VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI - 590018, KARNATAKA

A Project Report
on

DECENTRALISED CHATROOM APPLICATION USING SMART CONTRACTS ON ETHEREUM NETWORK

Submitted in partial fulfillment of the requirements for the VIII Semester of degree of ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belagavi*

Submitted by

Apoorv Setpal	**1RN16IS015**	**Aryan Kapoor**	**1RN16IS017**
Paritosh S	**1RN16IS063**	**Rohit Kumar**	**1RN16IS125**

Under the Guidance of

Mr. R Rajkumar
Assistant Professor
Department of ISE

Department of Information Science and Engineering

RNS Institute of Technology

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru-560098

2019-2020

RNS INSTITUTE OF TECHNOLOGY

Dr. Vishnuvaradhan Road, Rajarajeshwari Nagar post,
Channasandra, Bengaluru - 560098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the project work entitled ***Decentralised Chatroom Application using Smart Contracts on Ethereum network*** has been successfully completed by **Apoorv Setpal (1RN16IS015), Aryan Kapoor (1RN16IS017), Paritosh S (1RN16IS063) and Rohit Kumar (1RN16IS122),** bonafide students of **RNS Institute of Technology, Bengaluru** in partial fulfillment of the requirements for the award of degree in **Bachelor of Engineering in Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year **2019-2020**. The project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Mr. R Rajkumar Project Guide Assistant Professor Department of ISE	**Dr. M V Sudhamani** Professor and HOD Department of ISE RNSIT	**Dr. M K Venkatesha** Principal RNSIT

External Viva

Name of the Examiners	**Signature with Date**
1. _____	**1.** _____
2. _____	**2.** _____

DECLARATION

We, **APOORV SETPAL [USN: 1RN16IS015], ARYAN KAPOOR [USN: 1RN16IS017]**, **PARITOSH S [USN: 1RN16IS063], ROHIT KUMAR [USN: 1R16IS125]** students of VIII Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Project entitled ***Decentralised Chatroom Application using Smart contracts on Ethereum network*** has been carried out by us and submitted in partial fulfillment of the requirements for the *VIII Semester of degree of* ***Bachelor of Engineering in Information Science and Engineering*** *of Visvesvaraya Technological University, Belgaum* during academic year 2019-2020.

Place : Bengaluru

Date :

APOORV SETPAL (1RN16IS015)

ARYAN KAPOOR (1RN16IS017)

PARITOSH S (1RN16IS063)

ROHIT KUMAR (1RN16IS125)

ABSTRACT

A blockchain is a distributed append-only server running within a peer-to-peer (P2P) network. In such networks, each peer has a partial or complete copy of the blockchain. Owing to their decentralized existence, blockchains are readily accessible and failure tolerant even if there is a large-scale attack on the system. Thus the blockchain has gained widespread attention as a means of ensuring data integrity, privacy and accessibility in an untrusted environment.

The distributed chat-client software is designed to protect information from internal and external cyber attacks by using the Ethereum blockchain network's aggregated power. The proposed methodology aims to create a decentralized messaging application enabling a P2P messaging platform on the Ethereum network in which there is no manager or administrator. An intermediary sends or receives the messages using secure SHA-256 algorithm. The user will be able to safely and anonymously send and receive encrypted messages. The novel features on blockchain such as broadcasting and group messaging are also contributed by means of the application. To build this blockchain network, the Ethereum platform will be used. As Ethereum follows a decentralized design and adaptability of its communication protocol, this application would be resilient to most suppression tactics.

Blockchain technology has been seeing widespread interest as a means to ensure the data integrity, confidentiality and availability in an untrusted environment. They are designed to protect data from both internal and external cyber attacks by utilizing the aggregated power of the network to resist malicious efforts. In this project, a decentralized messaging application is developed, which utilizes the Ethereum Whisper protocol. Encryption techniques have been known to obscure content, such that it is available only to the intended users. But certain information needs to be available to specific groups of people, and it invites additional risk of the information getting manipulated. Blockchains tackle these issues of data being tampered with. When data is accessed and updated, any change made is recorded and verified. Thereafter, it is encrypted so that further changes cannot be made. These changes are then updated into the main records. This application would be resistant to most suppression tactics due to its distributed nature and adaptability of its communication protocol.

ACKNOWLEDGMENT

At the very onset we would like to place our gratefulness to all those people who helped us in making this project a successful one.

Coming up, this project to be a success was not easy. Apart from the shear effort, the enlightenment of our very experienced teachers also plays a paramount role because it is they who guide us in the right direction.

First of all, we would like to thank **Management of RNS Institute of Technology** for providing such a healthy environment for the successful completion of project work.

In this regard, we express our heartfelt and sincere gratitude to our beloved Director **Dr. H N Shivashankar** and the Principal **Dr. M K Venkatesha,** for providing us all the facilities in this college.

We are extremely grateful to our own and beloved Professor and Head of Department of Information Science and Engineering, **Dr. M V Sudhamani**, for having accepted to patronize us in the right direction with all her wisdom.

We place our heartfelt thanks to **Mr. R Rajkumar**, Assistant Professor, Department of Information Science and Engineering for having guided for project and all the staff members of our department for helping us out at all times.

We thank **Mr. Rajkumar R** and **Mrs. Kusuma S,** Project coordinators, Department of Information Science and Engineering, for their timely support. We would also like to thank all the teaching and non-teaching staff for helping us throughout the project. We thank our beloved friends for having supported us with all their strength and might. Last but not the least, we thank our parents for supporting and encouraging us throughout. We made an honest effort in this assignment.

