CHAPTER-1

INTRODUCTION

In recent times, it is becoming increasingly vital that communication not only be secure but also anonymous. The presence of mass surveillance programs, as well as cyber attacks, focused on compromising messaging applications highlight the need for maintaining the anonymity of communicating parties. In this article, we developed a decentralized messenger application utilizing the Ethereal Whisper protocol. Using our application, two users can engage in secure and anonymous communication which is encrypted end-to-end and resistant to network traffic analysis.

1.1 BACKGROUND AND DEFINITIONS

This section provides all the necessary background required for this work. We will briefly discuss secure messaging as well as blockchain technology. This will be followed by an overview of the Ethereal platform with emphasis made on the Whisper protocol.

A. SECURE MESSAGING

A significant amount of current electronic communication is still placed over several legacies protocols such as SMS/GSM, SMTP, and centralized messengers were not designed with end-to-end security as a requirement. These methods routinely broadcast recipient and sender information and therefore provide limited anonymity capabilities. Besides, they are more prone to suppression due to the storage and transmission requirements by intermediate servers. Communication systems with a peer-to-peer (P2P) architecture attempt to exchange messages directly between the participants in the network rather than rely on centralized servers for the storage and forwarding of messages. These systems commonly utilize Distributed Hash Tables (DHTs) for mapping usernames to IP addresses without the need for a centralized authority [1, pp. 20]. However, these P2P solutions such as Kademlia only partially anonymize the recipient and sender. Besides, they do not provide any capability to hide which participants are engaged in a conversation, and do not prevent protocol messages from being associated with a particular conversation. Furthermore, global network adversaries can view the flow of traffic between the participants of a conversation.

B. BLOCKCHAIN

A blockchain is an append-only distributed database operating within a P2P network whereby each peer has a partial or full copy of the blockchain. Due to their distributed nature, blockchains are highly available and fault-tolerant even if a large scale attack is mounted on the network. Changes in the blockchain are conducted through transactions that are broadcasted and verified by all the nodes in the network. After verification, the change is appended to the blockchain. To circumvent 'double-spending' several transactions are grouped into a block and are then verified simultaneously. The block is then appended to the blockchain usually through a proof-of-work (PoW) mechanism which requires computationally intensive work to be conducted by a 'mining' node. Mining nodes are incentivized through rewards while other nodes interested in only posting transactions provide a fee. In this way, the integrity of the data is ensured.

Ownership of accounts is maintained through asymmetric cryptography whereby a public key is shared with the network and the private key is known only to the holder. Only the holder of the private key can digitally sign transactions on the blockchain, whereas the public key can be utilized as a personal address that other users can send assets digitally or interact with in some way. Furthermore, users can usually create public/private key pairs anonymously thus protecting their identities. Therefore, blockchains provide an elegant means for handling the transmission of critical data or digital assets in untrusted distributed environments.

C. ETHEREAL

Ethereal is an open-source blockchain platform with distributed computing that allows developers to run smart contracts. These are collections of code that exist on the Ethereal blockchain and can run without the possibility of censorship, fraud, third-party interference or downtime. Data integrity is ensured through the use of Merkle Patricia tree which guarantees a data structure that is cryptographically authenticated.

The Ethereal technology stack is given in figure:

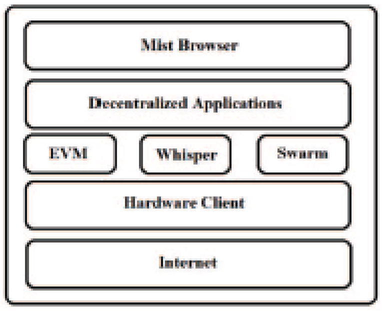


FIG. 1.1. ETHEREAL TECHNOLOGY STACK.

1) Mist Browser: an interface to access various dApps.

2) Decentralized Applications:

3) Ethereal Virtual Machine (EVM): The EVM is an abstraction layer that sits above the hardware clients and handles internal computation and state. All full nodes perform the same code execution on the EVM. The EVM is essentially a computer that can execute code and contain data with absolute availability and fault tolerance as long as the network is sufficiently large.

4) Whisper: It is Eternal's P2P communication protocol for decentralized applications. P2P Communication between nodes in the Whisper network utilizes the DΞVp2p Wire Protocol. A dApp instance can create an identity within a node that is connected to Whisper. This identity is needed to send or receive messages. Once a message is sent, it is, in theory, supposed to be routed through every Whisper node. This makes it necessary to implement a PoW algorithm to prevent denial of- service (DoS) attacks. Messages are only processed and further routed if their PoW is found to exceed a predefined threshold. Furthermore, Whisper allows dApp developers to configure the anonymity and security of their messages. All messages on Whisper are initially encrypted and sent through a basic wire-protocol called DΞVp2p which Supports various sub-protocols such as Whisper. The message is further encrypted by the DΞVp2p protocol. Currently, all Whisper messages are required to be either asymmetrically encrypted using the Elliptic Curve Integrated Encryption Scheme (ECIES) together with the SECP-256k public key or symmetrically encrypted using the Advanced Encryption Standard Galois/Counter Mode (AES-GCM) with a random 96-bit nonce. Multiple asymmetric and symmetric keys may be owned by a single node. If a message is successfully decrypted it is then Forwarded to its respective dApp. Using the web3-shh package, one can interact with the Whisper protocol.

The following methods within the web3-shh package are used.

- web3.shh.newKeyPair (): This method generates a new private and public key pair used for message encryption and decryption.
- web3.shh.newSymKey (): This method randomly generates a symmetric key and stores the key under an ID. This key is shared between communicating parties and used to encrypt and decrypt messages.
- web3.shh.getPublicKey (kId): This method returns the associated public key for a given key pair ID.
- web3.shh.getSymKey (id): This method returns the symmetric key for a given ID.
- web3.shh.newMessageFilter (options): This method creates a new message filter within the node to be used for polling messages that satisfy a set of criteria given.
- web3.shh.post (object ,[callback]): This method is called when Whisper message is to be posted to the network.

1.2.FUNDAMENTAL OF BLOCKCHAIN

The blockchain network can be described as a data structure used to create the ledgers that contain a lot of information related to the transaction. As it has for centuries, commerce relies on trust and verified identity with the cryptography protocol module embedded in the system to make sure the credibility of the data and the other security manners. The timestamp as shown in Figure 2 is used in digital documents to prevent the tamper-proof by the attacker. The block in the blockchain is like a seal, if the attacker tries to break the seal, everyone allows to know the action. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. The hash is produced by running contents of the block in question through a cryptographic hash function e.g. Bitcoin uses SHA-256. An ideal cryptographic hash function can easily produce a hash for any input, but it is difficult to derive the input.

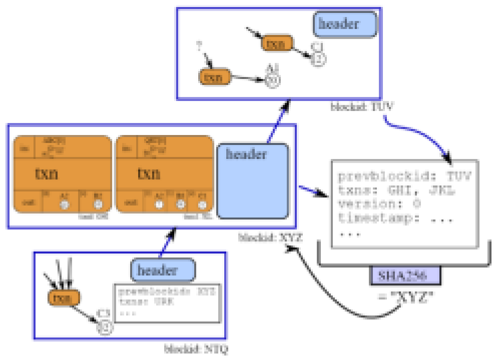


Fig 1.2. The information a block (transaction)

Table 1. Bitcoin block header format

Field	Description	Size
Version	Block Vers.num	4 bytes
Hash-Prev block	Hash of prev.header	32 bytes
Merkle root hash	Tx Merkle root hash	32 bytes
Time	Unix time stamp	4 bytes
nBits	Current difficulty	4 bytes
Nonce	Allows miners search	4 bytes

In various sectors, blockchain is being used to build some purposes such as financial registries, operational registries and one of the most popular ones is smart contracts: a. Financial registries: cryptocurrencies such as Litecoin, Bitcoin, and Dogecoin can be used as an alternative for the real currencies in the blockchain system. b. Operational registries: blockchain allow tracking and certification of specific products or assets, including renting contracts, land registers and notary deals or votes.

Smart contracts (automated actions on the blockchain): is account holding objects and contain several code functions to make decisions, store data and send the cryptocurrencies to the next owner. The smart contract can execute the code (self-executing). Proof-of-work in bitcoin, proof-of-stake and so on are various consensus protocols used to keep the blockchain secure. It depends on the consensus protocol; the blocks are created and added to the blockchain differently. In proof of work, blocks are created by a procedure called mining, which keeps the blockchain safe.

A probability of finding nNonce of proof H for given target T is:

$$P(H \leq T) = \frac{T}{2^{256}} \qquad (1)$$

The disadvantage of proof-work is related to efficiency that wastes too many computational resources to find the target value (hash puzzle). The hash in PoW begins with many zero bits hash (SHA-256) and involves kind of scanning for value when hashing a data.

RESILIENT OVERLAY NETWORK

An overlay network capable of delivering content, applications, and services to a global audience is a large distributed system [9]. Chord algorithm in overlay network provides a fast-distributed computation of hash function mapping keys to nodes responsible for them. It uses consistent hashing [10] which has several good properties. With high probability, when a Nth node joins (or leaves) the network, only an O(1/N) fraction of the keys are moved to a different location, this is the minimum necessary to maintain a balanced load [11]. Fig. 3 shows a possible three-layered software structure for a cooperative mirror system. The highest layer would provide an interface to users, including names.

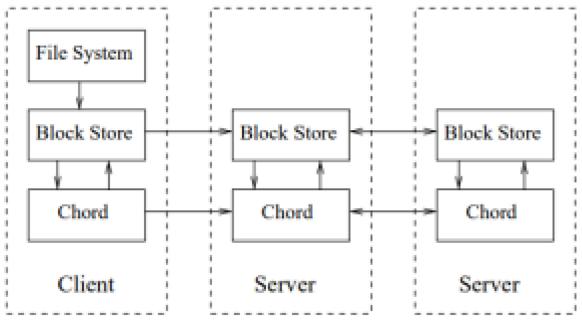


Fig 1.3. Chord-based distributed system

As shown in Fig. 3 a basic structure of the chord algorithm from client to server. From the client's side, there is a block such as a file system, block store and the chord itself. Whilst, from the server, there are block store and chord that connected to another server and the client. The main usage of the chord protocol is a query value [8] from a client to find a successor (k). This refers to an O(N) query time, and the N is referred to the number of rings applied. In our system model, the chord-based distributed system is used to know the location of the node among the neighbors in the decentralized trading system.

CHAPTER-2

LITERATURE SURVEY

1. **N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, "SoK: Secure Messaging", 2015 IEEE Symposium on Security and Privacy, pp. 22, 2015.**

Motivated by recent revelations of widespread state surveillance of personal communication, many products now claim to offer secure and private messaging. This includes both a large number of new projects and many widely adopted tools that have added security features. The intense pressure in the past two years to deliver solutions quickly has resulted in varying threat models, incomplete objectives, dubious security claims, and a lack of broad perspective on the existing cryptographic literature on secure communication. In this paper, we evaluate and systematize current secure messaging solutions and propose an evaluation framework for their security, usability, and ease-of-adoption properties. We consider solutions from academia, but also identify innovative and promising approaches used “in the wild” that are not considered by the academic literature. We identify three key challenges and map the design landscape for each: trust establishment, conversation security, and transport privacy. Trust establishment approaches offering strong security and privacy features perform poorly from a usability and adoption perspective, whereas some hybrid approaches that have not been well studied in the academic literature might provide better trade-offs in practice. In contrast, once trust is established, conversation security can be achieved without any user involvement in most two-party conversations, though conversations between larger groups still lack a good solution. Finally, transport privacy appears to be the most difficult problem to solve without paying significant performance penalties.

2. **P. Maymounkov and D. Mazieres, “Kademlia: A Peer-to-peer Information System Based on the XOR Metric,” in Peer-to-Peer Systems. Springer, 2002, pp. 53–65.**

We describe a peer-to-peer distributed hash table with provable consistency and performance in a fault-prone environment. Our system routes query and locates nodes using a novel XOR-based metric topology that simplifies the algorithm and facilitates our proof. The topology has the property that every message exchanged conveys or reinforces useful contact information. The system exploits this information to send parallel, asynchronous query messages that tolerate node failures without imposing timeout delays on users.

DDDAS is a conceptual framework that synergistically combines models and data to facilitate the analysis and prediction of physical phenomena. In an SSA application, DDDAS is a variation of adaptive state estimation that uses computational feedback rather than physical feedback to enhance the information content of measurements. The feedback loops in DDDAS include a data assimilation loop and a sensor reconfiguration loop. The data assimilation loop calculates the physical system simulation by using sensor data error to ensure that the trajectory of the simulation more closely follows the trajectory of the physical system. As a fundamental aspect of DDDAS, the sensor reconfiguration loop seeks to manage the physical sensors to enhance the information content of the collected data. The simulation-based on computational feedback process guides the sensor reconfiguration and the data collection, and in turn, improves the accuracy of the physical system environmental assessment (e.g., space weather and RSO tracking). For sensor management, DDDAS develops runtime software methods for real-time control such as access control.

