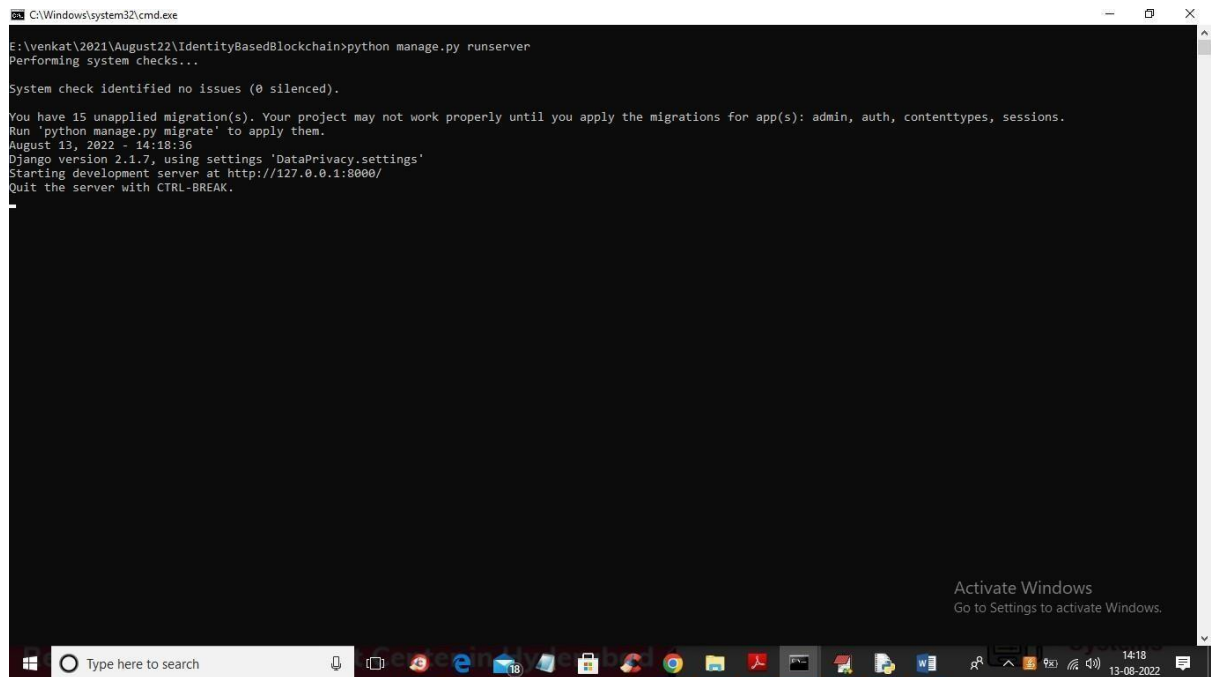
    ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMCX9Ba8nUH4uVv/readme

E:\venkat\2021\August22\IdentityBasedBlockchain>ipfs daemon
initializing daemon...
Swarm listening on /ip4/10.102.37.150/tcp/4001
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/169.254.131.210/tcp/4001
Swarm listening on /ip4/169.254.177.21/tcp/4001
Swarm listening on /ip4/169.254.221.206/tcp/4001
Swarm listening on /ip4/169.254.80.27/tcp/4001
Swarm listening on /ip4/172.23.81.17/tcp/4001
Swarm listening on /ip4/192.168.0.5/tcp/4001
Swarm listening on /ip6:::1/tcp/4001
Swarm listening on /p2p-circuit/ipfs/QmUmwAMgjFGZXPQqyEYLAKTN8w2QVytDqNjzpBuXscs6S6
Swarm announcing /ip4/10.102.37.150/tcp/4001
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/169.254.131.210/tcp/4001
Swarm announcing /ip4/169.254.177.21/tcp/4001
Swarm announcing /ip4/169.254.221.206/tcp/4001
Swarm announcing /ip4/169.254.80.27/tcp/4001
Swarm announcing /ip4/172.23.81.17/tcp/4001
Swarm announcing /ip4/192.168.0.5/tcp/4001
Swarm announcing /ip6:::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

**Fig 7.1.1: Starting of IPFS Server**

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

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain



**Fig 7.1.2: URL is Displayed**

In above screen python 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 home page



**Fig 7.1.3: URL is opened in the browser**

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

In above screen click on 'New User Signup Here' link to signup user

Protecting Data Privacy for Perm x +

127.0.0.1:8000/Signup.html

For quick access, place your bookmarks here on the bookmarks bar. [Import bookmarks now...](#)

Home User Login Here New User Signup Here

**Data Privacy & Blockchain**

**User Signup Screen**

Username

Password

Contact No

Gender

Email ID

Address

Activate Windows  
Go to Settings to activate Windows.

Type here to search

14:21 13-08-2022

**Fig 7.1.4: New User Signup**

In above screen user is entering signup details and press submit button to store details In Blockchain and get below output

Protecting Data Privacy for Perm x +

127.0.0.1:8000/SignupAction

For quick access, place your bookmarks here on the bookmarks bar. [Import bookmarks now...](#)

Home User Login Here New User Signup Here

**Data Privacy & Blockchain**

**User Signup Screen**

Signup process completed and record saved in Blockchain

Username

Password

Contact No

Gender

Email ID

Address

Activate Windows  
Go to Settings to activate Windows.

Type here to search

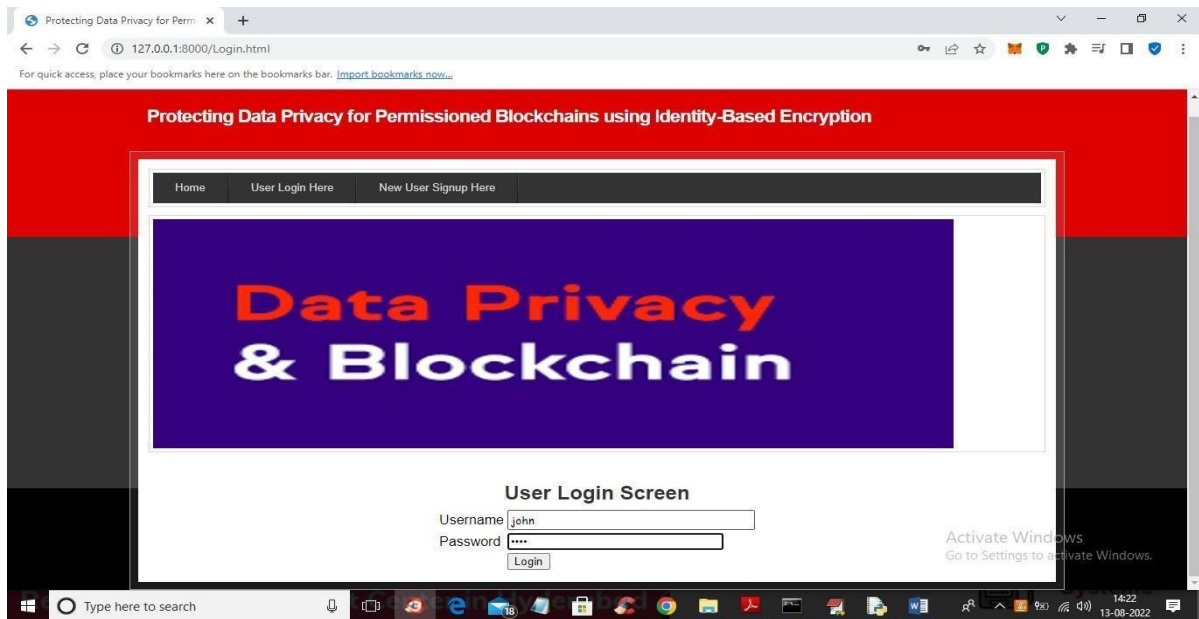
14:21 13-08-2022

**Fig 7.1.5: Storing of User Details in Blockchain**

In above screen user signup completed and details saved in Blockchain and now click on 'User Login' link to get below screen



# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain



**Fig 7.1.6: User Login**

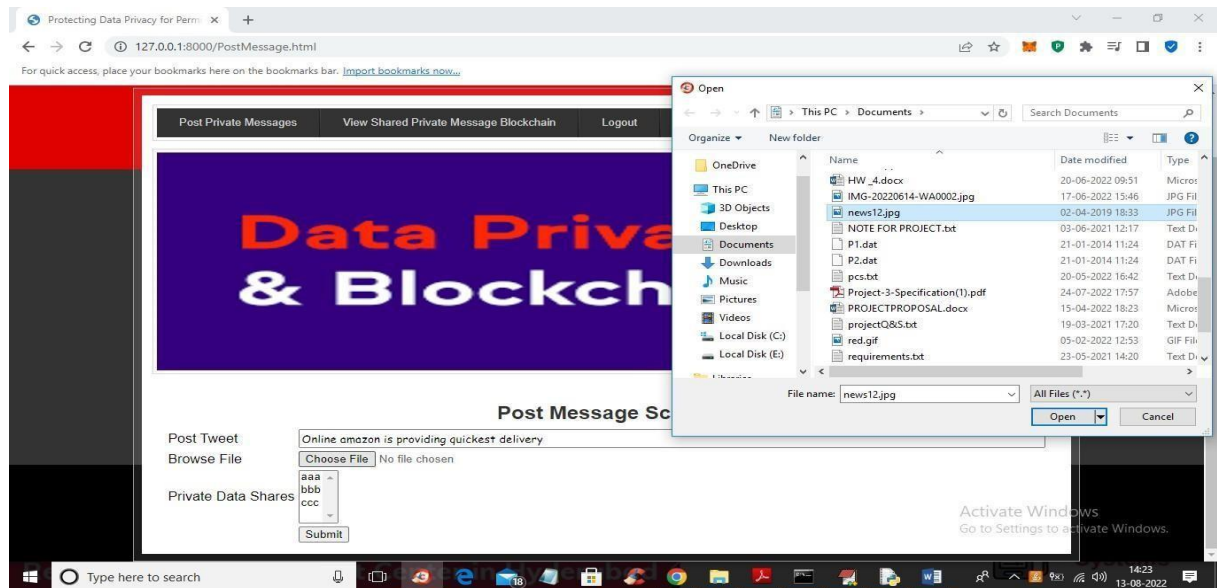
In above screen user is login and after login will get below screen



**Fig 7.1.7: Login Page**

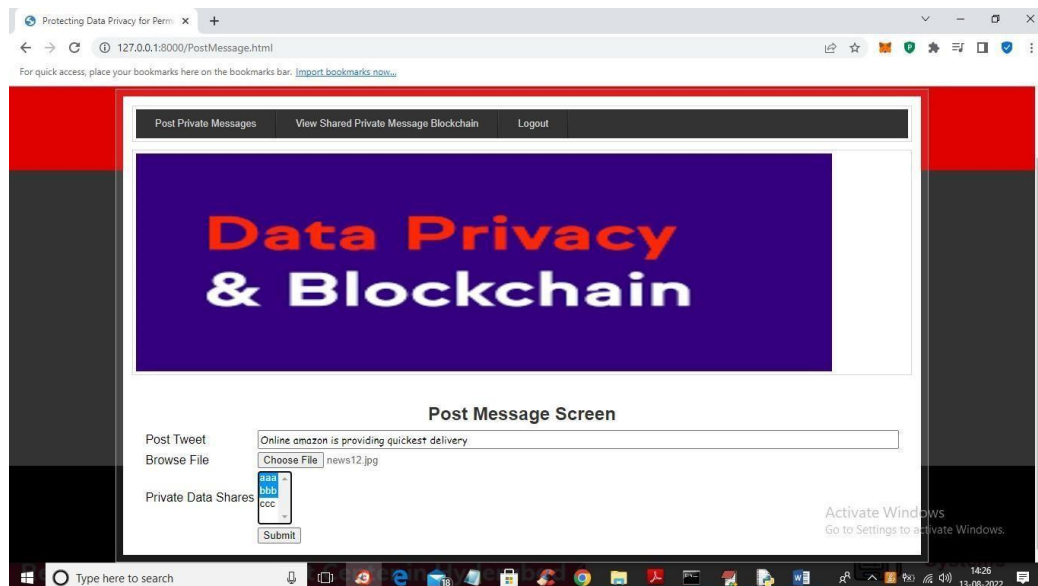
# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