APOORV SETPAL
ARYAN KAPOOR
PARITOSH S
ROHIT KUMAR

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

GSM	Global System for Mobile Communication
DHT	Distributed Hash Tables
P2P	Peer-to-Peer Systems
EVM	Ethereum Virtual Machine
SPOF	Single Point of Failure
dApp	Decentralized application
DOM	Document Object Model
SMTP	Simple Mail Transfer Protocol
PKI	Public Key Infrastucture
NPM	Node Package Manager
HTTP	Hyper Text Transfer Protocol
ABI	Abstract Binary Interface
RPC	Remote Procedure Call
IPFS	Inter Planetary File System
IPNS	Inter Planetary Name System
SHA	Secure Hashing Algorithm
DAG	Directed Acyclic Graph

Chapter 1

INTRODUCTION

1.1 Background

One of the biggest cybersecurity issues faced by individual computer users as well as corporate firms is data theft, not only because it threatens an individual's privacy, but also because it defeats one of the primary purposes of cybersecurity, i.e. confidentiality.

Over the last few decades, several techniques have been proposed to deal with the issue, and many of them have been short lived, the reason being highly skilled cyber criminals. The latest addition is the Blockchain Technology. Data dispersed over the network is prone to pilferage and plagiarism and often it is impossible to trace back to the cyber criminal. Blockchain technology eliminates the issue on many levels. A blockchain may be defined as a distributed database incorporating information or a book that marks all the events and transactions, executed and shared among concerned parties. The transactions are verified and information entered can never be erased. Every transaction made had a verifiable record. Blockchain technology finds its use in financial as well as non-financial sectors.

Blockchains are public registers such that all transactions are accumulated in list of blocks. When several blocks keep on adding, it leads to a chain like formation. Blockchain Technology is primarily based on the concept of Cryptography and Distributed Systems. Encryption techniques have been known to obscure content, such that it is available only to the intended users. But certain information needs to be available to specific groups of people, and it invites additional risk of the information getting manipulated. Blockchains tackle the issue. When data is accessed and updated, any change made is recorded and verified. Thereafter, it is encrypted so that further changes cannot be made. These changes are then updated into the main records. It is a repetitive process and every time a change is made, the information is preserved in a new block. It is fascinating to note that the first version of the information is well connected to the latest one. Thus, the changes made could be seen by everyone, but only the latest block can be modified.

Blockchain imitates a distributed database by incorporating information duplicated across the network in real time. This means that the database has multiple locations and the records are public and easily verifiable. Since there is not centralized version, data corruption is futile. Modifying records is tedious, thus making it easier to detect if someone is trying to do so.

Blockchains are classified as permission-less blockchains and permissioned blockchains. Permission-less blockchains are open, they can be connected and abandoned by any peer as are a reader or writer. They are decentralized and the information is readable by users. Permissioned blockchains authorize limited readers and writers. They are managed by a central entity which decides that individuals can read or write.

1.1.1 Smart Contracts

Smart contracts based on blockchain is one of the bleeding-edge technologies of the digital age. It is considered a breakthrough technology that can be beneficial for multiple domains including government, healthcare, law, real estate, supply chain and financial services. Smart contracts are self-executing contracts that eliminate the need for intermediaries in the contracting system. They are able to do this because they run in a decentralized and conflict–free ecosystem, saving on time, money and manual effort. Smart contracts also ensure safety and prevent loss of documentation as everything is backed up.

As per experts, there is no universally accepted definition for smart contracts but in order to consider a transaction to be a smart contract transaction must involve more than a transfer of a virtual currency from one person to another and implementation of contract requires no direct human involvement after the smart contract has been made a part of the blockchain. This last element that makes these contracts “smart”.

Smart contracts can be coded on multiple blockchain platforms, Ethereum being the most popular one. This could lead to the question of why a bitcoin protocol is not the choice here. The answer is simple. The foremost objective of introducing an Ethereum platform was to support Smart contracts. The bitcoin protocol is in a constantly evolving state and it offers fewer functionalities as opposed to the Ethereum platform.

However, it has to be noted that Ethereum's protocol is built to allow for flexibility and functionality that provides the capability to program different types of Smart contracts within the system. Ethereum is written in Turing complete language, which includes seven different programming languages. This is quite different from bitcoin, which is written in C++ language.

Ether is the cryptographic token offering from the Ethereum platform and acts as a payment exchange for executing the terms or more precisely for running a line of code. Thus, for executing a Smart contract or for a transaction, Ether will be consumed, and the value will be based on how process-intensive the line of code is.

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 cyberattacks focused on compromising messaging applications highlight the need for maintaining the anonymity of communicating parties. In this project, a decentralized messenger application is developed utilizing the Ethereum Whisper protocol. Using this application, two users can engage in secure and anonymous communication which is encrypted end-to-end and resistant to network traffic analysis.

This proposed project makes the use of decentralized Application approach (dApps). In the proposed work, all the user data is stored on a block which is connected to other blocks forming a chain. As the name suggests, a decentralized application does not have a centralized server. Decentralized Application consists of multiple nodes connected to each other in a mesh topology type network. They are connected to each other in a Peer-to-Peer fashion. Blockchain is a sequence of blocks, which holds a complete list of transaction records like conventional public ledger.

1.1.2 Secure Messaging

A significant amount of current electronic communication is still placed over a number of legacy protocols such as SMS/GSM, SMTP and centralized messengers which 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. In addition, 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.