3. **Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," 2017 IEEE International Congress on Big Data (Big Data Congress), pp. 558–560, 2017.**

The Blockchain, the foundation of Bitcoin, has received extensive attention recently. The Blockchain serves as an immutable ledger that allows transactions to take place in a decentralized manner. Block chain-based applications are springing up, covering numerous fields including financial services, reputation system and Internet of Things (IoT), and so on. However, there are still many challenges of blockchain technology such as scalability and security problems waiting to be overcome. This paper presents a comprehensive overview of blockchain technology. We provide an overview of blockchain architecture firstly and compare some typical consensus algorithms used in different blockchains. Furthermore, technical challenges and recent advances are briefly listed. We also layout possible future trends for blockchain.

The blockchain protocol has been recognized as the potential candidate to revolutionize the fundamentals of IT technology because of its many attractive features and characteristics, such as supporting decentralization and anonymity maintenance, as well as a fundamental protocol of Bitcoin, the first digital currency. In this paper, a Blockchain-Enabled, Decentralized, Capability-based Access Control (Blend CAC) scheme is proposed to enhance the security of space applications. Blend CAC provides a decentralized, scalable, fine-grained and lightweight authentication and access control mechanism to protect devices, services, and information in space networks. To achieve secure identity authentication, a decentralized authentication mechanism is implemented on the blockchain and aims at creating virtual trust zones to allow all distributed entities to identify each other and communicate securely in the trustless network environment. An identity-based capability token management strategy is presented and the federated authorization delegation mechanism is illustrated. A proof-of-concept prototype has been developed and evaluated on a private Ethereal blockchain network, and the experimental results demonstrate the feasibility and effectiveness of the proposed Blend CAC scheme.

4. **V. Buterin, "A next-generation smart contract and decentralized application platform," white paper, 2014.**

Blockchain technologies paired with smart contracts exhibit the potential to transform the global insurance industry. The recent evolution of smart contracts and their fast adoption allow to rethink processes and to challenge traditional structures. Therefore, a special focus is on the analysis of the underlying technology and recent improvements. Further, we provide an overview of how the insurance sector may be affected by blockchain technology. We emphasize current challenges and limitations by analyzing two promising use cases in this area. We find that realizing the full potential of blockchain technology requires overcoming several challenges including scalability, the incorporation of external information, flexibility, and giving access to the schemes.

Capability-based Access Control (Cap AC) utilizes the concept of capability that contains rights granted to the entity holding it. The capability is defined as tokens, tickets, or keys that give the possessor permission to access an entity or object in a computer system. The Cap AC has been implemented in many large-scale projects, like IoT@Work. However, the direct application of the original concept of the Cap AC model in a distributed network environment has raised several issues, like capability propagation and revocation. To tackle these challenges, a Secure Identity-Based Capability (SICAP) System was proposed to provide a prospective capability-based AC mechanism in distributed networks. By using an exception list, the SICAP enables monitoring, mediating, and recording capability propagations to enforce security policies as well as achieving rapid revocation capability. By introducing a delegation mechanism for the capability generation and propagation process, a Capability-based Context-Aware Access Control (CCAAC) model was proposed to enable contextual awareness in federated devices.

5. **Sandi Rahmadika , Diena Rauda Ramdania., "Security Analysis on the Decentralized Energy Trading System Using Blockchain Technology" 2016**

Blockchain turns both currencies and commodities into a digital form without relying on middleman which allows one person to trade with another include trading renewable energy. Blockchain technology as a secure and low-cost platform to track the billions of eventual transactions in a distributed energy economy has attracted the attention of experts in various fields of science. The current form of centralized energy trading system is still suffering from security concerns, quality of service, and to name a few. A decentralized energy system using blockchain technology allows the parties to create a trading energy transaction via micro-grid. The blockchain technology offers the promise of an immutable, single source of truth from multiple sources without third-party involvement. In this paper, we describe, explore and analyze the prominent implementation of blockchain technology in the energy sector. Furthermore, we analyze the security issues and highlight the performance of several attacks that might be occurred in the proposed system.

The concept of blockchain technology is used for trading the renewable energy system in an environment among the neighbors in the peer-to-peer network. We discussed a model for trading energy in a small environment by using blockchain technology and then we analyzed security issues that might be occurred. The performance of the attacker is also presented. Blockchain technology with cryptographic embedded to support the security issues can become a possible solution for the future to create a secure trading renewable energy system in the environment among the neighbors. The energy and commodity transaction life cycle, even for simple transactions, involves a multitude of processes within each company and across market participants. Blockchain turns both currencies and commodities into a digital form without relying on middleman which allows one person to trade with another. For the future, the strategy is needed to prevent the various attacks, especially in the overlay network.

CHAPTER-3

PROBLEM IDENTIFICATION & OBJECTIVES

3.1. EXISTING SYSTEM

Suggestions have been made to use the blockchain as a platform to store and transfer messages. The data to be stored on the blockchain is encrypted to prevent unauthorized access in these systems. In [11], a decentralized personal management system in which encrypted user data was suggested. Sending encrypted messages on payment platforms has also been suggested [12]. However, due to the open nature of blockchain, there are difficulties in ensuring the anonymity of two users trying to conduct a transaction or communicate [13]. Others have suggested using existing P2P communication protocols to route messages in a more anonymous fashion [14] [15]. Some of these protocols are unable to prevent the network from being attacked through flooding and do not provide guarantee anonymity for the sender [1, pp. 21].

The Whisper protocol is designed to provide complete anonymity and is resistant to certain attacks and network analysis. In [16], a messenger and coupon exchanging application were built using the Whisper protocol. Their implementation necessitated the exchange of a topic between the sender and the recipient for each session. Furthermore, this topic needed to be shared over some other secure channel other than the application. Therefore, this implementation was not designed for functioning in a trustless environment. Also, the implementation relied on some features which do not exist in the newer versions of the protocol such as the ability to send unencrypted messages with a topic. In their implementation, the unencrypted message could reveal information about the recipient and sender. For this reason, newer versions of the Whisper protocol require all messages to be encrypted to ensure anonymity by default. As far as we are aware, we are the first to implement a decentralized messaging application using the Whisper protocol that preserves the complete anonymity of the participants. Due to the Whisper protocol being an ongoing rapidly evolving project, certain portions of the official documentation and user guides were inconsistent, incorrect or outdated. This article brings together the most up-to-date information at the time of writing concerning the Whisper protocol through a direct analysis of the open-source code.

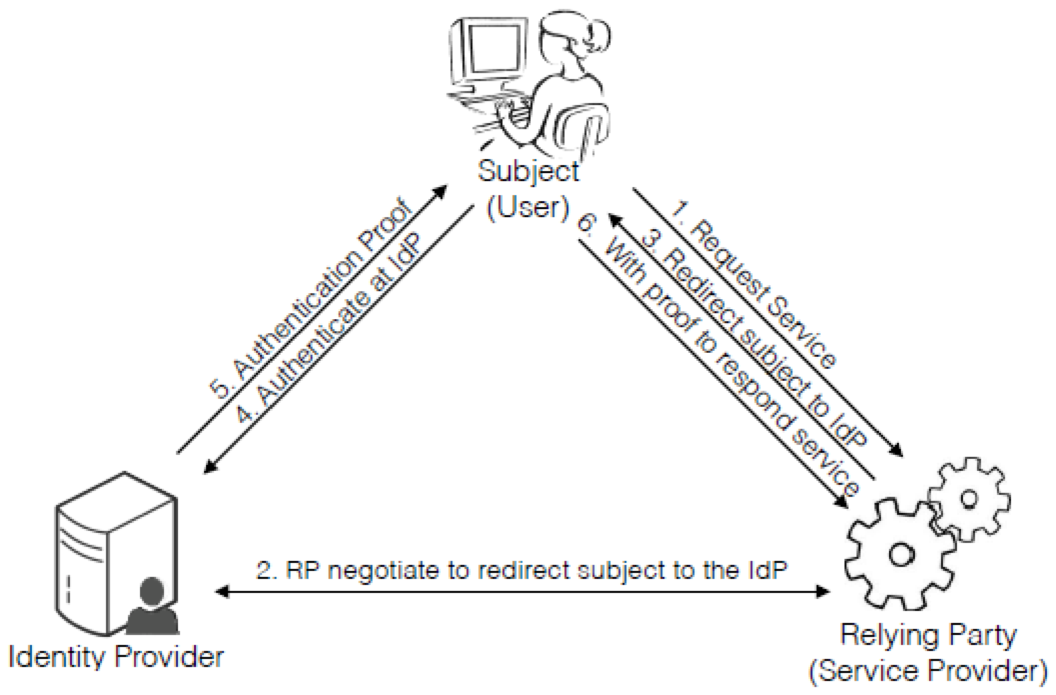


FIG 3.1. STAKEHOLDERS FROM THE TRADITIONAL IDMS MODEL.

Although the proposed blend CAC mechanism has demonstrated these attractive features, using Blockchain to enforce AC policy in space systems, it also incurs new challenges in performance and security. The transaction rate is associated with the confirmation time of the blockchain data, which depends on the block size and the time interval between the generations of new blocks. Thus, the latency for transaction validation may not be able to meet the requirement in real-time SSA scenarios. Besides, as the amount of transactions increases, the blockchain becomes large. The continuously growing data introduces more overhead on the storage and computing resources of each client, especially for resource-constrained devices. Furthermore, the blockchain is susceptible to majority attack (also known as 51% attacks), in which once an attacker takes over 51% of the computing power of the network by colluding selfish miners, they can control the blockchain and reverse the transactions. Finally, since the blockchain data is open to all nodes joined the blockchain network, such a property of transparency inevitably brings privacy leakage concerns. More research efforts are necessary to improve the trade-off when applying the Blend CAC in practical scenarios.

3.2. PROPOSED SYSTEM

SYSTEM DESIGN:

This section is divided into four parts. Firstly, we will describe the problem. In the second part, we will discuss the design choices we made. Thirdly, we will explore the software architecture of the implemented solution. Lastly, we describe the operation of the application.

A. PROBLEM STATEMENT

Listed below are characteristics we deemed vital for an anonymous secure messaging application expected to operate in untrusted environments:

- **END-TO-END ENCRYPTION**: Only the users should be able to decipher the data being stored or communicated.

- **ANONYMOUS SENDER**: The source of a particular message cannot be attributed to a specific entity.

- **ANONYMOUS RECIPIENT:** The destination of a particular message cannot be attributed to a specific entity.

- **ANONYMOUS PARTICIPANTS:** The set of participants engaged in a conversation cannot be determined.

- **UNLINKABILITY:** Only the participants engaged in a conversation can associate two or more protocol messages as belonging to the same conversation.

- **RESISTANT TO ATTACKS BY A GLOBAL ADVERSARY:** The anonymity of the protocol should not be threatened by global adversaries.

- **RESISTANT TO FLOOD AND SPAM ATTACKS:** Bulk messaging and DoS attacks should not be able to significantly impact the availability of the system.

B. DESIGN CONSIDERATIONS

First, it is necessary to explore whether there is a need for the major functions of the application “account management” and “messaging” function to be moved to a smart contract. The “account management” function contains methods to create an account as well as manage friends. Due to the open nature of smart contracts on Ethereal, it is possible for the contact information to be viewed by anyone if stored ‘on-chain’ or on the blockchain without encryption. Even with encryption, there are performance and cost concerns on storing, updating and retrieving the data needed. Therefore, it was decided to implement the ‘account management’ functionality locally. Possible future extensions could include backing-up encrypted user data on a P2P distributed file system. Once the application is connected to a local Geth client who serves as an Ethereal node, it can communicate with other nodes that are its peers on the Whisper Network using the ‘messaging function’. The ‘messaging function’ utilizes the Ethereal JavaScript API web3.js library and is capable of transmitting signed encrypted messages to peers on the Whisper network. Since the Geth instance handles Whisper identities it would not be in the user’s interest for the Geth instance to be run by a third-party. Another issue was handling messaging between users that would best preserve their anonymity and allow for plausible deniability. Since messages can encrypt both asymmetrically and symmetrically we first explored using signed encrypted messages for every message.

C. SOFTWARE ARCHITECTURE

The client uses Geth, an Ethereal client, to run a node and to serve as an interface to interact with the Whisper network. The front end consists of a web application built on Node.js and was chosen due to the abundant support for web3.js, the Ethereal JavaScript API. This makes it possible to make calls through the local Ethereal client using the JavaScript API on either the backend or frontend to interact with the Ethereal and Whisper network. The application UI is rendered using React.js and the frontend state management and JavaScript API requests are handled by Redux.js. The user data is stored locally.
Details of all of the parts are covered below:

1) GETH: the command-line interface for running a Go implementation of a full ethereal node.

2) NODE.JS: a JavaScript runtime environment used for building and executing event-driven, scalable applications.

3) REACT.JS: a JavaScript library used for building user interfaces.

4) **REDUX.JS**: a JavaScript library used for managing application state.

5) **APPLICATION**

a) ACCOUNT MANAGEMENT: included various functions such as creating accounts and adding/removing friends.

b) MESSAGING: allows the user to send encrypted messages to peers in the Whisper network.

6) WEB3.JS: is an Ethereal JavaScript API used to interact with the Ethereal and Whisper Network through a local Ethereal or Whisper node.

7) JSON-RPC: is a light-weight, stateless remote procedure calls (RPC) protocol.