In above screen user can click on 'Post Private Messages' link to upload message



**Fig 7.1.8: Posting Private Messages**

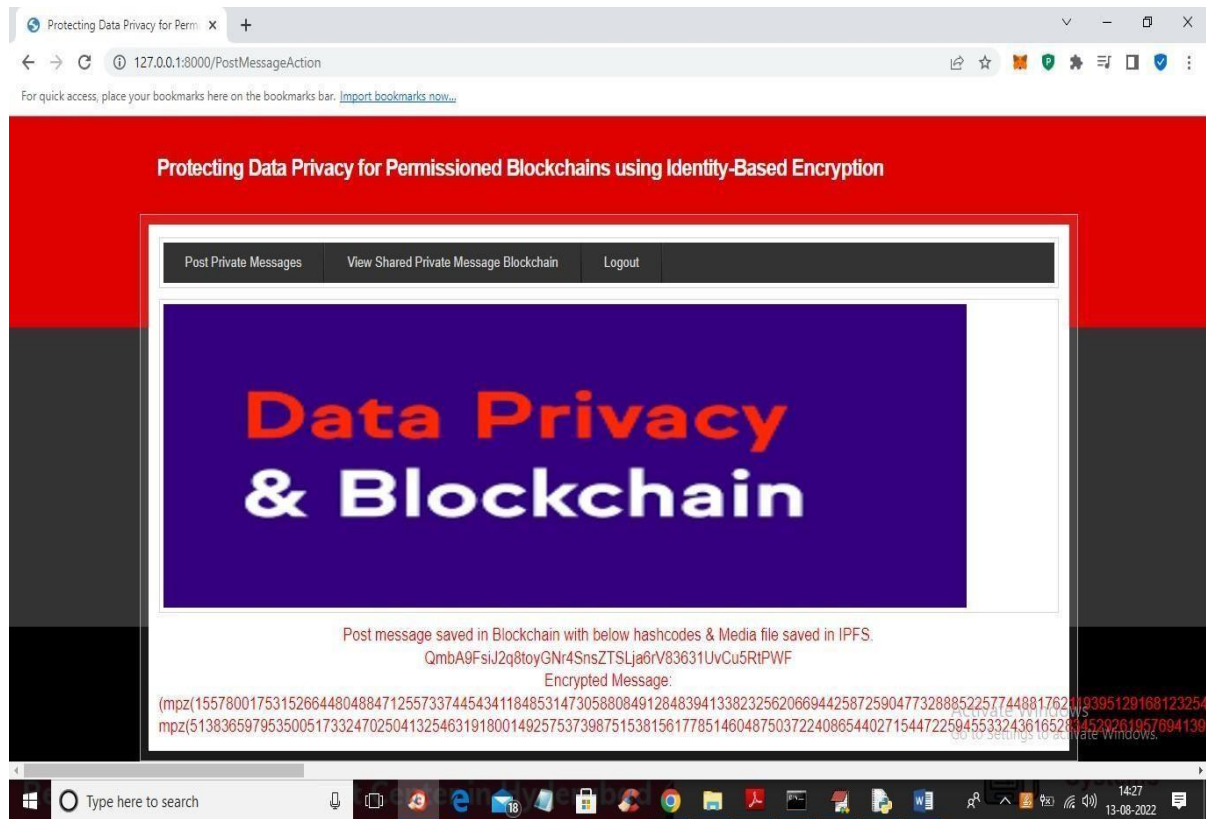
In above screen user type some message and then uploading image and then select list of users to share with and you can select multiple users by holding CTRL key like below screen



**Fig 7.1.9: Posting Messages**

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

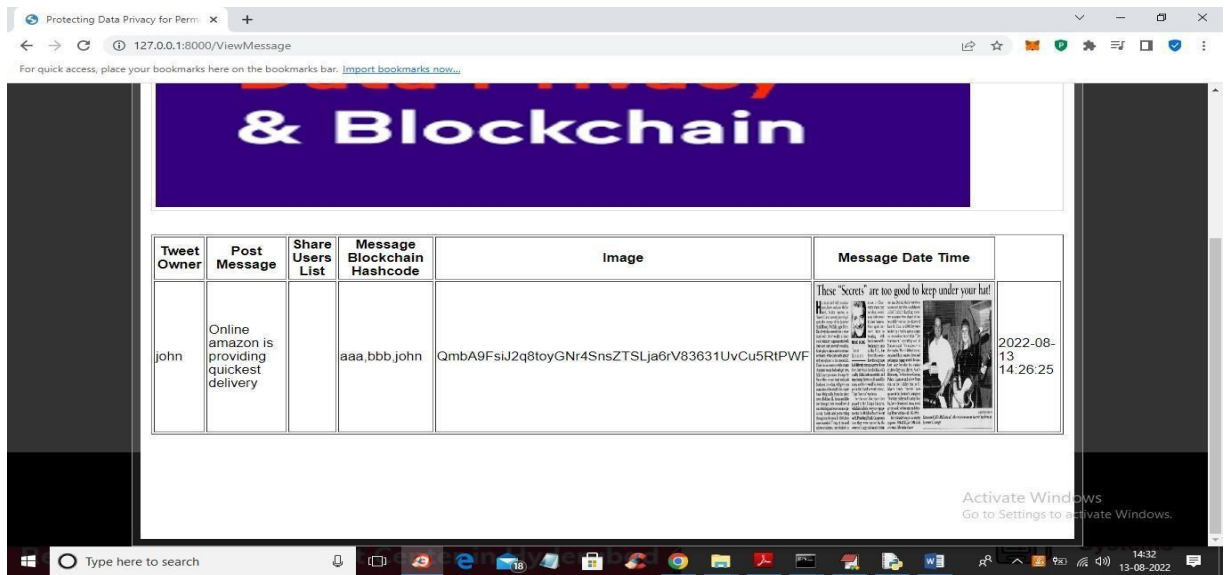
In above screen user John is uploading some post and then giving share access to user 'aaa and bbb' and user 'ccc' cannot access and now press 'submit' button to save post in Blockchain and get below output



**Fig 7.1.10: Posted Messages saved in Blockchain**

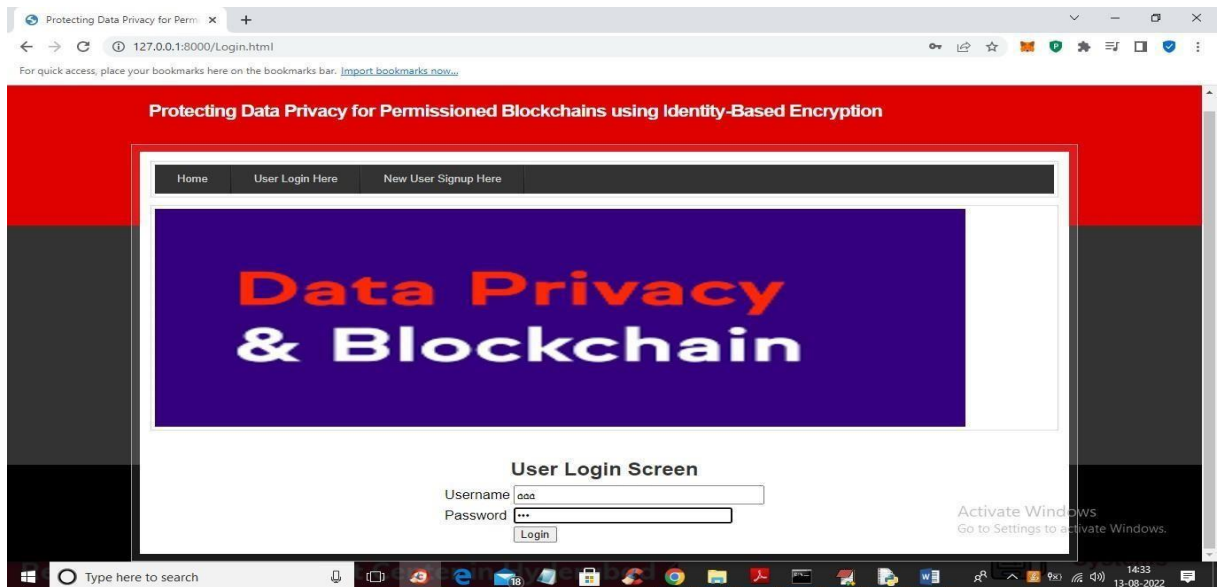
In above screen we can see message in red color as POST MESSAGE saved in Blockchain and with Hash code and we can see IBE encrypted message and now click on ‘View Shared Private Message Blockchain’ link to view message in decrypted format