However, these P2P solutions only partially anonymize the recipient and sender. In addition, they do not provide any capability to hide which participants are engaged in a conversation, and do not prevent protocol messages from being associated to a particular conversation. Furthermore, it is possible for global network adversaries to view the flow of traffic between the participants of a conversation.

1.2 Existing Systems and their Drawbacks

Many Internet applications and services face the challenge of explosive attacks on centralized servers that render the whole architecture helpless and vulnerable within a short period of time. Behind the complexity of Internet, there exist masses of real time response to user request, distribution of tasks or transactions, message pushing, log analyzing and personality recommend service. These tasks are usually driven by asynchronous operations and queue is one of the best solutions to asynchronous problems. Message-oriented Middleware (MOM) is software or hardware infrastructure supporting sending and receiving messages between distributed systems. It provides distributed communication on the basis of the asynchronous interaction model. In order to keep a low latency when system meets heavy traffic, throughput often drops down, similarly in the opposite way, high throughput often means high latency.

Most of the traditional messaging systems are far from ability to deal with the problems above because traditional message queue protocols represented by AMQP often formulate too many semantics for reliability, acknowledgments and transactions. As a result it is hardly probable to build a messaging systems based on these protocols enjoying both high performance and good elasticity. On the other hand, there exist some implementations of distributed messaging systems such as Amazon SQS, EQS and Kafka etc. But some of them do not scale well when system load increases, or they are not suitable to deal with the situations above.

1.2.1 Shortcomings in centralized systems

Researchers have been working on the implementation of anonymous transactions since the 1980s. Before the advent of Bitcoin, academia has established solid foundations in this topic. The blockchain concept, the fundamental form of public ledger, was first introduced for time-stamped digital documents in 1991. Later, Merkle tree was incorporated into the cryptographically secured chain by allowing several documents to be collected into one block, which improves the system efficiency and reliability. However, such a ledger implemented with a chain of blocks is still a centralized database, which relies on the maintenance of a trusted third party financial institute.

1.2.2 Synchronization Issue

Centralized systems are criticized for their vulnerability, due to the single-point-of-failure (SPOF) issue. By contrast, decentralized systems implemented in a distributed manner suffer from the data synchronization issue, which is well summarized as the Byzantine Generals' Problem. In other words, the participants in the decentralized ledger system need to achieve consensus, an agreement upon every message being broadcasted to each other. A common Byzantine fault tolerance can be achieved if the "loyal generals", the honest peers in this context, have a majority agreement on their decisions. Nevertheless, intruders may perform Sybil attack to control a substantial fraction of the public P2P system by representing multiple identities, which may lead to a critical "Double Spending" issue in the blockchain-empowered decentralized ledger.

1.2.3 Double Spending Issue

Thanks to the hash-linking feature of the blockchain, each coin in the ledger can be traced back to the first record when it was created. Therefore, forgery on a non-existing coin is impossible in a public decentralized ledger. However, different from a physical coin, a digital coin can be easily replicated by duplicating the data. In this context, it is critical to prevent the dishonest behavior of spending a coin more than once. If a dishonest user of the public ledger is capable of performing a Sybil attack, the coins that the user double spends will be legalized by the majority of parties, which diminishes user trust as well as the circulation and retention of the currency.

1.3 Proposed System

This section discusses a one-to-one messaging service on a distributed network system in which no manager or administrator exists. A message is sent by a sender, then the receiver receives the message via an intermediary. It is assumed using the Whisper protocol in a situation in which a message contains large data such as movie files, and broadcast is small data. Therefore, the cost of broadcast is not a problem. The permission of send a message is the source of the incentive. This messaging is recorded in the blockchain to develop an autonomous distributed network system. Encapsulated information in a message is needed to chain a block to a blockchain. When the receiver receives a message, the intermediary obtains the confirmation of receipt. The reward is permission to send a new message. Therefore, relaying communications makes incentive.

There are many nodes on a network, and one node can directly communicate with only some of the others. A node communicates with another node that cannot directly communicate via an intermediary. This section considers unicast as well as broadcast messaging. A message is sent by a sender, relayed by an intermediary, and received by a receiver constantly. A participating node is allowed to participate but cannot leave. The communication to synchronize blocks is called “broadcast”. Broadcast is spread without incentive, and each node helps in the broadcast of communication. All blocks are shared by all nodes in the system by broadcast. However, detailed methods of broadcast have not considered. A nonce is replaced from Bitcoin with proof communication relay between participants.

When the transactions in a block are confirmed, the block is chained to the blockchain. A chaining block takes a certain amount of time to prevent malicious participating nodes from confirming wrong transactions. Therefore, a block including wrong transactions are rarely chained to a blockchain if many participating nodes chain blocks competitively. This mechanism is called the “proof of work”. By providing a permission to chain within a certain time that is proportional to the calculation power, when the calculation power of malicious participants is not more than that of honest participants, blocks including only valid transactions are chained. If a malicious node confirms a wrong transaction, the blockchain splits into a block including only valid

transactions and one including wrong transactions. In Bitcoin, the longest blockchain is adopted, so a malicious node has to win the competition that nodes primarily discover a nonce from the time to generate a wrong transaction to eternity. This requires more than half the calculation power of the system.

1.4 Advantages of Proposed System

- **Secure end-to-end Transmission**: The user is enabled with seamless confidential communication in a trustless environment, due to decentralized nodes with untraceable data.
- **Anonymous Participants**: The set of participants engaged in a conversation cannot be determined.
- **Unlinkability**: Only the participants engaged in a conversation are able to 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.
- **Plausible deniability**: It is possible for the any entity to deny having sent a particular message as the nodes only store encrypted message, not the sender's information, and the access to that node is given by proof-of-work only to concerned users.