8) WHISPER: it is Eternal's P2P communication protocol for decentralized applications.

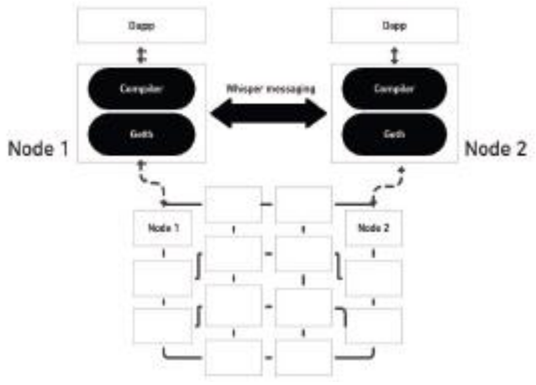


Fig. 3.2 Software network architecture.

D. OPERATION

Both the sender and recipient should have a generated asymmetric key pair. The sender initially transmits a message which is asymmetrically encrypted with the recipient's public key. The message contains a randomly generated symmetric key and a partial Topic for each participant in the chat. Topics are probabilistic filters known as bloom filters used to partially classify the message without compromising security. Nodes can filter for messages that are probably meant for them using the partial topic. After the recipient decrypts the message using their private key. The recipient verifies the identity of the sender and then retrieves the symmetric keys in the message. The receiver subscribes for messages with the first partial Topic and the sender subscribes for messages with the second partial Topic. The sender encrypts outgoing messages using the first symmetric key while the receiver encrypts outgoing messages with the latter symmetric key. After successfully sharing the symmetric keys and partial Topics, the sender and receiver can send each other secure messages with plausible deniability as all the participants have access to the encryption keys. Besides, the message number is included in each outgoing message to properly

order messages and check whether any messages are missing. Upon completion of the session, the symmetric keys are deleted and the corresponding subscriptions are discontinued.
The process of establishing a session and messaging between two parties is displayed in the following figure:

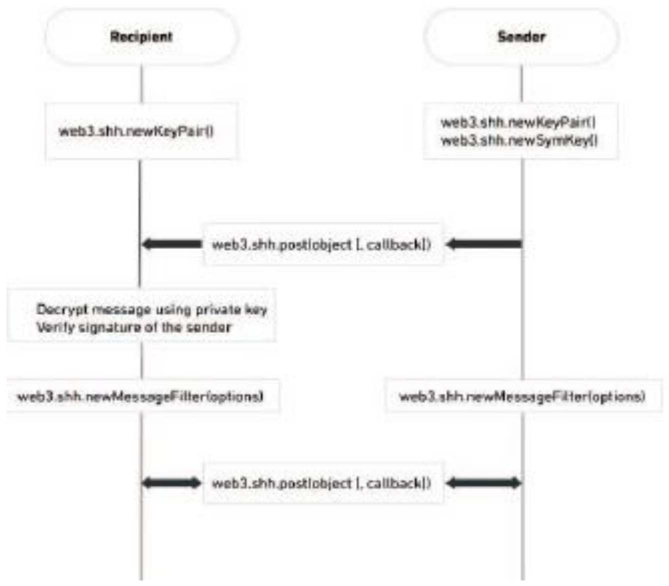

Recipient
Sender
web3.shh.newKeyPair()
web3.shh.newKeyPair()
web3.shh.newSymKey()
web3.shh.post(object [, callback])
Decrypt message using private key
Verify signature of the sender
web3.shh.newMessageFilter(options)
web3.shh.newMessageFilter(options)
web3.shh.post(object [, callback])

Fig. 3.3 Message transmission process.

CHAPTER-4

SYSTEM METHODOLOGY

The blockchain system isn't easy to implement as it has many implementations across the technical world around us and is modified according to the preferences of the project. So, in this project we use the blockchain for secure messaging with the help of the encryption with SHA-512 and also use the proof-of-work algorithm for the validity checking of the mined blockchain by the user.

The user can do a certain things which are shown in the use case diagram below

4.1. USE CASE DIAGRAM

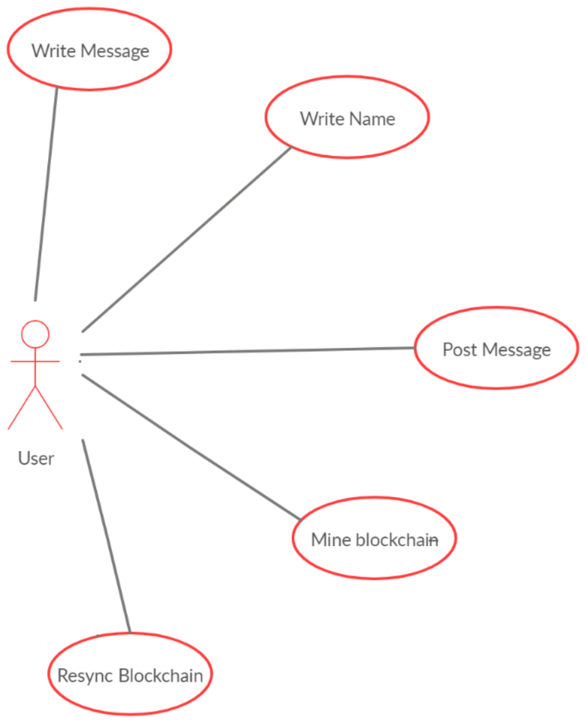


Fig 4.1 USE CASE DIAGRAM FOR USER

The use case diagram shows all the actions of the user which are to be performed by the user while operating the application.

4.2. SEQUENCE DIAGRAM

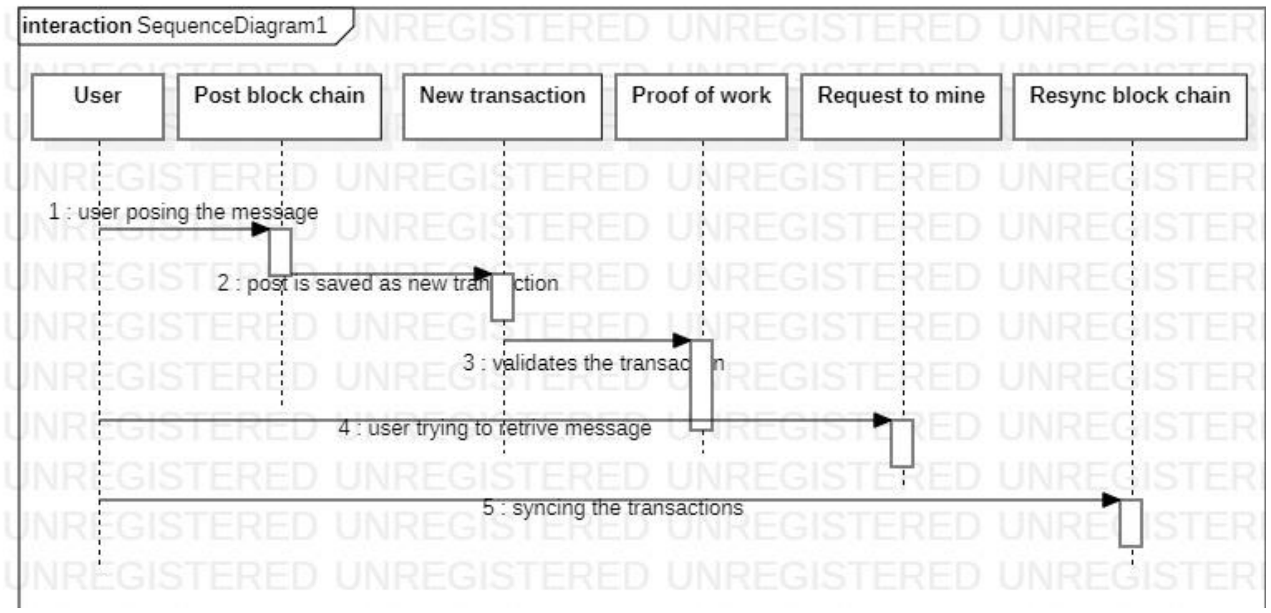

interaction SequenceDiagram1
User
Post block chain
New transaction
Proof of work
Request to mine
Resync block chain
1 : user posing the message
2 : post is saved as new transaction
3 : validates the transaction
4 : user trying to retrive message
5 : syncing the transactions
UNREGISTERED

Fig 4.2 SEQUENCE DIAGRAM OF THE APPLICATION

This diagram shows the sequence in which the application is running so in this first the user enters the message which is to be mined and then he posts it and the algorithm at backend validates it and then when user requests to mine it is mined and on resyncing the blockchain the message is displayed. Even in the below diagram the client is the user and this happens at the backend with algorithm inside.

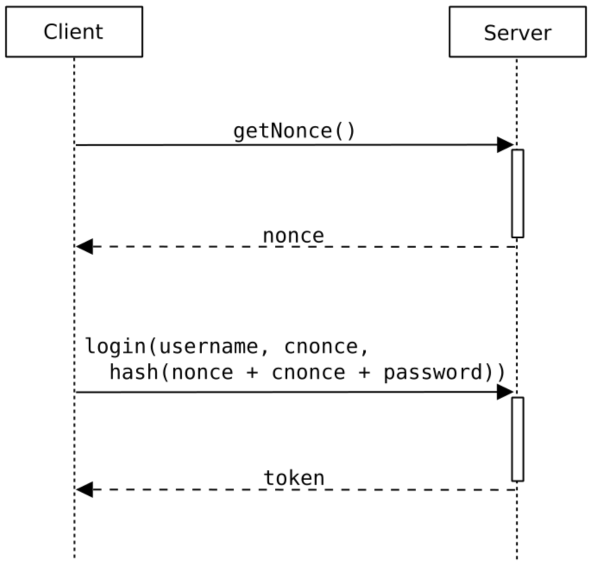


Fig 4.2.1 SEQUENCE OF PROJECT IN BACKEND

4.3 ACTIVITY DIAGRAM

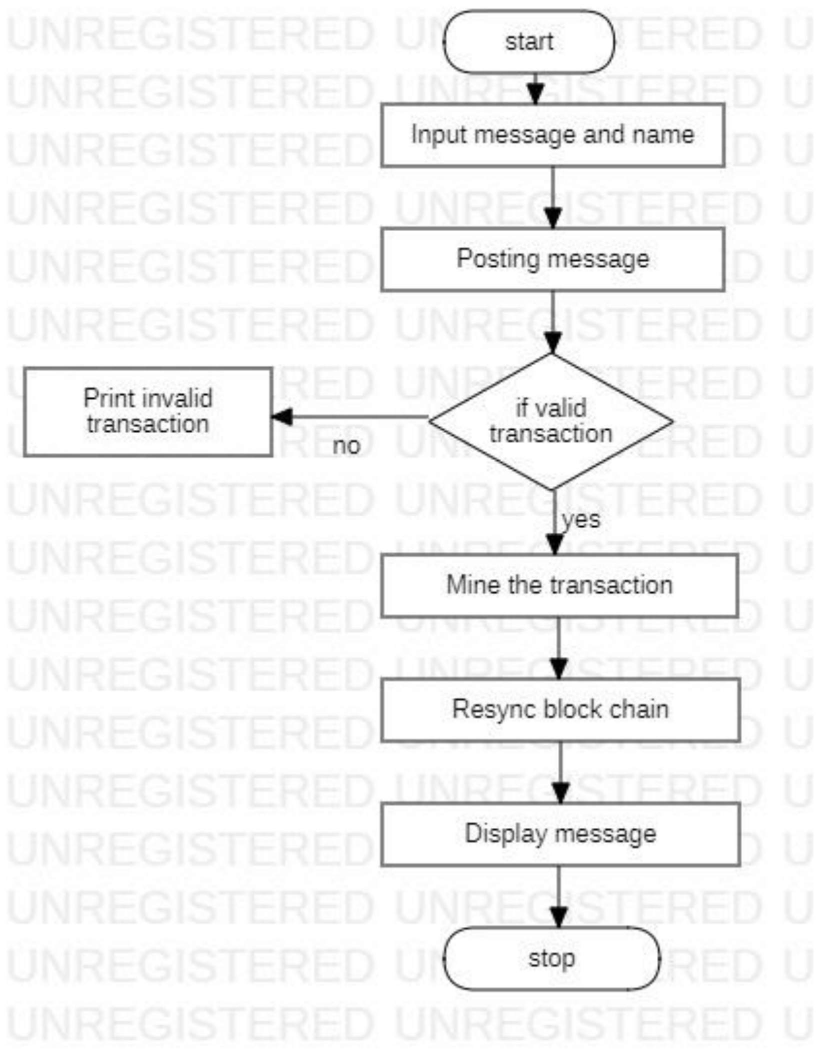


Fig 4.3 FLOW OF DATA THRO UGHOUT THE PROJECT

The data in the project flows right into the algorithm of the blockchain from the user's post which is to be mined and then after the validation of the transaction the blockchain has to be resynced in order to display the mined message from the user.