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain



**Fig 7.1.11: Messages Stored as Blocks**

In above screen use can view decrypted message with image and hash code and this user hash shared post with user 'aaa' and now we login as 'aaa' and check message



**Fig 7.1.12: User Login**

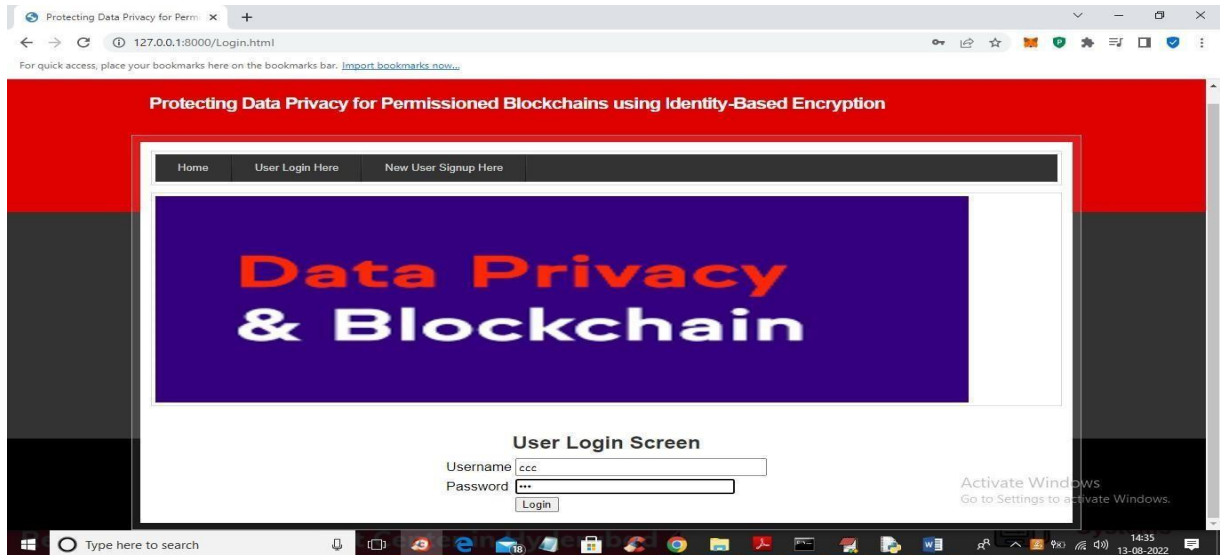
In above screen user 'aaa' is login and after login will get below screen

---

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

In above screen user aaa can view all his and shared messages and now we will login as user 'ccc' and check message as this user has no sharing permission



**Fig 7.1.15: User Login**

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



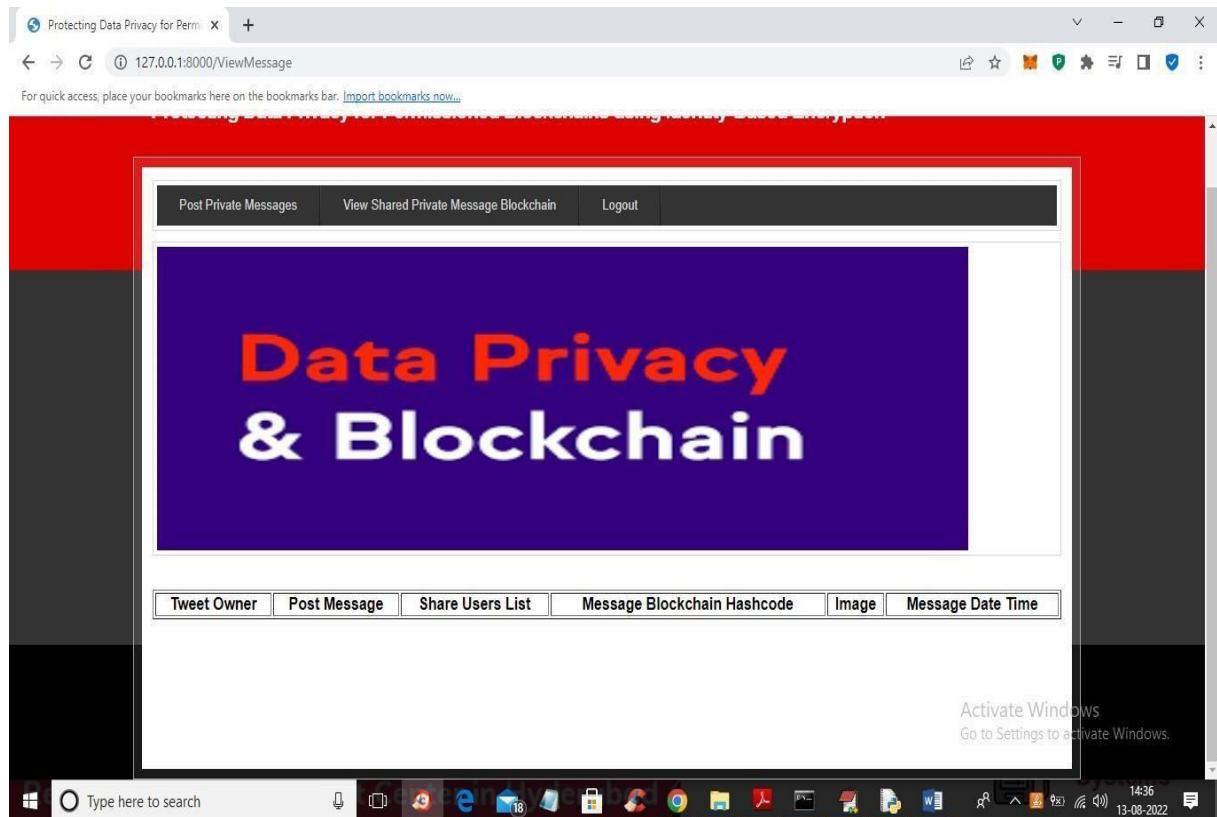
**Fig 7.1.16: Login Page**

In above screen user can click on 'View Shared Message' link to view messages



# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---



**Fig 7.1.17: Viewing messages**

In above screen we can see user CCC has no share permission so he cannot decrypt and view messages and privacy will be achieved

## CHAPTER 8

### CONCLUSION

We have proposed an improved delicately scheme on top of non-transactional cases in permissioned blockchain to improve the privacy. Our scheme can hide the information by encrypting the plaintext into the ciphertext, without using advanced technologies such as ring signature, homomorphic encryption and zero-knowledge proofs. Our scheme not only avoids the complicated certificate management and issuance in the traditional PKI system, but has a high security level which can prevent both disguise and passive attacks, and is functional, effective and practical for applications. This scheme provides an inspiring way to achieve delicate confidentiality of the transactions in many applications for non-transactional scenarios.