Chapter 2

LITERATURE REVIEW

A literature review is a type of review article. A literature review is a scholarly paper, which includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literature reviews are secondary sources and do not report new or original experimental work. Most often associated with academic-oriented literature, such reviews are found in academic journals and are not to be confused with book reviews that may also appear in the same publication. Literature reviews are a basis for research in nearly every academic field. A narrow-scope literature review may be included as a part of a peer-reviewed journal article presenting new research, serving to situate the current study within the body of the relevant literature and to provide context for the reader.

Blockchain is a decentralized ledger or data structure. It can be referred as blocks in a chain where the corresponding blocks refer to the blocks, prior to them. Once the details of the transactions or events are fed into the Blockchain, it is impossible to tamper the details are shared with the members of the network. Users of the Blockchain network are completely aware of the transactions taking place. An analogy will be drawn to justify the concept. It is considered to be a book based data structure where each page of the book refers to its previous page by a page. Here, book refers to the Blockchain, page refers to the book and an entry in any page refers to the blockchain transaction. It is easy to detect whether a page or a block has been tampered or not. Pages can be arranged in any manner so pages aren't important in a distributed ledger.

According to [1], each block is built on top of the previous block and it uses the latter's nonce and signature as a key for going into the next block. Miners of the network do the job of building a block and adding it to the chain. It is easy for the miners to guess a random string or nonce in order to tamper the block just by knowing the signature in a Public Blockchain. It is not easy to add blocks in the Blockchain and there is a reward of 12.5 bitcoins for that. In a Private Blockchain, the miners are given a contract.

As discussed by the author in [2], the rapid growth of Blockchain technology over the recent years has opened up a plenty of research gaps and directions for the research community. As a result, a remarkable amount of research endeavours have been conducted within the domain of Blockchain in recent years. Based on this data, more than 1000 scientific papers have been indexed only by Web of Science (WoS) in recent years. As the number of research publications in the Blockchain domain is increasing, there is a demand for conducting research studies in which a comprehensive overview of the current body of knowledge in this field is investigated. To fulfil this demand, a few review papers have been published in order to provide the researchers and practitioners with the recent achievements and challenges in the Blockchain community. The main objective of this paper is to systematically collect, characterize and analyse all the Blockchain research papers that have been indexed by WoS Core Collection.

Briefing the conclusions in [3], it is understood that blockchain is a form of database storage that is non-centralized, reliable, and difficult to use for fraudulent purposes. Bitcoin, on the other hand, is a form of digital currency that uses a Blockchain public ledger to make transactions across peer to peer networks. Bitcoin is just one of the financial applications that use Blockchain technology; there are also others such as smart contract and hyperledger. Blockchain technology can therefore be used to create many applications. The technology that has had the most impact on our lifestyles in the last decade is Blockchain. A word that often arises when talking about Blockchain is Bitcoin. Many people still confuse Blockchain with Bitcoin; however, they are not the same. Bitcoin is just one of many applications that use Blockchain technology. In this paper, the authors conduct a survey of Blockchain applications using Blockchain technology and the challenges these face.

As per the discussion in [4], the authors define Blockchain as a decentralized and distributed digital ledger that is used to record transactions across many computers so that the record cannot be altered retroactively without the alteration of all subsequent blocks and the collusion of the network". The formation of Blockchain is such where the longest chain, called the main chain (active chain), comes from the genesis block and the orphan block is the block that exists outside the main block. According to Christian Cachin et al. the Blockchain has 4 elements that are replicated: the ledger, cryptography, consensus

and business logic. For more information about these characteristics the reader can consult.

In [5], it's been observed that recently, it is becoming increasingly vital that communication not only be secure but also anonymous. The presence of mass surveillance programs as well as cyberattacks focused on compromising messaging applications highlight the need for maintaining the anonymity of communicating parties. In this article, a decentralized messenger application is developed utilizing the Ethereum Whisper protocol. Using this application, two users can engage in secure and anonymous communication which is encrypted end-to-end and resistant to network traffic analysis.

The author describes the modern scenario in [6] that the blockchain has fuelled one of the most enthusiastic bursts of activity in applied cryptography in years, but outstanding problems in security and privacy research must be solved for blockchain technologies to go beyond the hype and reach their full potential. At the first IEEE Privacy and Security on the Blockchain Workshop, peer-reviewed papers were presented, bringing together academia and industry to analyse problems ranging from deploying newer cryptographic primitives on Bitcoin to enabling use-cases like privacy-preserving file storage. The overview is not only the larger problems the workshop has set out to tackle, but also outstanding unsolved issues that will require further cooperation between academia and the blockchain community.

As mentioned in [7], a blockchain is simply a cryptographically verifiable list of data. One of the reasons for the enthusiasm around the blockchain is that databases do not have any cryptographic guarantees of integrity, guarantees that are necessary for any database operating in an adversarial environment. If the field of systems security and privacy-enhancing technologies has learned one lesson since the Snowden revelations, it is that all databases are likely operating in an adversarial environment. Therefore, some of the "hype" around blockchains is for good reason. As exemplified by Bitcoin, the primary advantage of blockchain technologies is that the data itself can be decentralized. A distributed public ledger built with a blockchain where all users have the same data, which is necessary for high-value use-cases such as currency, is clearly privacy-invasive for many use-cases.