4.4 CLASS DIAGRAM

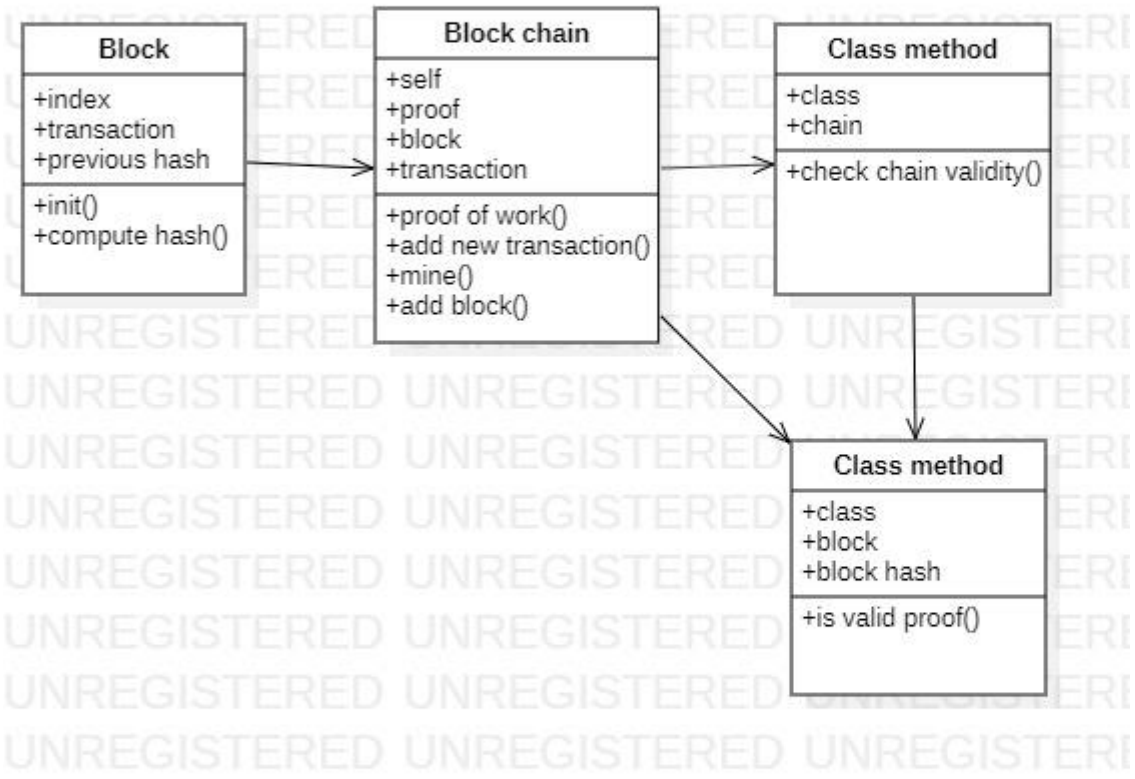


Fig 4.4 SOME CLASSES INVOLVED THROUGHOUT THE PROJECT

The above mentioned classes are only few classes which are prominent in the blockchain technology Which contains some primary methods and parameters without which the algorithms won't have the apt functionality though.

CHAPTER-5
OVERVIEW OF TECHNOLOGIES

5.1. DJANGO FRAMEWORK

Django is a Python-based free and open-source web framework, which follows the model-template-view (MTV) architectural pattern. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability and "pluggability" of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

WHY SHOULD YOU USE DJANGO?

Django is designed in such a way that encourages developers to develop websites fast, clean and with practical design. Django's practical approach to getting things done is where it stands out from the crowd.

If you're planning to build a highly customizable app, such as social media website, Django is one of the best frameworks to consider. Django strength lies in its interaction between users or its ability to share different types of media. One of the great advantage of Django is its ability to utilize large community-based support which gives you highly customizable third-party ready to use plugins in your applications

In Django, we have Long Term Support (LTS) versions of the software and a defined release process as shown in the below image

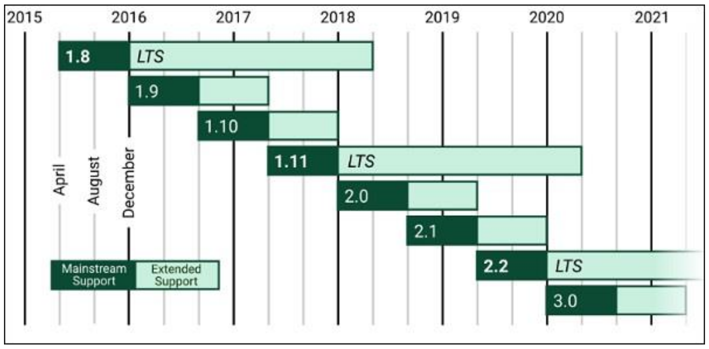


Fig 5.1 Long Term Support (LTS) versions of the software and a defined release process.

5.2.BLOCKCHAIN TECHNOLOGY

Blockchain technology, which was initially introduced by the Bitcoin crypto-currency, opens opportunities to multiple initiatives and research topics in the context of IoT security. Moreover, Blockchains, like distributed ledgers, keep permanent records of all transactions used to transfer bitcoin values between members, participating in the Bitcoin peer-to-peer network. They also define the structure of how transactions should be organized into blocks, mined, confirmed and stored. Mining is the mechanism that allows the Blockchain to be decentralized and secure. Nakamoto introduced the concept of Proof-of-Work (PoW) as a mining process to ensure consistency of transactions and solve the double-spending problem in decentralized networks. With the PoW Blockchain, however, there is no need for any kind of a trusted authority, such as a bank, to keep track of the money transfer, all members have their tamper-proof copy of the Blockchain ledger.

Each node in the Bitcoin peer-to-peer network maintains a copy of the Blockchain. Besides, the Blockchain is simultaneously updated through the peer-to-peer network so all members can validate any transaction instantly. Since 2014, Blockchain entered the 2.0 era led by Ethereal, which is a decentralized platform based on Blockchain technology. It aims at creating a general-purpose decentralized computer via the Turing-completeness smart contract concept, which allows writing and deploying all kinds of decentralized applications (Dapps) without any possibility of downtime, censorship or fraud. Buterin explained Ethereal as: “combining the cryptographic algorithms with the economic incentives to create a decentralized network with memory.” To sum up, Blockchain provides us a new perspective to reconstruct the Internet in distributed P2P networks without any unnecessary intermediaries.

5.2.1 BLOCKCHAIN ARCHITECTURE

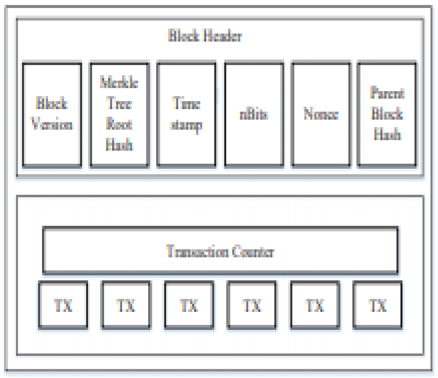


Fig.5.2: Block structure

A Blockchain is a sequence of blocks, which holds a complete list of transaction records like conventional public ledger. Figure 5.2 illustrates an example of a Blockchain. It is worth noting that uncle blocks (children of the block's ancestors) hashes would also be stored in the Ethereal Blockchain. The first block of a Blockchain is called genesis block which has no parent block.

5.3 CONSENSUS ALGORITHMS (PROOF-OF-WORK (PoW))

In the blockchain, how to reach consensus among the untrustworthy nodes is a transformation of the Byzantine Generals (BG) Problem, which was raised in. In BG problem, a group of generals who command a portion of Byzantine army circle the city. Some generals prefer to attack while other generals prefer to retreat. However, the attack would fail if only part of the generals attacks the city. Thus, they have to reach an agreement to attack or retreat. How to reach a consensus in a distributed environment is a challenge. It is also a challenge for blockchain as the blockchain network is distributed. In the blockchain, there is no central node that ensures ledgers on distributed nodes are all the same. Some protocols are needed to ensure ledgers in different nodes are consistent.

In PoW, each node of the network is calculating a hash value of the block header. The block header contains a nonce and miners would change the nonce frequently to get different hash values. The consensus requires that the calculated value must be equal to or smaller than a certain given value. When one node reaches the target value, it would broadcast the block to other nodes and all other nodes must mutually confirm the correctness of the hash value. If the block is validated, other miners would append this new block to their blockchains.

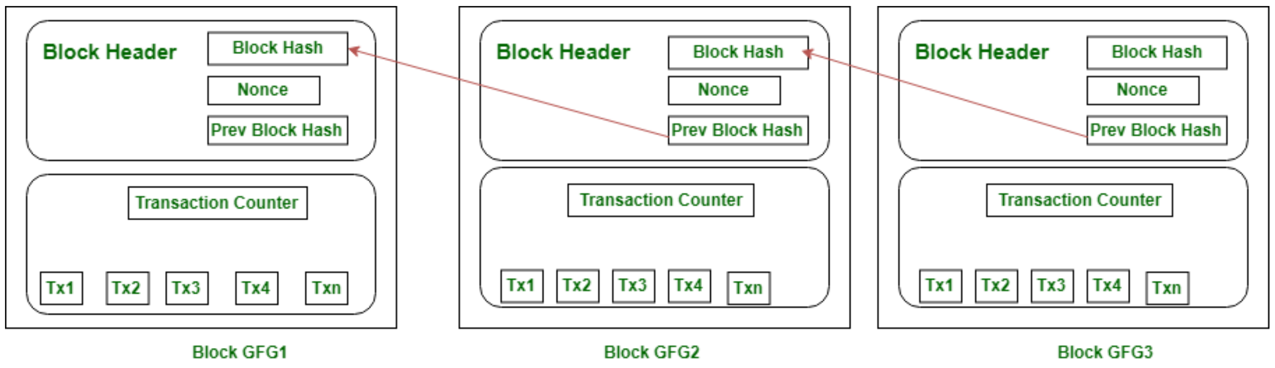


FIG. 5.3 PROOF-OF-WORK

ALGORITHM -PoW

```
def proof_of_work(self, block):
    block.nonce = 0
    computed_hash = block.compute_hash()
    while not computed_hash.startswith("0" * Blockchain.difficulty):
        block.nonce += 1
        computed_hash = block.compute_hash()
    return computed_hash
```

The algorithm works on the blocks which have been created with an specified difficulty while creating a block in an blockchain during a transaction in going on.

```
class Blockchain:
    # Difficulty of PoW algorithm.
    difficulty = 2
    # One or more blocks will be stored and chained together on the Blockchain, starting with the
genesis block.
    def __init__(self):
        self.unconfirmed_transactions = [] # Pieces of data that are not yet added to the
Blockchain.
        self.chain = [] # The immutable list that represents the actual Blockchain.
```

5.4 JSON (JAVASCRIPT OBJECT NOTATION)

JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange. The JSON format was originally specified by Douglas Crockford, and is described in RFC 4627. The official Internet media type for JSON is application/json. The JSON filename extension is .json.

- JSON stands for JavaScript Object Notation.
- The format was specified by Douglas Crockford.
- It was designed for human-readable data interchange.
- It has been extended from the JavaScript scripting language.
- The filename extension is **.json**.
- JSON Internet Media type is **application/json**.
- The Uniform Type Identifier is public.json.

USES OF JSON

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

CHARACTERISTICS OF JSON

- JSON is easy to read and write.
- It is a lightweight text-based interchange format and language independent.

CHAPTER-6

IMPLEMENTATION

The implementation of the project is based upon the usage of the technologies which were mentioned in one of the above chapters. Majorly we use the web technologies for the user interaction part of the project and also adding to that the backend process is implemented in Django framework which uses python language and also some various algorithms and packages for running the project at backend with a local server. Here is the outline of the project for the user interaction.