## CHAPTER 9

### FUTURE ENHANCEMENTS

- Implement threshold cryptography to distribute the Private Key Generator's (PKG) power across multiple, independent servers. This mitigates the single point of failure (SPOF) risk associated with a centralized PKG and enhances fault tolerance.
- Identity-Based Signatures (IBS): Investigate the use of IBS for data integrity and non-repudiation within the permissioned blockchain. IBS allows users to sign data using their identities, ensuring data authenticity and preventing denial-of-origin attacks.
- Explore alternative data structures within the blockchain (e.g., Merkle Trees) to optimize data storage, retrieval, and verification. This can improve transaction processing efficiency, especially for large datasets.
- Identity Management Integration: Integrate with existing identity management systems (e.g., LDAP, Active Directory) to leverage established user identities and streamline access control within the permissioned blockchain.
- Decentralized Key Management (DKM): Explore DKM solutions that distribute key management across multiple nodes, eliminating the need for a central authority and fostering trust in the system.
- Interoperability with Other Blockchains: Develop mechanisms for secure communication and data exchange between the permissioned blockchain and other blockchain platforms, enabling broader ecosystem integration and collaboration.
- Formal Security Analysis: Conduct rigorous formal security analysis of the enhanced system to identify and address potential vulnerabilities. This analysis should be performed by qualified security experts.
- Usability and User Experience: Design user-friendly interfaces and tools to simplify key generation, access control management, and data encryption/decryption processes for both technical and non-technical users.
- Performance Benchmarks: Establish performance benchmarks to evaluate the enhanced system's efficiency under varying workloads and data sizes. This helps identify areas for further optimization.

## CHAPTER 10

### APPLICATIONS

Identity-based encryption (IBE) in permissioned blockchains offers various applications for enhancing data privacy:

- **Secure Data Sharing:** IBE enables secure sharing of sensitive data within a permissioned blockchain network. Authorized parties can encrypt and decrypt data using their unique identities, ensuring that only intended recipients can access the information.
- **Access Control:** IBE can be used for fine-grained access control, allowing blockchain participants to control who can view or modify specific data. This ensures that sensitive data is only accessible to authorized users, enhancing privacy and confidentiality.
- **Secure Messaging:** IBE can facilitate secure messaging and communication within a permissioned blockchain network. Participants can exchange messages securely using their identities as encryption keys, protecting the confidentiality of communications.
- **Secure Transactions:** IBE can enhance the security of transactions within a permissioned blockchain by encrypting transaction data using the identities of involved parties. This ensures that transaction details remain confidential and protected from unauthorized access.
- **Identity Management:** IBE can streamline identity management within permissioned blockchains by eliminating the need for separate encryption keys for each participant. Participants can use their identities as encryption keys, simplifying key management and reducing overhead.
- **Compliance and Regulatory Requirements:** IBE can help organizations comply with data privacy regulations and industry standards by providing robust encryption mechanisms for protecting sensitive data stored and transmitted within permissioned blockchains. Overall, identity-based encryption plays a crucial role in safeguarding data privacy and enhancing security within permissioned blockchain environments,

## Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

making it a key technology for various applications in sectors such as finance, healthcare, supply chain, and government.

## CHAPTER 11

### REFERENCES

- [1] The Linux Foundation Helps Hyperledger Build the Most Vibrant Open-Source Ecosystem for Blockchain. <http://www.linuxfoundation.org/>.
- [2] S. Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI Matters*, 1(2):19C21, Dec. 2014.
- [3] D. D. Detwiler. One nation moves to increase food safety with blockchain. <https://www.ibm.com/blogs/blockchain/2018/02/one-nationsmove-to-increase-food-safety-with-blockchain/,2018>. [Online; accessed 1-May-2018].
- [4] Shamir, A. Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47C53. Springer, Heidelberg (1985)
- [5] Boneh, D., Franklin, M. Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213C229. Springer, Berlin, Germany (2001)
- [6] Boneh, D., Boyen, X. Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223C238. Springer, Berlin, Germany (2004)
- [7] Boneh, D., Boyen, X. Secure identity-based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, Springer, Berlin, Germany (2004).
- [8] Gentry, C. Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445C464. Springer, Berlin, Germany (2006).
- [9] Labs, Shen NoetherMrl. Ring confidential transactions. 2016.

# Identity Based Encryption: A Key to Data Privacy in Permissioned Blockchain

---

- [10] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler and M. Walfish. DoublyEfficientzkSNARKs Without Trusted Setup. 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, 2018, pp. 926-943.
- [11] B. Bnz J. Bootle D. Boneh A. Poelstra P. Wuille G. Maxwell. Bulletproofs: Efficient range proofs for confidential transactions”, IEEE S&P May 2018.
- [12] A. Chiesa E. Tromer M. Virza. Cluster computing in zero knowledge, EUROCRYPT Apr. 2015.
- [13] A. Chiesa M. A. Forbes N. Spooner. A zero knowledge sumcheck and its applications. CoRR abs1704.02086 2017.
- [14] T. P. Pedersen et al. Non-interactive and information-theoretic secure verifiable secret sharing. in Crypto, vol. 91, pp. 129C140, Springer, 1991.
- [15] P. Paillier et al. Public-key cryptosystems based on composite degree residuosity classes. in Eurocrypt, vol. 99, pp. 223C238, Springer, 1999

# ANNEXURES

# IDENTITY BASED ENCRYPTION:A KEY TO DATA PRIVACY IN PERMISSIONED BLOCKCHAIN

S. Reshma<sup>1,a)</sup>, N.Ramadevi<sup>1,b)</sup>, S.Saliya<sup>1</sup>, B.Sravani<sup>1</sup>, S.Asifa<sup>1</sup>, U.Vaishnavi<sup>1</sup> and T.Sharada<sup>1</sup>

<sup>1</sup> Dept of Computer Science and Engineering (Data Science),

Santhiram Engineering College, Nandyal,

518501, India.

a) Corresponding Author:

reshma.cse@srecnandyal.edu.in

b) ramadevi.cse@srecnandyal.edu.in

**Abstract:** The emergence of blockchain technology has garnered significant attention due to its decentralized architecture and distributed public ledger, notably underlying Bitcoin. Governments, financial institutions, and high-tech enterprises are exploring its potential to enhance efficiency, reduce costs, and bolster data security. However, serious privacy concerns persist, potentially impeding widespread adoption. In this paper, we introduce a practical solution leveraging Identity-Based Encryption (IBE) to address these privacy issues in non-transactional applications. Our proposed scheme effectively enhances data privacy, offering protection against disguise and passive attacks. Through rigorous analysis, we demonstrate the high-security level of our proposal, showcasing its functionality, effectiveness, and practicality across various non-transactional scenarios.