According to the author in [8], an autonomous distributed system, such as an Ad hoc network or peer-to-peer-based file sharing system, consists of nodes contributing to the system. However, egoistic nodes could collapse such a system. A distributed system of a peer-to-peer (P2P) or ad hoc network consists of nodes contributing to the system, i.e., relaying communication between participants. Therefore, such a system could collapse when egoistic nodes do not relay the communication because there is no advantage for such a node to participate in the system. As a general rule, the blockchain, i.e., the entire all data blocks are shared with all nodes in Bitcoin by broadcast which propagates data of the blockchain. Therefore, the nodes of Bitcoin can confirm transactions. Multiple transactions are gathered in a block. A block contains transactions, the hash of the previous block, and nonce.

As it has been concluded in [9], the cost of cyber-crime costs quadrupled from 2013 to 2015 however a large portion of cybercrime goes undetected. Gartner report says cost of cyber-crime is expected to reach $2 trillion by 2019. IBM's CEO, Ginni Rometty said that cybercrime is the greatest threat to every company in the world at IBM Security Summit. Around two years ago Standard Chartered lost around $200 million in a fraud at China's Qingdao port. Banking and financial institutions are using Blockchain based technology to reduce risk and prevent cyber fraud. Blockchain can play crucial role in Internet of Things(IoT) and development of smart systems since the history of individual devices can be tracked by tracking a ledger of data exchanged. It can enable smart devices to act like an independent agent which can autonomously perform several transactions. For example, smart home appliances competing with one another for priority so that laundry machine, thermostats, dishwasher and smart lighting run at an appropriate time to minimize cost of electricity against current grid prices.

According to the author in [10], by definition a blockchain is a continuously growing chain of blocks, each of which contains cryptographic hash of the previous block, a time-stamp, and its conveyed data. Maintenance of peer-to-peer (P2P) ledgers for cryptocurrencies has become the first killer application of blockchain. Thousands of cryptographic tokens, or coins, were delivered to the public market, after the huge leap in market cap of Bitcoin . However, due to the lack of legal regulation and auditing, a large number of scams, so-called "air coins", also brought bad reputations to the blockchain

technology. Doubts on the value of cryptocurrencies have been raised. Warren Buffett—the famous billionaire investor—insisted that cryptocurrencies will come to a "bad ending", and claimed that Bitcoin is " probably rat poison squared".Instead of discussing cryptocurrencies, this paper surveys the state-of-the-art of blockchain technology and introduces decentralized applications (dApps), which is a novel form of the blockchain-empowered software system.

As it has been discussed in [11], with the rapid development of cryptocurrency and its underlying blockchain technologies, platforms such as Ethereum and Hyperledger began to support various types of smart contracts. Smart contracts are computer protocols intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts have broad range of applications, such as financial services, prediction markets and Internet of Things (IoT), etc. However, there are still many challenges such as security issues and privacy disclosure that await future research. First is a systematic introduction for smart contracts, including the basic framework, operating mechanisms, platforms and programming languages. Second, application scenarios and existing challenges are discussed. Smart contracts are self-executing contracts with the terms of the agreement between interested parties. The contracts are written in the form of program codes that exist across a distributed, decentralized blockchain network. Smart contracts allow transactions to be conducted between anonymous or untrusted parties without the need for a central authority.

Summing up the conclusions in [12], Blockchain based smart contracts are computer programs that encode an agreement between non-trusting participants. Smart contracts are executed on a blockchain system if specified conditions are met, without the need of a trusted third party. Centralized systems rely on a trusted third party (e.g., banks) to allow non-trusting participants to communicate and send financial transactions between each other. Relying on a trusted third party, however, could result in security and privacy issues as well as high transactional costs. Blockchain technology aims to address this by allowing non-trusting participants to reach a consensus on their transactions and communications without the involvement of a trusted third party. Blockchain has evolved to support a number of decentralized applications beyond financial applications. Many of these applications rely on the execution of smart contracts on top of the blockchain.

Different platforms offer different features for developing smart contracts. Like Bitcoin, Ethereum is a permission-less blockchain and cryptocurrency. In addition to the ability to transfer currency it supports building and running complex applications based on smart contracts on the blockchain.

As the authors describe in [13], Smart contracts are becoming more and more popular nowadays. They were first conceived in 1997 and the idea was originally described by computer scientist and cryptographer Nick Szabo as a kind of digital vending machine. Nick described how users could input data or value and receive a finite item from a machine, in this case a real-world snack or a soft drink. In the example of a crowd-funding platform for supporting projects, the smart contract would hold all the received funds from a project's supporter (it is possible to pay a smart contract). If the project fully meets its funding goals, the smart contract will automatically transfer the money to the project. Otherwise, the smart contract will automatically refund the money to the supporters. The decentralised model of immutable contracts implies that the execution and output of a contract is validated by each participant to the system and, therefore, no single party is in control of the money. No one could force the execution of the contract to release the funds, as this would be made invalid by the other participants to the system. Tampering with smart contracts becomes almost impossible.

A secure communications infrastructure featuring email, chat, and a MIPR application was developed to a proof of concept level using a blockchain database to store the contents. The blockchain, which already provides a completely secure method of exchanging cryptocurrencies, can be extended to any transactional or communications paradigm. Digital transactions require that users trust the system's integrity. Infrastructure and databases must be maintained and kept secure, costly tasks that fall on a central service to perform. Notable hacks have rendered the security of even well-established servers questionable. In 2014-2015, the Office of Personnel Management was broken into and the records of an estimated 21 million people, including security clearance information, were taken . In 2008, U.S. Central Command in Afghanistan was infected with a worm from an infected flash drive which required 14 months to clean and led to the creation of US Cyber Command. The situation is further complicated by requiring of users stronger passwords, the recall of personal questions for identification, and new

procedures like two-factor authentication to maintain the security of sensitive records. For large organizations, the holding of sensitive information poses an awesome responsibility in terms of balancing accessibility and security. This situation can be ameliorated through the use of blockchain databases, the same technology underlying cryptocurrencies and smart contracts. The blockchain, in brief, is both a technology and a social consensus/governance mechanism.