6.1 SAMPLE CODES OF PROJECT

THE USER INTERACTION CODE

```
        <div class="content">
                <center>
                        <br>
                        <form action="/submit" id="textform" method="post">
                                <textarea class="post-textarea" name="content" rows="4" cols="50"
placeholder="Write some content to add to the network ..."></textarea>
                                <br><br>
                                <input class="name-input" type="text" name="author"
placeholder="Name">
                                <input class="btn" type="submit" value="Post">
                        </form>
                        <br>
                </center>
                <div class="posts">
                        {% for post in posts %}
                        <div class="post_box">
                                <div class="post_box-header">
                                        <!--<div class="post_box-options"><button class="option-
btn">Reply</button></div>-->
                                        <div style="background: rgb(52, 152, 219) none repeat scroll
0% 0%; box-shadow: rgb(52, 152, 219) 0 0 0 0.2rem;" class="post_box-
avatar">{{post.author[0]}}</div>
```

```
                                        <div class="name-header">{{post.author}}</div>
                                        <div class="post_box-subtitle"><i>Posted at
{{readable_time(post.timestamp)}}</i></div>
                                </div>
                                <div>
                                        <div class="post_box-body">
                                                <p>{{post.content}}</p>
                                        </div>
                                </div>
                        </div>
                        {% endfor %}
                </div>
        </div>
```

REGISTERING NEW TRANSACTIONS

```
def new_transaction():
        tx_data = request.get_json()
        required_fields = ["author", "content"]
        for field in required_fields:
                if not tx_data.get(field):
                        return "Invalid transaction data", 404
        tx_data["timestamp"] = time.time()
        blockchain.add_new_transaction(tx_data)
        return "Success", 201
```

REGISTERING NEW USERS

```
def register_new_peers():
        nodes = request.get_json()
        if not nodes:
                return "Invalid data", 400
        for node in nodes:
                peers.add(node)
        return "Success", 201
```

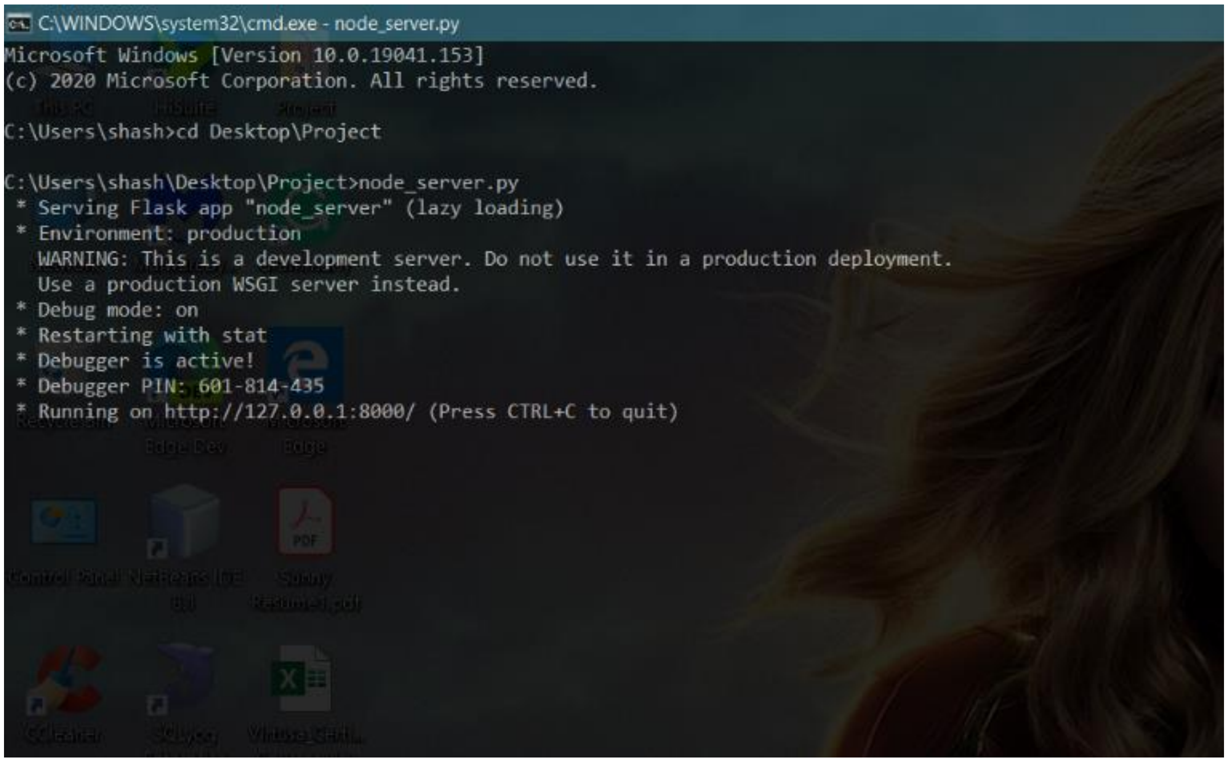
CODE FOR FETCHING THE POST BY THE USER

```
def fetch_posts():
    get_chain_address = "{0}/chain".format(CONNECTED_NODE_ADDRESS)
    response = requests.get(get_chain_address)
    if response.status_code == 200:
        content = []
        chain = json.loads(response.content.decode("utf-8"))
        for block in chain["chain"]:
            for tx in block["transactions"]:
                tx["index"] = block["index"]
                tx["hash"] = block["previous_hash"]
                content.append(tx)
        global posts
```

6.2 TESTING

While testing the project it is kept in mind that all the functions are well settled and functioning as expected to be so in that case we go on for a series if testing of the project. Well in this project we test it by having many number of users using this medium for communication.

1..RUNNING THE BACK-END SERVER OF THE PROJECT

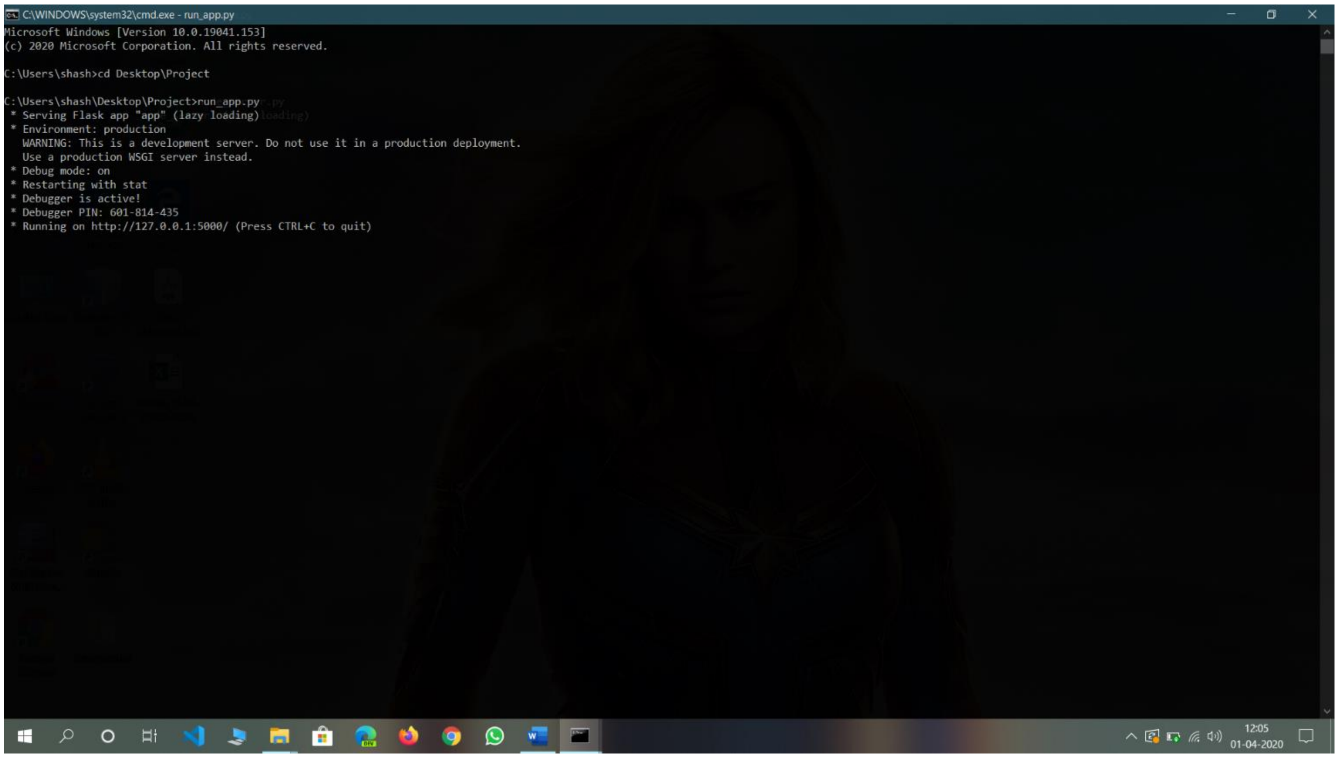


```
C:\WINDOWS\system32\cmd.exe - node_server.py
Microsoft Windows [Version 10.0.19041.153]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\shash>cd Desktop\Project

C:\Users\shash\Desktop\Project>node_server.py
 * Serving Flask app "node_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 601-814-435
 * Running on http://127.0.0.1:8000/ (Press CTRL+C to quit)
```

Fig 6.1 Running the local server on the particular port mentioned.

The backend server is being run on the windows default terminal by executing the direct command of running the node_server.py file .