*Index Terms*—Permissioned Blockchain, Privacy Protection, Bilinear Map, Identity-Based Encryption

## 1. INTRODUCTION

In today's interconnected digital landscape, ensuring data privacy has become a paramount concern for organizations across various sectors. With the proliferation of centralized systems and reliance on conventional verification methods, such as SMS or

email, traditional approaches to safeguarding data have proven inadequate in addressing the evolving threats to privacy and security. As a result, there is a pressing need for innovative solutions that not only enhance data privacy but also ensure secure data sharing among authorized users, while maintaining data integrity and confidentiality. Blockchain technology has emerged as a promising solution to these challenges, offering a decentralized and transparent framework for securely recording and sharing data. Initially introduced as the underlying technology for Bitcoin, blockchain has garnered widespread attention from governments, financial institutions, and high-tech enterprises due to its potential to revolutionize various industries through enhanced efficiency, reduced costs, and improved data security [1]. Unlike traditional centralized systems, where data is stored in a single vulnerable location, blockchain distributes data across multiple computers (nodes), making it significantly more difficult for malicious actors to tamper with or compromise the system [2].

The blockchain operates as a digital ledger that records transactions in a secure and transparent manner. Each transaction is encapsulated in a block of data, which is linked to the previous block through a unique cryptographic hash, forming a chain of blocks [3]. This

decentralized architecture not only enhances security but also promotes transparency, as all transactions are recorded and visible to authorized users in real-time. Furthermore, blockchain ensures data immutability, meaning once a transaction is recorded in the blockchain, it cannot be altered or deleted [4]. These inherent properties of blockchain make it an attractive solution for addressing privacy concerns and ensuring data integrity in various applications.

However, despite its potential benefits, blockchain technology is not without its challenges, particularly concerning data privacy. In permissioned blockchains, where access to data is restricted to authorized users, ensuring privacy while maintaining data integrity is crucial. Traditional encryption methods, such as public-key cryptography, can provide a level of data security but may introduce complexities in key management and access control [5]. Moreover, relying solely on encryption may not be sufficient to address privacy concerns in permissioned blockchains, where the identities of users need to be securely managed and authenticated.

To address these privacy challenges effectively, we propose the incorporation of Identity-Based Encryption (IBE) into permissioned blockchains. IBE is a cryptographic scheme that simplifies key management by using unique identifiers, such as email addresses or user names, as public keys [6]. By integrating IBE into permissioned blockchains, we aim to enhance data privacy while ensuring secure data sharing among authorized users. IBE enables administrators to have precise control over data access, allowing them to specify which users can access particular data based on their identities. This granular access control mechanism not only enhances privacy but also simplifies key management and authentication in permissioned blockchains.

In addition to IBE, our project leverages the Ethereum blockchain for its advanced smart contract

capabilities, which enable secure, transparent, and programmable data storage and management [7]. Ethereum's support for smart contracts allows us to implement complex access control policies and data management workflows, further enhancing the privacy and security of the system. Furthermore, to handle file storage efficiently, we integrate the InterPlanetary File System (IPFS) into our project. IPFS provides a distributed and efficient file storage solution, allowing us to store and retrieve files securely and reliably [8].

Overall, our project aims to address the pressing need for enhanced data privacy in permissioned blockchains by leveraging the combined capabilities of blockchain technology, Identity-Based Encryption, Ethereum smart contracts, and IPFS file storage. By integrating these technologies, we seek to provide a secure, efficient, and privacy-preserving solution for securely sharing data in permissioned blockchains, while ensuring data integrity and confidentiality.

## **2. LITERATURE SURVEY**

Blockchain technology has witnessed widespread adoption and exploration across various sectors due to its potential to revolutionize data management, security, and transparency. As organizations strive to harness the benefits of blockchain while addressing privacy concerns, researchers have proposed innovative solutions and frameworks. In this literature survey, we delve into seminal works and recent advancements in blockchain technology, smart contracts, and identity-based encryption (IBE) to provide insights into the evolving landscape of data privacy and security in permissioned blockchains.

The Linux Foundation has played a pivotal role in fostering the growth of open-source blockchain ecosystems through initiatives like Hyperledger [1]. Hyperledger Fabric, one of the prominent blockchain frameworks under the Hyperledger umbrella, provides a modular and scalable platform for building permissioned



blockchain networks. By leveraging Hyperledger Fabric, organizations can design customized blockchain solutions tailored to their specific requirements, including data privacy and security.

Omohundro (2014) explores the intersection of cryptocurrencies, smart contracts, and artificial intelligence (AI), highlighting the transformative potential of blockchain technology [2]. Smart contracts, programmable self-executing contracts deployed on blockchain networks like Ethereum, enable automated and transparent execution of predefined terms and conditions. Integrating AI with blockchain can further enhance the capabilities of smart contracts, enabling advanced functionalities such as predictive analytics and automated decision-making.

Detwiler (2018) discusses the application of blockchain technology in enhancing food safety and traceability [3]. By leveraging blockchain's immutable ledger, organizations can track and trace the journey of food products from farm to fork, ensuring transparency and accountability throughout the supply chain. This use case underscores the importance of data integrity and privacy in blockchain-based solutions, particularly in sensitive domains like food safety and supply chain management.

In the realm of cryptography, Shamir (1985) introduces identity-based cryptosystems and signature schemes, laying the foundation for Identity-Based Encryption (IBE) [4]. Unlike traditional public-key cryptography, IBE simplifies key management by using unique identifiers, such as email addresses or user names, as public keys. Boneh and Franklin (2001) further elaborate on IBE schemes based on the Weil pairing, offering efficient and secure solutions for data encryption and key distribution [5]. Building upon this work, Boneh and Boyen (2004) propose efficient selective-ID secure IBE schemes without relying on random oracles, addressing practical limitations and enhancing the scalability of IBE systems [6]. Subsequently, Boneh and Boyen (2004) present secure IBE schemes without random oracles,

advancing the state-of-the-art in identity-based encryption [7].

Integrating IBE into permissioned blockchains presents a promising approach to enhancing data privacy and access control. By combining the security benefits of blockchain with the simplicity and efficiency of IBE, organizations can establish fine-grained access policies and securely share sensitive data among authorized users. This integrated approach aligns with the overarching goal of ensuring data privacy and confidentiality in permissioned blockchain networks.

In conclusion, the literature survey highlights the diverse applications and advancements in blockchain technology, smart contracts, and identity-based encryption. From Hyperledger frameworks to innovative cryptographic schemes, researchers and practitioners continue to explore novel approaches to address privacy concerns and enhance data security in permissioned blockchains. By leveraging these insights and advancements, organizations can develop robust and privacy-preserving blockchain solutions tailored to their specific use cases and requirements.

### **3. METHODOLOGY**

#### **a) Proposed Work:**

The proposed system aims to address the limitations of the existing encryption methods by leveraging Identity-Based Encryption (IBE) to enhance data privacy in permissioned blockchain networks. In this system, each participant's unique identity, such as an email address or username, serves as their public key, eliminating the need for complex key management infrastructure.