In the modern day scenario as discussed in [14], smart contracts that build up on blockchain technologies are receiving great attention in new business applications and the scientific community, because they allow untrusted parties to manifest contract terms in program code and thus eliminate the need for a trusted third party. The creation process of writing well performing and secure contracts in Ethereum, which is today's most prominent smart contract platform, is a difficult task. The EVM handles the computation and state of contracts and is build on a stack-based language with a predefined set of instructions (opcodes) and corresponding arguments. So, in essence, a contract is simply a series of opcode statements, which are sequentially executed by the EVM. The EVM can be thought of as a global decentralized computer on which all smart contracts run. Although it behaves like one giant computer, it is rather a network of smaller discrete machines in constant communication.

According to [15], smart contracts are designed to facilitate the performance of trackable and irreversible transactions without the need for third party involvement. Therefore, as a result of this lack of oversight, it is essential that these smart contracts are written and properly tested. As a new technology smart contracts are not without numerous risks and challenges, many of which are a direct result of their stated benefits, such as lack of third-party oversight and their immutability. Therefore, they have drawn considerable criticism from many who have observed the resulting chaos. The most prominent framework for smart contracts is Ethereum , whose capitalization has exceeded $12.2B since its mid-2015 launch.

Cryptocurrency platforms such as Bitcoin and Ethereum have become more popular due to decentralized control and the promise of anonymity. Ethereum is particularly powerful due to its support for smart contracts which are implemented through Turing complete scripting languages and digital tokens that represent fungible

tradable goods. It is necessary to understand whether de-anonymization is feasible to quantify the promise of anonymity. Cryptocurrencies are increasingly being used in online black markets like Silk Road and ransomware like CryptoLocker and WannaCry. Ethereumis the second largest cryptocurrency platform after Bitcoin by market capitalization,$24 billion to $53billion(as of August 1, 2017).The currency used in Ethereum is called ether (ETH) whereas the currency used in Bitcoin is also called bitcoin (BTC). Ethereum relies on public blockchain where consensus is maintained by proof-of-work like Bitcoin.

A proof-of-work system is one where miners maintain the blockchain by competing to solve computationally intensive problems (which are easy to validate). Since anonymity is a key promise of cryptocurrency platform and due to the popularity of Bitcoin, much of the existing research deal with weaknesses in Bitcoin and recommendations to fix them. In this paper, the work focuses on applying the approaches from Bitcoin onto Ethereum. Ethereum Virtual Machine (EVM) which provides a runtime environment for smart contracts. Digital tokens that represent digital assets where the issuance is governed by smart contracts. The digital tokenscan be treated as currencies like ETH and BTC andcan be issued through initial coin offerings (ICOs) akin to IPOs.

EtherChat.co is a web app that allow us to interact with EtherChat's smart contract. It's source code is available on Github, so, if https://etherchat.co is not trusted, EtherChat project can be cloned and run the web app on the local computer. A message will be encrypted using an encryption key. The same key will be used to decrypt the message. Only the sender and the recipient can generate the same encryption key. The sender will compute it from his private key and the recipient's public key. The recipient will compute it from his private key and the sender's public key. Every Ethereum account has a private key and a public key associated with it. The private key is what has to be kept in secret (or the wallet software will keep it in secret). The public key will be shared on the blockchain network so that other people can interact with the user's account.

It is written in Solidity and contains logics for handling messages and contacts using solidity events. The contract doesn't store any messages/contacts in its storage. It only generates events, which will be stored on the Ethereum network. The EtherChat's contract **does not** encrypt and decrypt messages. Instead, it is made such that the web

app will do it and submit the encrypted messages to the network. The information that are stored in the contract's storage are its members (who joined EtherChat) and the relationships between these members.

Reference Number	Author	Technique	Results	Limitations
[1]	Rishav Chatterjee Raj Chaterjee	Cryptocurrencies are digital currencies that combine the advantages of a peer-to-peer network with cryptographic techniques.	There are some possible application scenarios of smart contracts such as digital rights management, social media platforms, cloud storage, supply chain, intelligent transportation, etc.	Some challenges faced by smart contracts include Reentrancy vulnerability, Timestamp dependence and privacy issues.
[2]	Muhaammed Dadbagh	Blockchain is a decentralized ledger that stores all transactions.	An overview of existing bibliometric studies of blockchain research.	An in-depth observation of highly cited papers reveals an abundance of complex information.
[3]	Maher Alharby, Aad van Moorsel	A smart contract can invoke and create another smart contract by posting a message. This is used by	Application refers to the utilization of smart contracts to address issues in different domains such as Internet of Thing (IoT). The study also proposes	The challenges are the relative lack of research and performance benchmarking on the scalability

		smart contracts either for creating a new smart contract or for calling other smart contra ct functions.	an adaptive incentive mechanism for smart contract systems to defend against denial of service attacks.	issues of blockchain-based smart contract systems.
[4]	Mohamed Abdulaziz	They are designed to protect data from both internal and external cyberattacks.	Application is able to send encrypted messages both securely and anonymously.	Certain portions of the official documentation are either inconsistent,in correct or outdated.
[5]	Cristopher G. Harris	Smart Contracts are designed to facilitate the performance of trackable and irreversible transactions without third-party involvement.	A contract is an agreement that binds the two parties to a future state.	As mentioned in the paper, smart contracts sometimes require information from external resources. However, the reliability of the information can not be guaranteed.
[6]	Hiroyoshi	An ad hoc	A software system for	Several types