2. RUNNING THE MAIN APPLICATION OF THE PROJECT



```
Microsoft Windows [Version 10.0.19041.153]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\shash>cd Desktop\Project

C:\Users\shash\Desktop\Project>run_app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 601-814-435
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Fig 6.2 Running the application on the particular port mentioned.

The main application is being run on the local server which was launched in the previous step in the specific port and then the url which is created is pasted in any browser to launch the real application and the real project is live to work on.

3. USER INTERACTION PAGE

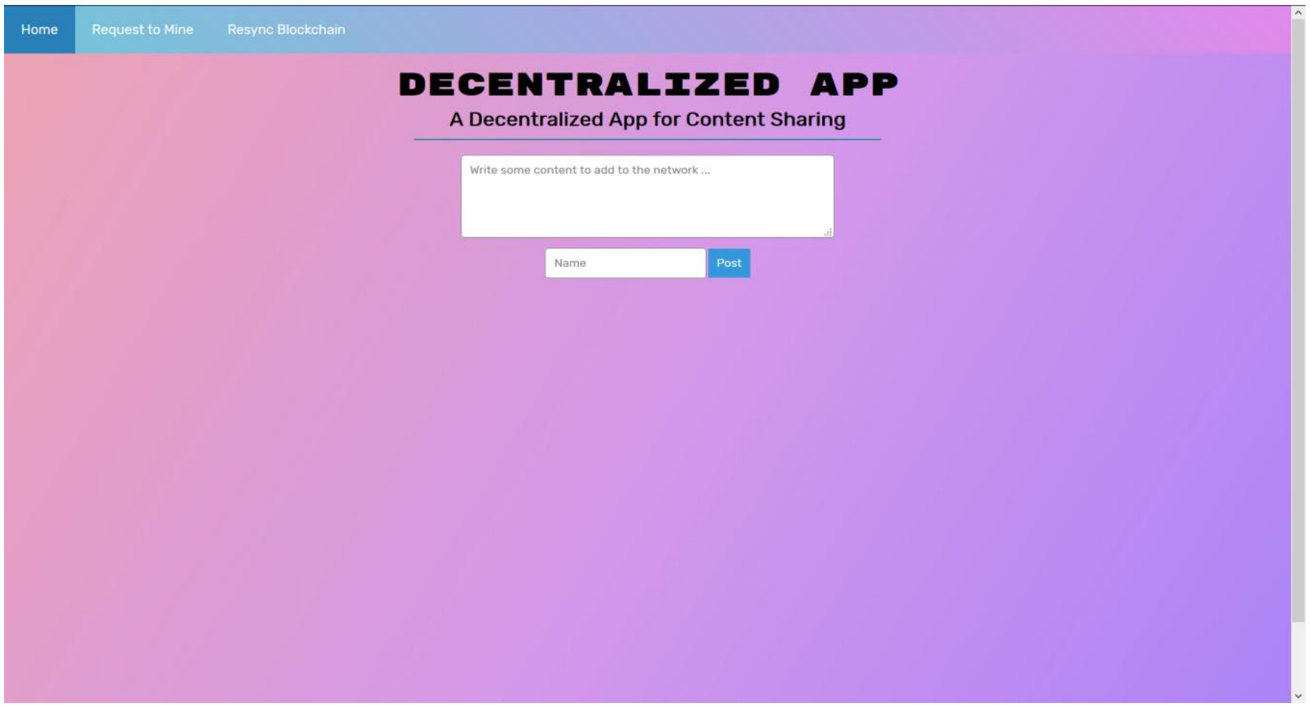

Home
Request to Mine
Resync Blockchain
DECENTRALIZED APP
A Decentralized App for Content Sharing
Write some content to add to the network ...
Name
Post

Fig 6.3 INITIAL PAGE AFTER LAUNCHING PROJECT

The initial page which would be seen by the user is seemed to be as shown in the above figure where the actions preferred by the user can be performed.

4. POSTING A MESSAGE WITH REGISTERING A NAME

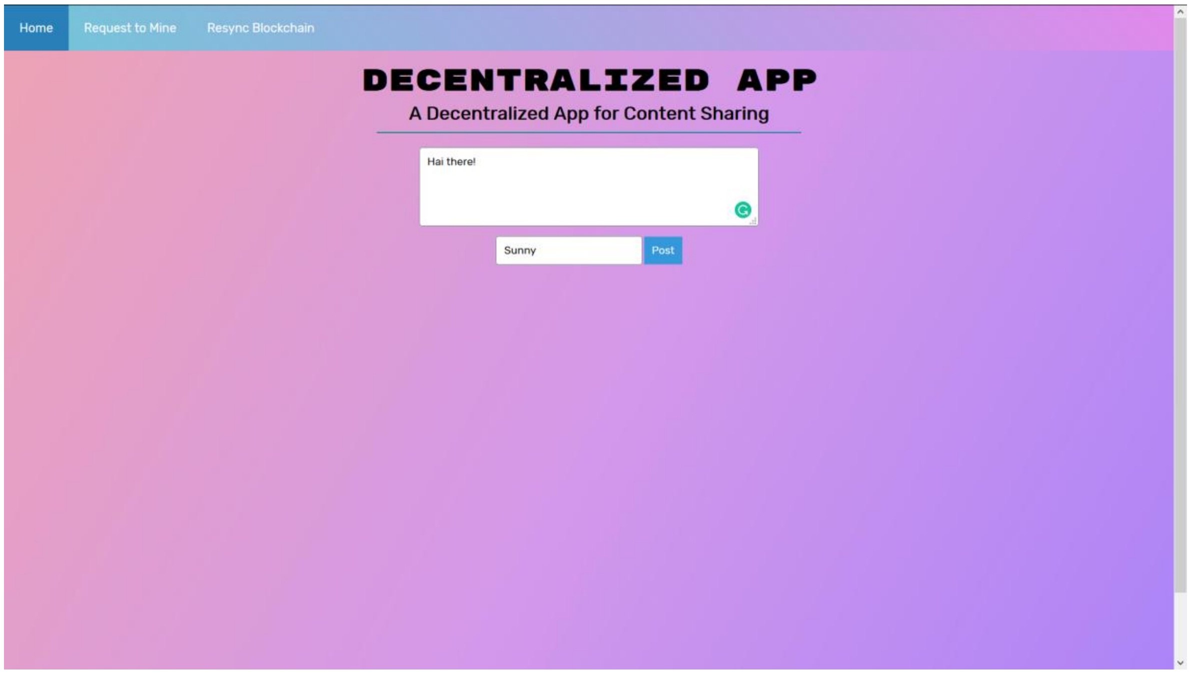
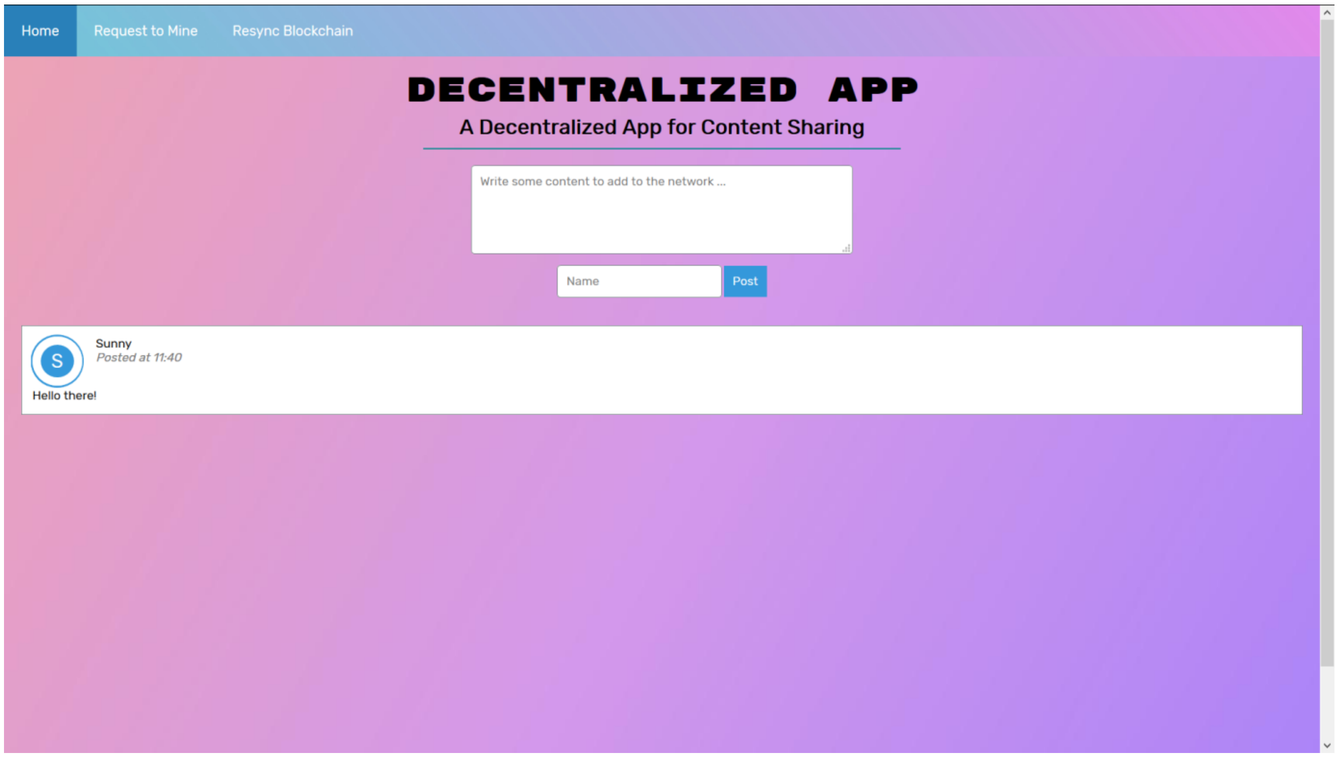


Fig 6.4. INDEX PAGE WHERE MESSAGE IS ENTERED

Firstly any message is written in the text area provided in the page and then after entering any name preferred by the user, the user can post it and the blockchain is mined in the back-end by the registered name.

5. OUTPUT AFTER RESYNCHRONIZING THE BLOCKCHAIN AFTER SUBMISSION

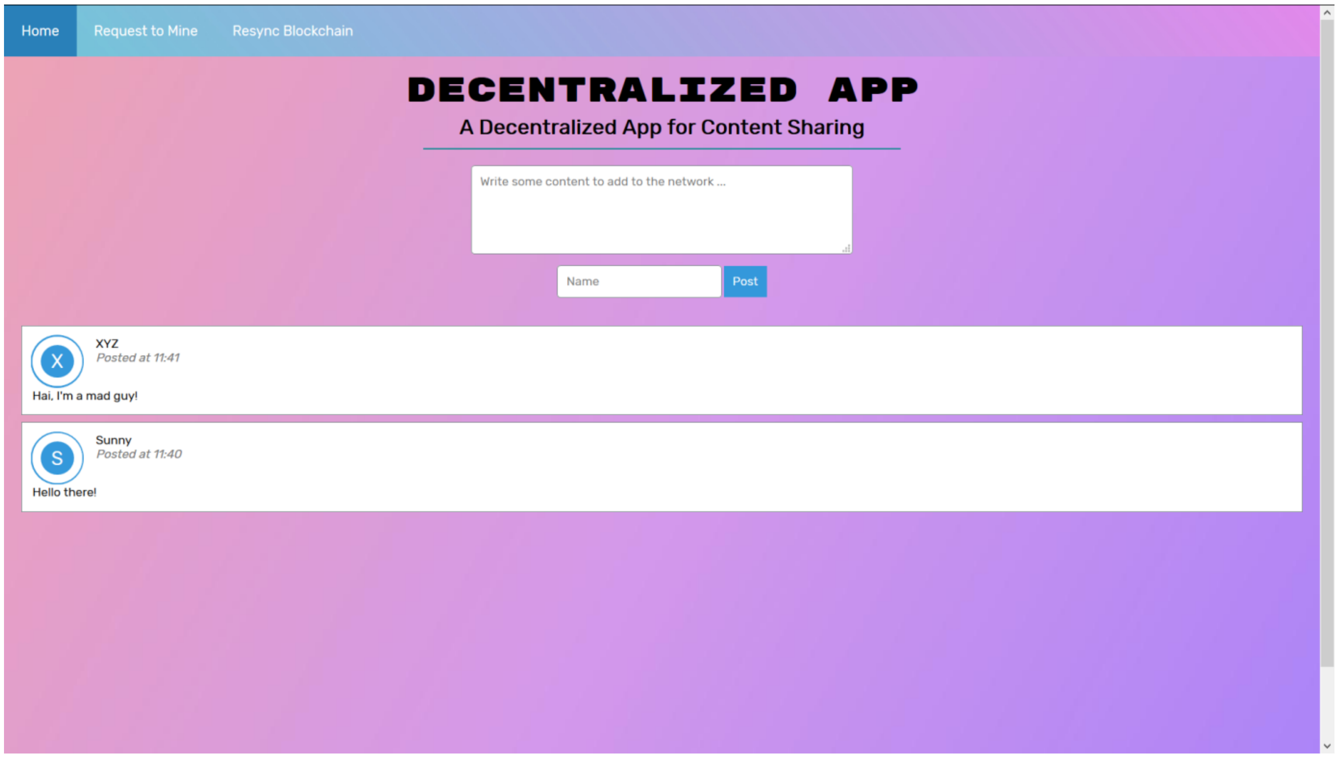

Home
Request to Mine
Resync Blockchain
DECENTRALIZED APP
A Decentralized App for Content Sharing
Write some content to add to the network ...
Name
Post

Sunny
Posted at 11:40
Hello there!

Fig 6.5 THE MESSAGE IS DISPLAYED ON THE SCREEN

After we request to mine the blockchain which is posted by the user the blockchain is mined and then synchronized when the button Resync Blockchain is clicked and then the message is displayed as above.

6.THE SAME STEPS WERE REPEATED TO GET MORE USERS


Home
Request to Mine
Resync Blockchain
DECENTRALIZED APP
A Decentralized App for Content Sharing
Write some content to add to the network ...
Name
Post

XYZ
Posted at 11:41
Hai, I'm a mad guy!

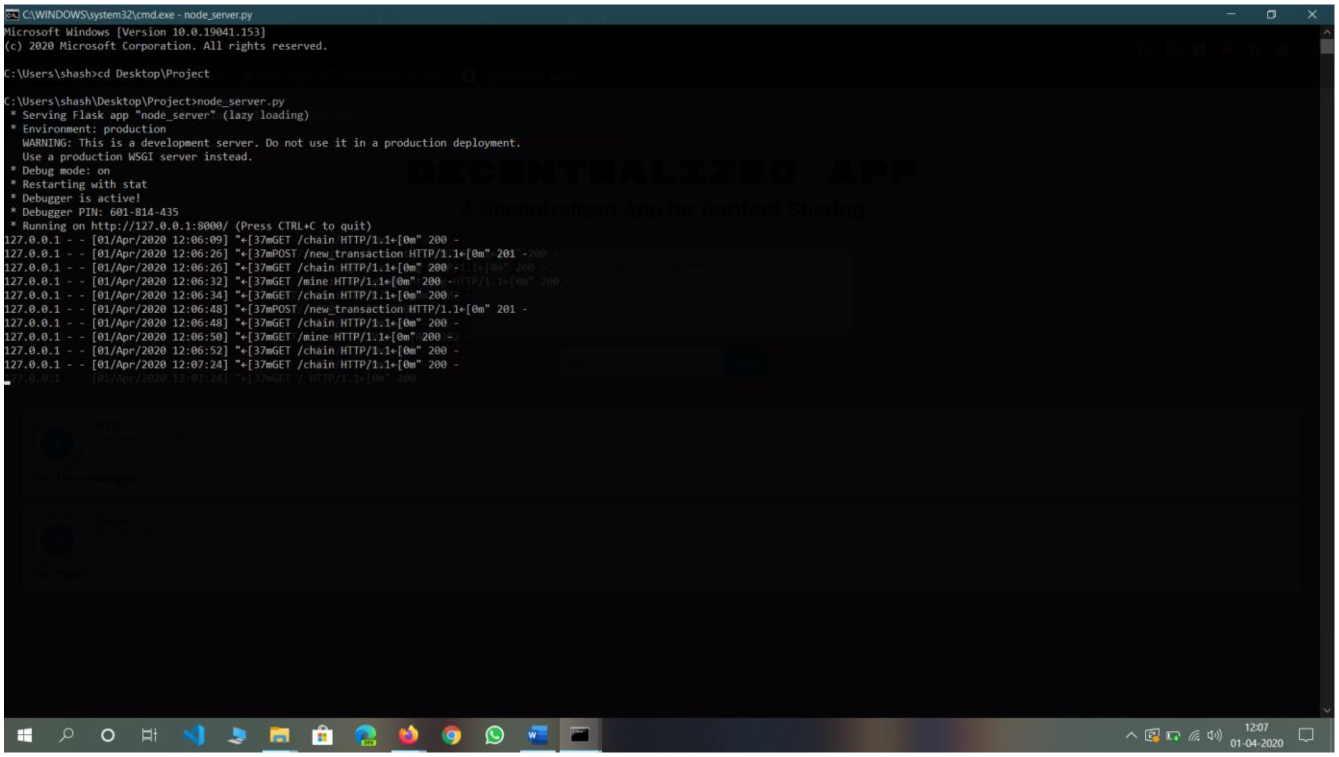
Sunny
Posted at 11:40
Hello there!

Fig 6.6 FINAL OUTPUT OF THE PROJECT AFTER RESYNCHRONIZING THE SECOND USER'S BLOCKCHAIN AS WELL.

After we have resynchronized the blockchain of the second user as well then the final output would be like shown and the number of messages is directly proportional to number of users posting the blockchain.

While the project is being run all the actions performed in the application are logged and can be seen in the terminal through which the application was launched initially. The local server and the application in action both the terminals logs the actions made by the user in it. The terminal windows of them are shown as below.

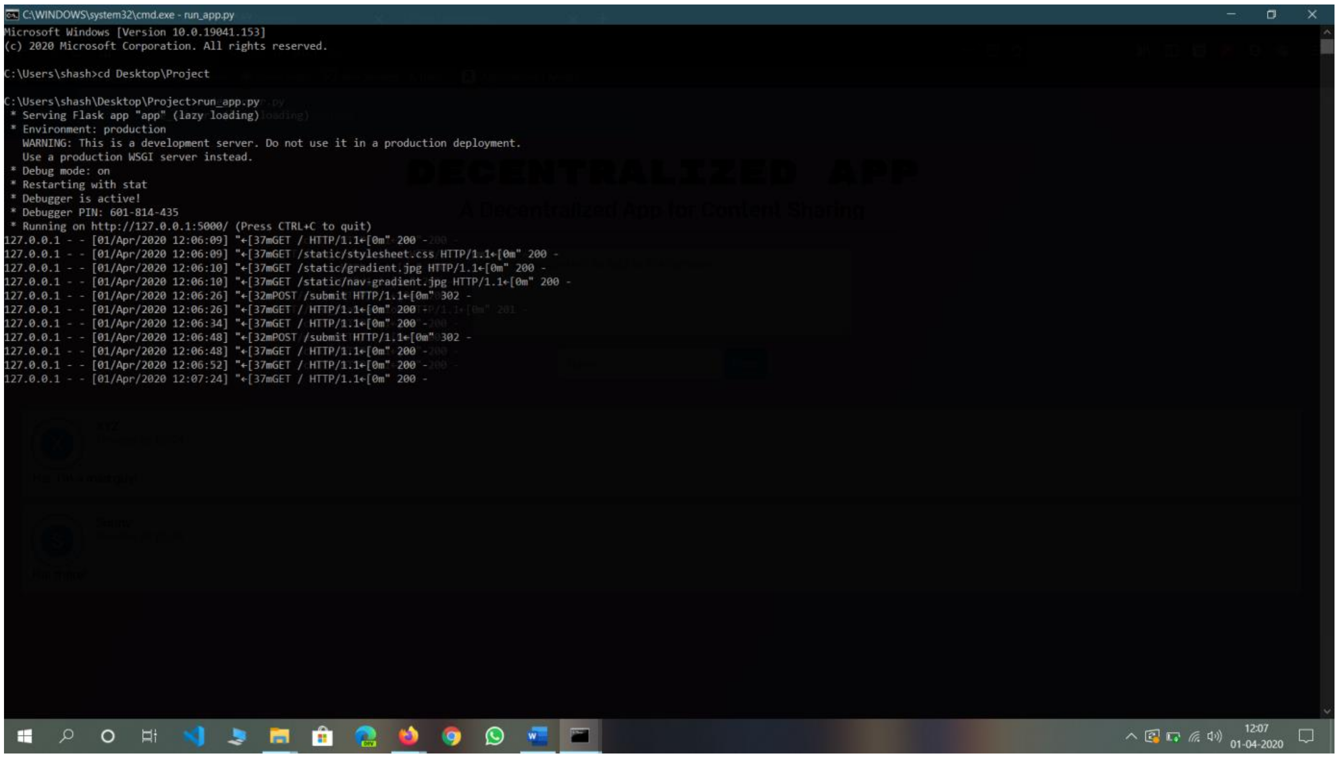
7.ACTIONS LOGGED BY THE LOCAL SERVER LAUNCHED