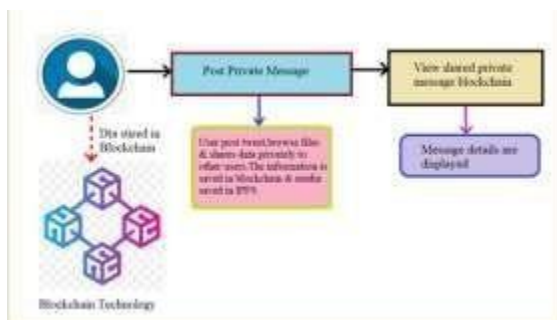
Instead of traditional public keys derived from cryptographic key pairs, the proposed system uses user identities directly as public keys in the encryption process.

Unlike centralized key management systems, the proposed system utilizes decentralized key generation mechanisms to ensure the security and integrity of cryptographic keys.

By leveraging IBE, the proposed system enhances user privacy by preventing unauthorized parties from linking users' public keys to their real-world identities. This mitigates the risk of identity theft and unauthorized surveillance while preserving the anonymity of participants within the blockchain network.

The proposed system offers a comprehensive solution for enhancing data privacy in permissioned blockchain networks through the adoption of Identity-Based Encryption. By addressing the limitations of existing encryption methods and leveraging the benefits of IBE, the proposed system provides a secure, scalable, and privacy-preserving framework for data sharing and transaction processing within permissioned blockchain ecosystems.

#### **b) System Architecture:**



**Fig1 Proposed Architecture**

The system architecture for managing private messages using blockchain technology comprises several key components. Firstly, a user interface facilitates interaction with the system, enabling users to compose, send, and view private messages securely. Upon composing a message, it undergoes encryption using Identity-Based Encryption (IBE) for enhanced security and privacy. The encrypted message, along with

metadata such as sender, recipient, and timestamp, is then packaged into a transaction. These transactions are validated and added to a block by network participants, leveraging consensus mechanisms like proof of work or proof of stake. Once added to the blockchain, the block becomes an immutable record, ensuring data integrity and preventing tampering. Users can retrieve and view shared private messages by querying the blockchain using appropriate cryptographic keys. Overall, this architecture leverages blockchain's decentralized, transparent, and tamper-resistant nature to provide a secure and efficient platform for managing private messages while preserving user privacy.

#### **c) New User Signup**

This module enables secure user registration within the application. Users input unique credentials like username, email, and password, which undergo validation and are securely stored on the blockchain, ensuring data integrity. Moreover, the module generates individual identity-based encryption key pairs for users, bolstering the security of their interactions within the system.

#### **d) User Login**

This module facilitates secure user authentication for registered accounts. Users input their unique credentials, usually comprising a username and password, to access their accounts. These credentials undergo authentication to verify the user's identity, ensuring only authorized individuals gain entry to the application. By implementing robust authentication mechanisms, the module enhances the security of user logins and protects against unauthorized access.

#### **e) Post Private Messages**

This module enables authenticated users to compose and share private messages securely. Users input message content and recipient details, triggering the generation of encrypted messages using Identity-Based Encryption (IBE) for confidentiality. The module securely stores these encrypted messages and associated data on the

blockchain, ensuring the integrity and privacy of shared information. By leveraging IBE and blockchain technology, the module facilitates secure communication while maintaining data confidentiality.

#### **f) View Shared Private Message**

This module enables message owners and authorized users to safely read shared private messages. It presents a list of shared messages and employs the recipient's private key to decrypt and reveal the content. By utilizing the recipient's private key, the module ensures that only authorized users can access and comprehend the messages, preserving the privacy and confidentiality of the information exchanged within the system.

#### **g) Blockchain Integration**

Ethereum's blockchain serves as a secure repository for various types of data, including user signup information and private messages.

Smart contracts on the Ethereum blockchain enforce predefined rules, improving security and transparency. They validate user actions, ensure data integrity, and enable secure sharing.

Ethereum's blockchain verifies and records all user interactions and transactions, forming a transparent and tamper-proof record. This enables users to easily verify and trace their actions, fostering transparency and accountability across the platform.

Blockchain's decentralization removes the need for a central authority, boosting security. Data isn't vulnerable to a single point of failure, reducing risks and ensuring data resilience.

And Data integrity is maintained in the system through the utilization of the SHA-256 algorithm (Secure Hash Algorithm 256-bit). Each block in a blockchain is linked with a unique Hashcode. These blocks are maintained across multiple nodes or servers. Before storing new records, blockchain verifies the Hashcode of each block.

If any block data is modified, it results in a different Hashcode, triggering security alarms and ensuring the integrity and immutability of the data.

IPFS (InterPlanetary File System) is used for efficient and distributed file storage. Rather than storing bulky files directly on the blockchain, which could be slow and resource-intensive, IPFS is employed. IPFS assigns a unique hash code to each stored file. Importantly, these hash codes are recorded on the blockchain, ensuring that files can be efficiently retrieved while benefiting from the data integrity and security offered by the blockchain.

### **4. EXPERIMENTAL RESULTS**

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



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



In above screen python 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 home page



In above screen click on 'New User Signup Here' link to signup user



In above screen user is entering signup details and press submit button to store details in Blockchain and get below output



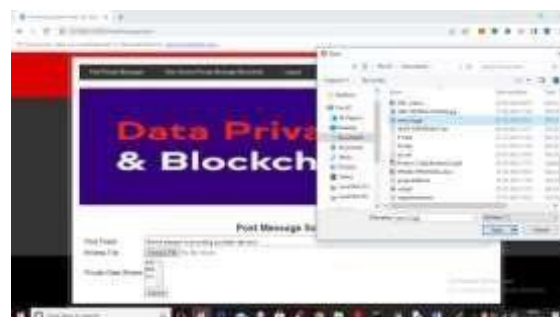
In above screen user signup completed and details saved in Blockchain and now click on 'User Login' link to get below screen



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



In above screen user can click on 'Post Private Messages' link to upload message



In above screen user type some message and then uploading image and then select list of users to share with and you can select multiple users by holding CTRL key like below screen



In above screen user John is uploading some post and then giving share access to user 'aaa and bbb' and user 'ccc' cannot access and now press 'submit' button to save post in Blockchain and get below output





In above screen we can see message in red colour as POST MESSAGE saved in Blockchain and with Hashcode and we can see IBE encrypted message and now click on 'View Shared Private Message Blockchain' link to view message in decrypted format



In above screen click on 'View Shared Message' link to get below output



In above screen use can view decrypted message with image and hashcode and this user has shared post with user 'aaa' and now we login as 'aaa' and check message



In above screen user aaa can view all his and shared messages and now we will login as user 'ccc' and check message as this user has no sharing permission



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



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



In above screen user can click on 'View Shared Message' link to view messages



In above screen we can see user CCC has no share permission so he cannot decrypt and view messages and privacy will be achieved

## **5. CONCLUSION**

The project has significantly enhanced data privacy within permissioned blockchains by addressing security concerns associated with traditional methods, ensuring the confidentiality of sensitive information. Through the integration of Identity-Based Encryption (IBE) and blockchain technology, users can securely share data while maintaining data integrity. Leveraging the Ethereum blockchain establishes a robust and decentralized foundation for transparent data storage, enhancing trust and reliability. The IBE system enables precise control over data access, ensuring only authorized users can decrypt and view shared information. By eliminating reliance on traditional transaction methods and external verification, the project simplifies the data sharing process. Integration with IPFS further expands capabilities, enabling secure storage of various data types, including images. Designed for scalability, the project can seamlessly accommodate growing user numbers and larger data volumes while maintaining optimal performance and security standards.