	Ichikawa	network or peer-to-peer-based file sharing system.	sharing files.	of fraud,but only less than half were prevented.
[7]	Peter Menegay	The work described here used two blockchains, BitShares and Steem.which are open-source softwares.	DPoS is fast since it relies on elected "witnesses" who take turns as block producers rather than deciding the block producer through a proof of work competition.	Scalability of the chat application (which requires low latency) over a blockchain remains to be proven, and DPoS might slow it down.
[8]	Nirmala Singh	Each member of the network has access to the latest copy of encrypted ledger so that they can validate a new transa ction.	Blockchain presents an immense possibility in Internet of Things (IoT) and providing security.	Pre-requisites must include Authentication , integrity and non-repudiation.
[9]	Shuai Wang	Smart Contract Architecture and Applications.	Set of protocols to build the application.	There are still many challenges such as security issues

				and privacy disclosure that await future research.
[10]	Pinyaphat Tasatanattakool	A blockchain is an append-only distributed database operating within a P2P network.	To circumvent 'double-spending' a number of transactions are grouped into a block and are then verified simultaneously.	The challenges include Resistance to Denial-of-Service (DoS) attacks, and network traffic analysis.
[11]	Giuseppe Destefanis	A discipline of Smart Contract and Blockchain programming.	Parity Wallet Dependency Graph.	Vulnerability of the library was mainly due to a negligent programming activity rather than a problem in the Solidity language.
[12]	Peter Menegay	A secure communications infrastructure featuring email, chat, and a MIPR application.	Blockchain Based Email Architecture	It is doubtful that blockchains are suited for massive data storage.
[13]	Maximilian Wöhrer and Uwe	Security Patterns in the	The application of a mutex pattern to	There is always a risk

	Zdun	Ethereum Ecosystem and Solidity	avoid re-entrancy.	that a contract gets compromised due to bugs in the code or yet unknown security issues within the contract platform, due to attacks.
[14]	Christopher G. Harris	Challenges involving writing and implementing smart contracts.	Set of modular objects to implement software system.	Number of risks can be observed in scams,bugs and errors.
[15]	Roman Graf	Neural Network and Blockchain.	Solution for Cyber Threat Intelligence and Situational Awareness.	The duration for Smart Contract operation was 0.252 seconds from Blockchain.

Chapter 3

ANALYSIS

In this section, a detailed intervention of the shortcomings faced is assimilated while using the existing system models for secure messaging, and the protocols involved. Various issues that need to be resolved are discussed so that this prototype model stands out as a secure and anonymous messaging tool.

3.1 Problem Identification

Problem identification provides a platform for investigating a broad range of interventions and generating options. Initiatives developed in subsequent steps of the framework should address the problems identified here. The process of problem identification involves the development of clear, straightforward problem statements that can be linked directly with the specific goals and objectives already identified.

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 cyberattacks focused on compromising messaging applications highlight the need for maintaining the anonymity of communicating parties. In this proposed project, a decentralized messenger application is deployed on blockchain network utilizing the Ethereum Whisper protocol. Using this application, two users can engage in secure and anonymous communication which is encrypted end-to-end and resistant to network traffic analysis.

A significant amount of current electronic communication is still placed over a number of legacy protocols such as SMS/GSM, SMTP and centralized messengers which 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. In addition, 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.

However, these P2P solutions such as Kademlia only partially anonymize the recipient and sender. In addition, they do not provide any capability to hide which participants are engaged in a conversation, and do not prevent protocol messages from being associated to a particular conversation. Furthermore, it is possible for global network adversaries to view the flow of traffic between the participants of a conversation.

When identifying problems, the following should be taken into account:

- Problems prevent the goals and objectives identified in the previous step from being achieved. This should include the full range of objectives identified in the previous step – including objectives for different levels of planning and markets.
- Problem identification should consider not only ‘problems’ or ‘challenges’ but also constraints on opportunities that are preventing the goals and objectives from being achieved.

Listed below are characteristics 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 are able to 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.
- **Deniability**: It should be possible for the any entity to deny having sent a particular message.

To ensure this application would satisfy these various security and anonymity features we have decided to build a decentralized messaging application which utilizes Whisper as its communication protocol.

3.2 Objectives

Objectives are more specific and easier to measure than goals. Objectives are basic tools that underlie all planning and strategic activities. They serve as the basis for creating policy and evaluating performance. Due to the availability of large amounts of data acquired in a variety of conditions, techniques that are both robust to uncontrolled acquisition conditions and scalable to large gallery sizes, which may need to be incrementally built, are challenges.

The systems currently used, have a centralized approach to resource sharing and communication. Here, all the data is stored on a centralized server. This may lead to loss of data if the server collapses. Also, there are countless counterfeit information and product publishing on social networking without any known root transgressor (like on WhatsApp, hike). The information shared can be hacked which is stored on the centralized server. Therefore, the main objective behind the proposed model is to develop a software that can provide all the features provided by currently available chat applications as well as overcome the drawbacks that they have. The resultant software will be more secure and reliable than the currently available ones. Some of the key objectives are listed as follows:

[1] **Public key infrastructure (PKI):** It uses public/private key encryption and data hashing to store and exchange data safely.