```
C:\WINDOWS\system32\cmd.exe - node_server.py
Microsoft Windows [Version 10.0.19041.153]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\shash>cd Desktop\Project

C:\Users\shash\Desktop\Project>node_server.py
 * Serving Flask app "node_server" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 601-814-435
 * Running on http://127.0.0.1:8000/ (Press CTRL+C to quit)
127.0.0.1 - - [01/Apr/2020 12:06:09] "←[37mGET /chain HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:26] "←[37mPOST /new_transaction HTTP/1.1←[0m" 201 -
127.0.0.1 - - [01/Apr/2020 12:06:26] "←[37mGET /chain HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:32] "←[37mGET /mine HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:34] "←[37mGET /chain HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:48] "←[37mPOST /new_transaction HTTP/1.1←[0m" 201 -
127.0.0.1 - - [01/Apr/2020 12:06:48] "←[37mGET /chain HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:50] "←[37mGET /mine HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:52] "←[37mGET /chain HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:07:24] "←[37mGET /chain HTTP/1.1←[0m" 200 -
```

Fig 6.7 TERMINAL OUTPUT FOR THE LOGS IN THE BACK-END SERVER.

8. ACTIONS LOGGED BY THE APPLICATION LAUNCHED



```
C:\WINDOWS\system32\cmd.exe - run_app.py
Microsoft Windows [Version 10.0.19041.153]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\shash>cd Desktop\Project