## **6. FUTURE SCOPE**

Future enhancements could involve integrating Identity-Based Encryption (IBE) with zero-knowledge proof

(ZKP) techniques to further enhance privacy in permissioned blockchains. By combining IBE with ZKP, users could authenticate their identities without revealing sensitive information, ensuring confidentiality while verifying access permissions. This integration could enhance the granularity of access control, allowing for more nuanced data sharing policies within permissioned blockchains. Additionally, integrating ZKP with IBE could mitigate potential vulnerabilities associated with key management and authentication processes, further bolstering the security of the system. Moreover, exploring advancements in cryptographic primitives and consensus mechanisms could contribute to the development of more robust and scalable solutions for privacy-preserving data sharing in permissioned blockchains. Overall, integrating IBE with ZKP represents a promising avenue for future research and development, offering the potential to enhance privacy, security, and usability in permissioned blockchain applications.

## **REFERENCES**

- [1] General Data Protection Regulation (GDPR).[Online]. Available: <https://gdpr-info.eu/art-5-gdpr>.
- [2] G. HILEMAN, M. RAUCHS, 2017 Global blockchain benchmarking study, 2017. [Online]. Available: <https://www.jbs.cam.ac.uk/faculty-research/centres/alternative-finance/publications/global-blockchain>.
- [3] DELOITTE, Breaking blockchain open - Deloitte's 2018 global blockchain survey, 2018. [Online]. Available: <https://www2.deloitte.com/content/dam/Deloitte/cz/Documents/financial-services/cz-2018-deloitte-global-blockchain-survey.pdf>.
- [4] M. EMEM, Facebook's Crypto Division Might Build a Blockchain Identity System: Mark Zuckerberg.

[Online]. Available: <https://www.ccn.com/facebook-crypto-mark-zuckerberg-blockchain>.

[5] Libra White Paper. [Online] Available: <https://libra.org/en-US/white-paper>.

[6] P. DUNPHY, F. A. P. PETITCOLAS, A First Look at Identity Management Schemes on the Blockchain, IEEE Security & Privacy, 16, 4, pp. 20-29, 2018.

[7] J. ROOS, H. NIEDERMAYER, Identity Management on the Blockchain, Seminars FI / IITM SS 18, Network Architectures and Services, 2018.

[8] A. TOBIN, D. REED, The inevitable rise of self-sovereign identity. The Sovrin Foundation, 2016.  
[Online]. Available: <https://sovrin.org/wp-content/uploads/2018/03/The-Inevitable-Rise-of-Self-Sovereign-Identity.pdf>.

[9] A. MÜHLE, A. GRÜNER, T. GAYVORONSKAYA, C. MEINEL, A Survey on Essential Components of a Self-Sovereign Identity, Computer Science Review 30 (2018): 80-86.

[10] X. ZHU, Y. BADR, Identity Management Systems for the Internet of Things: A Survey Towards Blockchain Solutions, Sensors, 18(12):4215, 2018.

[11] P. ANGIN, B. BHARGAVA, R. RANCHAL, N. SINGH, M. LINDERMAN, L.B. OTHMANE, L. LILIEN, An entity-centric approach for privacy and identity management in cloud computing. In Proceedings of the 29th IEEE Symposium on Reliable Distributed Systems (SRDS'10), New Delhi, India, pp. 177–183, 2010.

[12] D. RECORDON, D. REED, OpenID 2.0: A Platform for User-centric Identity Management. In Proceedings of the Second ACM Workshop on Digital Identity Management, Alexandria, VA, USA, 2006.

[13] L. AXON, Privacy-Awareness in Blockchain-Based PKI. Oxford University Research Archive: Oxford, UK, 2015.

[14] M. AL-BASSAM, SCPKI: A Smart Contract-based PKI and Identity System. In Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, Abu Dhabi, UAE, pp. 35–40, 2017.

[15] P. DUNPHY, L. GARRATT, F. PETITCOLAS, Decentralizing Digital Identity: Open Challenges for Distributed Ledgers. 2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), London, pp. 75-78, 2018.

[16] D. van BOKKEM, R. HAGEMAN, G. KONING, L. NGUYEN, N. ZARIN, Self-Sovereign Identity Solutions: The Necessity of Blockchain Technology, 2019. arXiv:1904.12816.

[17] C. ALLEN, The path to self-sovereign identity, 2016.[Online]. Available: <http://www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html>.

[18] R. SOLTANI, U.T. NGUYEN, A.AN, A New Approach to Client Onboarding Using Self-Sovereign Identity and Distributed Ledger. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1129-1136. IEEE, 2018.

[19] T. HARDJONO, A. PENTLAND, Verifiable Anonymous Identities and Access Control in Permissioned Blockchains.[Online]. Available: <https://arxiv.org/pdf/1903.04584.pdf>.

[20] Y. LIU, Z. ZHAO, G. GUO, X. WANG, Z. TAN, S. WANG, An Identity Management System Based on Blockchain, 2017 15th Annual Conference on Privacy, Security and Trust (PST) Calgary, Canada, 2017.

[21] D. AUGOT, H. CHABANNE, T. CHENEVIER, W. GEORGE, L. LAMBERT, A User-Centric System for Verified Identities on the BitcoinBlockchain. In Data Privacy Management, Cryptocurrencies and Blockchain Technology, Springer: Berlin, Germany, pp. 390–407, 2017.

[22] B. FABER, G. C. MICHELET, N. WEIDMANN, R. R. MUKKAMALA, R. VATRAPU, BPDIMS: A Blockchain-based Personal Data and Identity Management System. Proceedings of the 52nd Hawaii International Conference on System Sciences, 2019.

[23] Namecoin. [Online]. Available: <https://namecoin.org>.

[24] Blockstack. [Online]. Available: <https://blockstack.org>.

[25] uPort. [Online]. Available: <https://www.uport.me>.

[26] SOVRIN-FOUNDATION, A protocol and token for self-sovereign identity and decentralized trust, 2018.

[Online]. Available: <https://sovrin.org/wp-content/uploads/Sovrin-Protocol-and-Token-White-Paper.pdf>

[27] Everest [Online]. Available: <https://everest.org>.

[28] SelfKey. [Online] Available: <https://selfkey.org>.

[29] ShoCard [Online]. Available: <https://shocard.com>.

[30] M. TAKEMIYA, B. VANIEIEV, Sora identity: Secure, digital identity on the blockchain. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), IEEE, 2018.