C:\Users\shash\Desktop\Project>run_app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 601-814-435
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [01/Apr/2020 12:06:09] "←[37mGET / HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:09] "←[37mGET /static/stylesheet.css HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:10] "←[37mGET /static/gradient.jpg HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:10] "←[37mGET /static/nav-gradient.jpg HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:26] "←[32mPOST /submit HTTP/1.1←[0m" 302 -
127.0.0.1 - - [01/Apr/2020 12:06:26] "←[37mGET / HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:34] "←[37mGET / HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:48] "←[32mPOST /submit HTTP/1.1←[0m" 302 -
127.0.0.1 - - [01/Apr/2020 12:06:48] "←[37mGET / HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:06:52] "←[37mGET / HTTP/1.1←[0m" 200 -
127.0.0.1 - - [01/Apr/2020 12:07:24] "←[37mGET / HTTP/1.1←[0m" 200 -
```

Fig 6.8 TERMINAL OUTPUT FOR THE LOGS IN THE APPLICATION ON BROWSER.

CHAPTER-7

RESULTS AND DISCUSSION

7.1 RESULT OF THE PROJECT

This section is divided into three parts. In the first part, we review the encryption algorithm used. Secondly, an analysis of the application’s resistance to DoS attacks is made. Finally, the resistance of the application to network traffic analysis is examined.

A. ENCRYPTION ALGORITHM

Sha-512 is a function of cryptographic algorithm Sha-2, which is an evolution of famous Sha-1. Sha-512 is very close to its "brother" Sha-256 except that it used 1024 bits "blocks", and accept as input a 2^128 bits maximum length string. Sha-512 also has others algorithmic modifications in comparison with Sha-256. This cryptographic function is part of the U.S Federal Information Processing Standard. This decryption database contains more than 2.000.000.000 hashes coming from all the wordlists. Sha-512 is very secure, but also takes a lot of database space. If you want to use it, you should still use a salt to improve security. A salt is a string sequence that you add to the user's password to add special characters to it, and makes it longer. This will help by making brute force more difficult, and avoid the password to be found on an online database.

After every block of 1024 bits goes through the message processing phase, i.e. the last iteration of the phase, we get the final 512 bit Hash value of our original message. So, the intermediate results are all used from each block for processing the next block. And when the final 1024 bit block has finished being processed, we have with us the final result of the SHA-512 algorithm for our original message.

B. RESISTANCE TO DENIAL OF SERVICE ATTACKS

Since every Whisper message is routed to every node that it can reach, the network might be susceptible to two types of attacks:

EXPIRY ATTACK: Messages are set to have a long Time-to-Live (TTL) on the envelope.

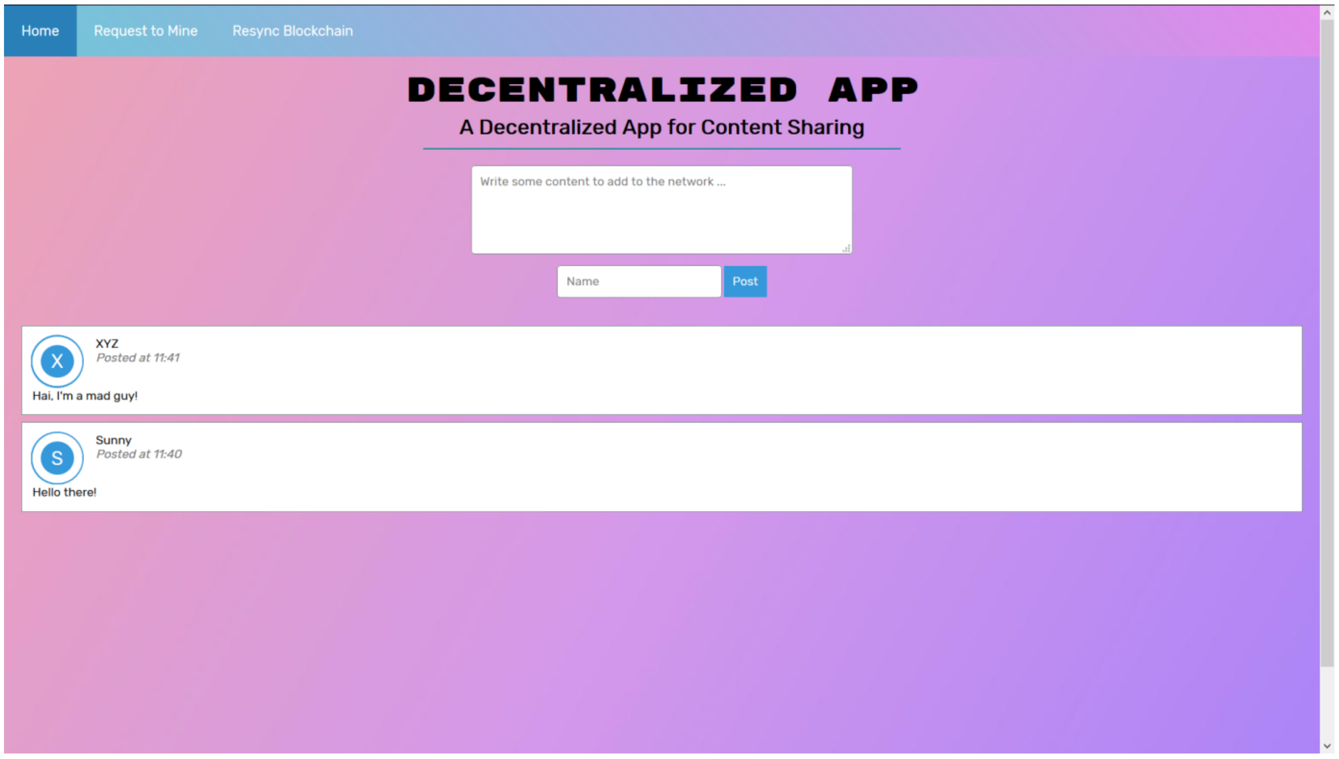
FLOOD ATTACK: The network is repeatedly sent messages. An expiry attack is averted by using a system that rates messages by taking into account the message size, TTL and POW.

Messages that have a smaller size, lower TTL and higher POW are considered to have a higher rating. This rating impacts how long the message is stored as well as its forwarding priority. Lower rated messages would be removed first in the event of a DoS attack, thus preventing an expiry attack.
A Flood attack can be averted by requiring the sender to conduct POW computation and posting the result to the EnvNonce field within the message envelope. If the PoW is below the amount required, then the message is not further routed.

C. RESISTANCE TO NETWORK TRAFFIC ANALYSIS

Since every message is routed to every node in the network, it is impossible to determine the identity of the recipient of a certain message. However, a global adversary might be able to ascertain the identity of the sender if they control a significant portion of the network. This type of analysis is curtailed by having the sender node generate an appropriate amount of encrypted junk messages. In this way, the adversary would not be able to determine the identity of the sender of a certain message.

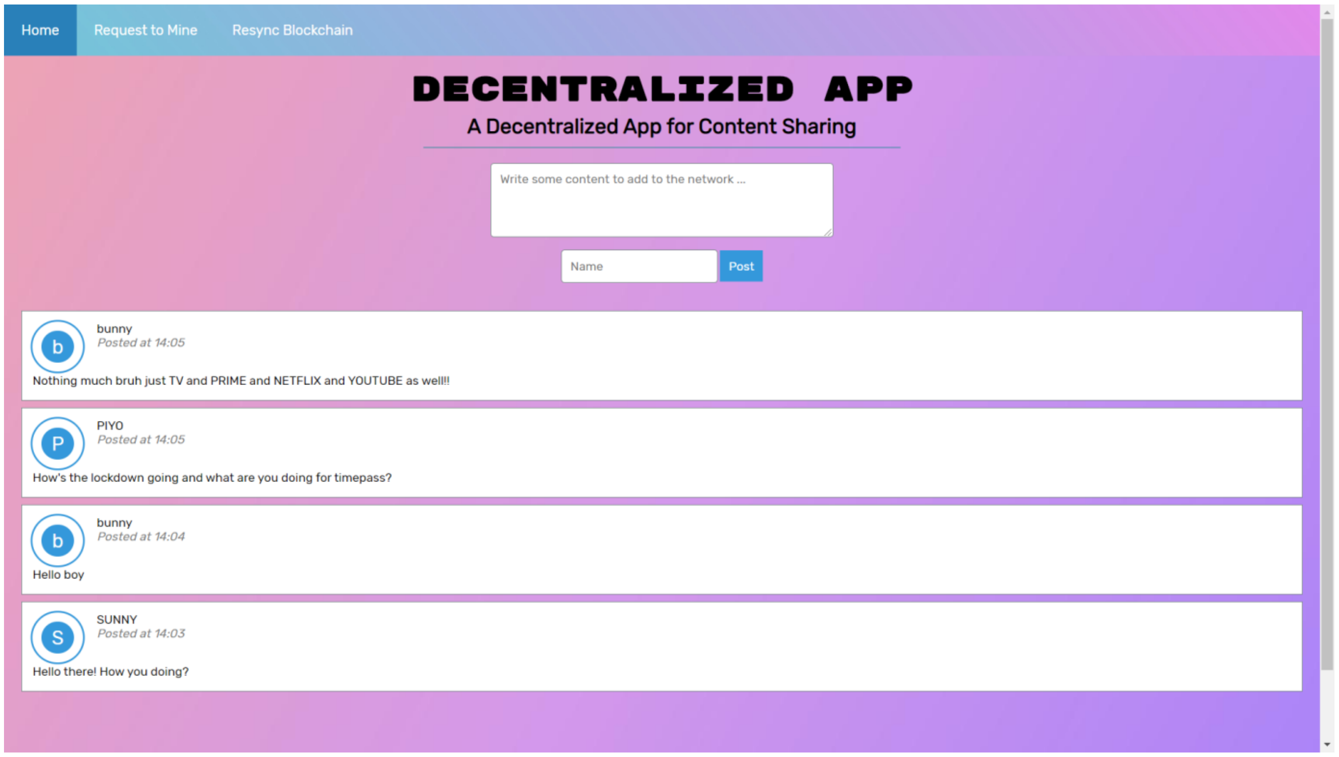
FINAL RESULT OF THE APPLICATION DEVELOPED (USER'S SCREEN)


Home
Request to Mine
Resync Blockchain
DECENTRALIZED APP
A Decentralized App for Content Sharing
Write some content to add to the network ...
Name
Post

XYZ
Posted at 11:41
Hai, I'm a mad guy!

Sunny
Posted at 11:40
Hello there!

Fig 7.1 OUTPUT OF THE PROJECT AFTER RESYNCHRONIZING BLOCKCHAIN


Home
Request to Mine
Resync Blockchain
DECENTRALIZED APP
A Decentralized App for Content Sharing
Write some content to add to the network ...
Name
Post

bunny
Posted at 14:05
Nothing much bruh just TV and PRIME and NETFLIX and YOUTUBE as well!!

PIYO
Posted at 14:05
How's the lockdown going and what are you doing for timepass?

bunny
Posted at 14:04
Hello boy

SUNNY
Posted at 14:03
Hello there! How you doing?

Fig 7.2 FINAL OUTPUT OF THE PROJECT AFTER RESYNCHRONIZING ALL USER'S BLOCKCHAIN.

The mining function links all the blocks generated by the users after resynchronizing the blockchain and the messages are displayed as in the figure 7.2

After we have resynchronized the blockchain of the all users as well then the final output would be like shown and the number of messages is directly proportional to number of users posting the blockchain which are hashed using sha-512 encryption algorithm which makes it difficult to decode as the decoder has to decode 1024 bits in various layers (rounds) of encryption.

7.2 DISCUSSION ON GOAL OF THE PROJECT

THE MODEL OF DECENTRALIZED ENERGY TRADING

The prominent example of a trading energy system that uses blockchain is Brooklyn microgrid as shown in figure 7.2 below which is designed in the USA. It can be described as a solution that combines the security and transparency between the neighbors that is offered by the blockchain concept. The goal of the system is to measure the ability of blockchain technology adapted to buy and sell the energy among the neighbors and how effective blockchain technology is adopted.

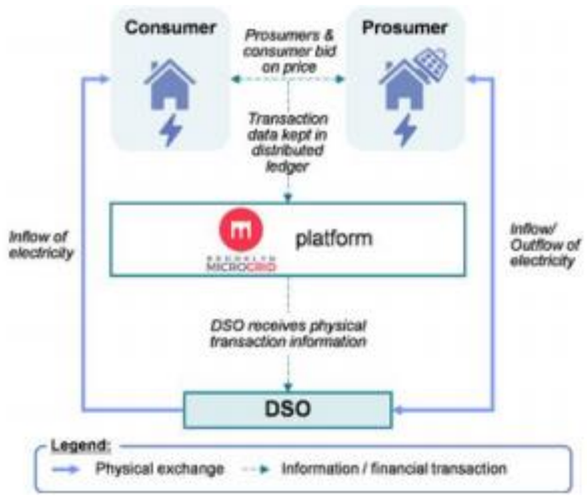

Consumer
Prosumers & consumer bid on price
Prosumer
Transaction data kept in distributed ledger
Inflow of electricity
platform
BROOKLYN MICROGRID
Inflow/ Outflow of electricity
DSO receives physical transaction information
DSO
Legend:
Physical exchange
Information / financial transaction

Fig 7.3. Brooklyn microgrid network

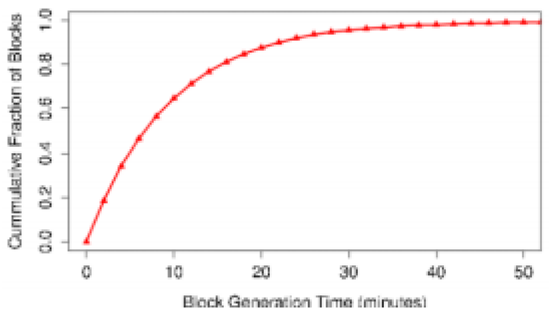


Fig 7.4. Block generation time in Bitcoin

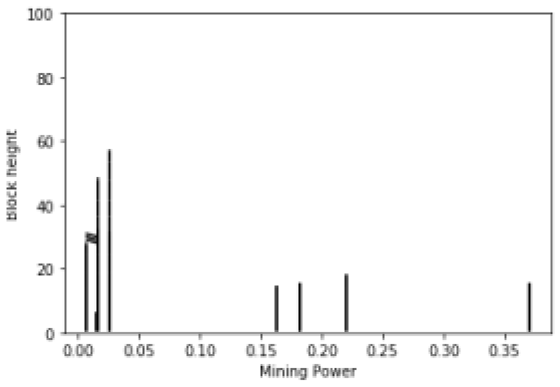


Fig 7.3. Performance of dishonest miner

Decentralized storage of the transaction on the blockchain system allows keeping a distribution, transparency and secure record of energy transaction between the party which is tamper-proof from the attacker. The decentralized energy trading system would no longer require third-party involvement e.g. banks, energy companies to do the energy trading activities among the traders. Instead, the process will run automatically and the energy will be sent after the miner solving the proof-of-work and propagates to the entire network which in the end the presume will get his/her reward. The router transmits the packet data of record transactions from a source to another computer in the peer-to-peer network. The wireless routers use WPA-PSK and PSK Pass Phrases to connect with other devices and every message is encrypted by using AES (Advanced Encryption Standard). The presume who has surplus energy and wants to sell the energy announces to the network by giving details of the information related to the amount of energy and the price.

The consumers will be able to see that announcement and if the consumers want to do the transaction, it will execute by the miners via smart contract. After a transaction is broadcast to the Bitcoin network. When that happens, it is said that the transaction has been mined at a depth of 1 block. With each subsequent block that is found, the number of blocks deep is increased by one. To be secure against double-spending, a transaction should not be considered as confirmed until it is a certain number of blocks deep. Cumulative distribution function (CDF) of the block based on figure 6, approximately 30% of Bitcoin blocks take between 10 and 40 minutes to be generated [13]. In the selfish mining attack, an attacker tries to find a new block by solving the proof-of-work puzzle and keep the block secret and doing mining continuously till they reach the longest chain on the blockchain network. The selfish chain publishes its secret block if only the honest network comes close to their secret network or when the selfish chain wants to claim the unfair rewards. It will affect the rational miner to join in the selfish mining pool. The rational miners are preferred to join the pool with the highest revenue.

The selfish miner is relying on power mining (resources) and always competes with the honest miner to find a new block. Once their network becomes the longest, the selfish miner will easily to invalidate the valid block from the honest miner. The current assumption of the Bitcoin system is safe as long as 51% of the mining power is under the honest miner, but Eyal and Sirer show the attacker can gain the unfair revenue with 25% hashing power. The components model in the network architecture performs a key role in supporting the continuity of a system and it has the input and output gateway of the data blocks.

CHAPTER-8

CONCLUSIONS & FUTURE SCOPE

8.1.CONCLUSIONS

The concept of blockchain technology is used for trading the renewable energy system in an environment among the neighbors in the peer-to-peer network. We discussed a model for trading energy in a small environment by using blockchain technology and then we analyzed security issues that might be occurred. The performance of the attacker is also presented. Blockchain technology with cryptographic embedded to support the security issues can become a possible solution for the future to create a secure trading renewable energy system in the environment among the neighbors. The energy and commodity transaction life cycle, even for simple transactions, involves a multitude of processes within each company and across market participants. Blockchain turns both currencies and commodities into a digital form without relying on middleman which allows one person to trade with another. For the future, the strategy is needed to prevent the various attacks, especially in the overlay network.

A secure and anonymous decentralized messaging application and built the proof-of concept using the Ethereum platform and the Whisper protocol. The application is capable of sending end-to-end encrypted messages while ensuring that the identity of the sender and receiver are anonymous even with the presence of an adversary controlling most of the network. The encryption algorithm used, the application's resistance to various attacks and network traffic analysis are discussed. Concerning future work, the ability for messages to be retrieved anonymously while offline is one area where the application can be extended. An implementation of this feature would likely use an incentivized storage scheme and might result in reduced levels of anonymity.

8.2.FUTURE WORK

An unexplored issue was accounting for the possibility of the user being offline or an unexpected network failure. The messages intended for a specific user could expire during this interval which would mean that the user would not be able to retrieve those messages. One solution could be a decentralized mail server capable of resending the expired messages to the network with sufficient POW. However, since the mail server would not know whether the recipient had received the message it would have to continue doing so indefinitely while new messages would continue being added. It is unlikely that any mail server could be capable of managing the appropriate POW and in any case, would result in a DoS attack on the Whisper Network.

An alternative solution might be to send the expired message directly to the node upon reconnection however this would result in the exposure of their identity as a recipient. Even if this could be solved by routing the message through the Whisper network upon an anonymous legitimate request by the recipient, there is an issue with storage and incentivizing storage by the participants in the network. One possible implementation could be a ‘social storage’ scheme that would utilize the user’s friends to store messages that correspond to a certain Whisper topic. However, this could potentially result in the recipient’s identity being discovered by an adversary if no mechanism to randomize the topic regularly is put in place.

Other areas for possible improvement include adding the ability to have group conversations. Also, the sharing of asymmetric keys rather than symmetric keys for encrypting the session messages could be explored.

CHAPTER-9

REFERENCES

[1] N.Z. Aitzhan and D. Svetinovic, "Security and Privacy in Decentralized Energy Trading through Multi-Signature, Blockchain and Anonymous Messaging Streams", **IEEE Transactions on Dependable and Secure Computing, 2016**.

[2] PwC Global Power, "Blockchain-an Opportunity for Energy Producers and Consumers?" PwC Global Power and Utilities, 2016.

[3] G. Karame, E. Androulaki, "Bitcoin and Blockchain Security", Information Security and Privacy Series, United States of America, Artech House, 2016.

[4] P. Danzi, A. Marko, C. Stefanovic, and P. Popovski, "Distributed Proportional-Fairness Control in Microgrids via Blockchain Smart Contracts", **arXiv: 1705.01453v2 [cs.MA], 2017.**

[5] F. Imbault, M. Swiatek, R.de Beaufort, and R. Plana, "The Green Blockchain Managing Decentralized Energy Production and Consumption", IEEE International Conference on Environment and Electrical Engineering **(EEEIC/ I&CPS Europe), 2017**.

[6] S. Nakamoto, "**Bitcoin: A Peer-to-peer Electronic Cash System", 2008**.

[7] P. Narayan, "Building Blockchain Projects: Develop realtime DApps using Ethereum and JavaScript", **Birmingham-Mumbai, Packt Publishing Ltd. 2017**.

[8] C. Troncoso, M. Isaakidis, G. Danezis and H. Halpin, Systematizing Decentralization and Privacy: Lesson from 15 Years of Research and Deployments, Proceeding on Privacy Enhancing Technologies: **307-329, 2017.**

[9] K.S. Ramesh, M. Kasbekar, W. Lichtenstein, J. Manish, "Overlay Network: An Akamai Perspective", **University of Massachusetts, Amherst, Akamai Technologies Inc***

[10] Chord (peer-to-peer): https://en.wikipedia.org/wiki/Chord

[11] S. Ion, R. Morris, D. Karger, M. Frans Kaashoek, H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", **SIGCOMM'01, San Diego, California, USA, 2001**.

[12] Brooklyn Microgrid, Available online at https://www.brooklyn.energy/

[13] Average Confirmation Time in Bitcoin: https://blockchain.info/charts/avg-confirmation-time

[14] I. Eyal, and E.G. Sirer, "Majority is not Enough: Bitcoin Mining is Vulnerable", Financial Cryptography and Data Security, Berlin, 2014.

[15] N. Aitzhan and D. Svetinovic, "Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams", *IEEE Transactions on Dependable and Secure Computing*, **vol. 15, no. 5, pp. 840-852, 2018.**

[16] B. Lee, M. Lee, H. Ko, S. Myung, M. Kim and J. Lee, "A Secure Mobile Messenger Based on Ethereum Whisper", *The Journal of Korean Institute of Communications and Information Sciences*, **vol. 42, no. 7, pp. 1477-1484, 20**