[2] **Distributed ledgers:** There is no central authority to hold and store the data, it removes the single point of failure.

[3] **Peer to peer Network:** The communication is based on the P2P network architecture and inherits the decentralized characteristics.

[4] **Cryptography:** Blockchain uses a variety of cryptographic techniques, hash functions, Merkle trees, public and private key. It is difficult to alter the blockchain, to make a modification, it is necessary to succeed in a simultaneous attack on more than 51% of the participants.

[5] **Consensus Algorithm:** The rules which the nodes in the network follow to verify the distributed ledger. Consensus algorithms are designed to achieve reliability in a system involving multiple unreliable nodes. A consensus of the nodes validates the transactions. Choice of consensus algorithm has significant implications for performance, scalability, latency and other Blockchain parameters. The consensus algorithms must be fault tolerant.

[6] **Transparency:** Each transaction is visible to anyone with access to the system. If an entry can not be verified, it is automatically rejected. The data is therefore wholly transparent. Each node of a blockchain has a unique address that identifies it. A user may choose to provide proof of identity to others.

3.3 Methodology

The methodology is described as the systematic, theoretical analysis of the methods applied to a field of study. It comprises the theoretical analysis of the body of methods and principles associated with a branch of knowledge. Typically, it encompasses concepts such as paradigm, theoretical model, phases and quantitative or qualitative techniques.

A methodology does not set out to provide solutions it is, therefore, not the same as a method. Instead, a methodology offers the theoretical underpinning for understanding which method, set of methods, or best practices can be applied to a specific case, for example, to calculate a specific result.

It has been defined also as follows:

1. The analysis of the principles of methods, rules, and postulates employed by a discipline.
2. The systematic study of methods that are, can be or have been applied within a discipline.
3. The study or description of methods.

The methodology is the general research strategy that outlines the way in which research is to be undertaken and, among other things, identifies the methods to be used in it. These methods, described in the methodology, define the means or modes of data collection or, sometimes, how a specific result is to be calculated. The methodology does not define specific methods, even though much attention is given to the nature and kinds of processes to be followed in a particular procedure or to attain an objective.

While properly studying the integral concepts in methodology, such processes constitute a constructive generic framework, and may, therefore, be broken down into sub-processes, combined, or their sequence changed.

Methodology and methods are not interchangeable. In recent years, however, there has been a tendency to use methodology as a "pretentious substitute for the word method". Using methodology as a synonym for method or set of methods leads to confusion and misinterpretation and undermines the proper analysis that should go into designing research.

This project implements the following features in the decentralized application.

3.3.1 Secure Messaging

A significant amount of current electronic communication is still placed over a number of legacy protocols such as SMS/GSM, SMTP and centralized messengers which 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. In addition, 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. However, these P2P solutions such as Kademlia only partially anonymize the recipient and sender. In addition, they do not provide any capability to hide which participants are engaged in a conversation, and do not prevent protocol messages from being associated to a particular conversation. Furthermore, it is

possible for global network adversaries to view the flow of traffic between the participants of a conversation.

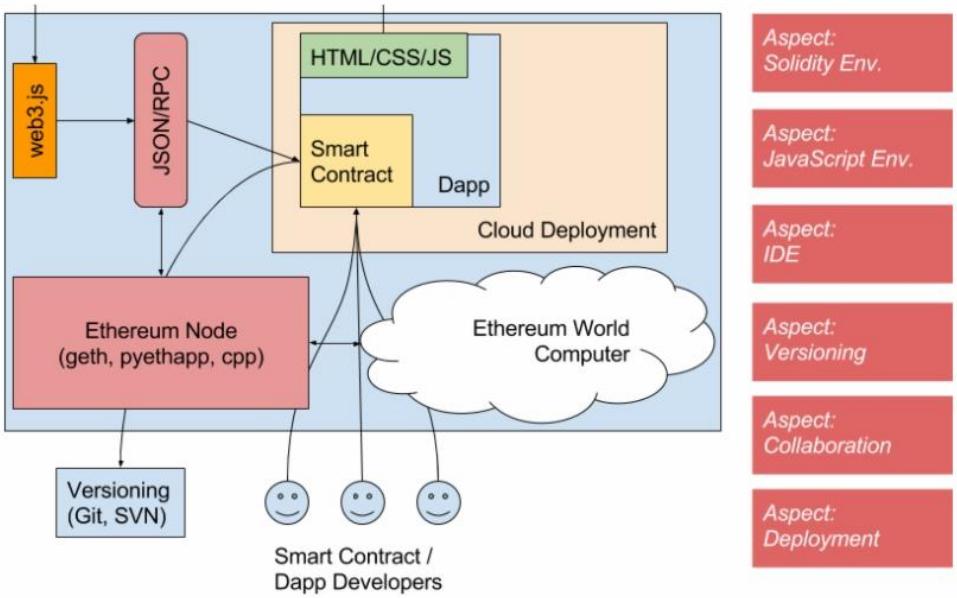

Figure 3.1 The flow of decentralized applications

3.3.2 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 which are broadcasted and verified by all the nodes in the network. After verification the change is appended to the blockchain.

A blockchain is essentially a fault-tolerant distributed database of records of a public ledger of all transactions that have been executed and shared among participating parties. Each transaction in the public ledger is verified by a consensus of a majority of participants in the system. Blockchain provides a tool for increasing data integrity using a combination of cryptography and consensus. As such, there is no central server point of control. The information is stored in these blocks. Each block contains transactions, a timestamp and a link to the previous block, and is cryptographically protected.

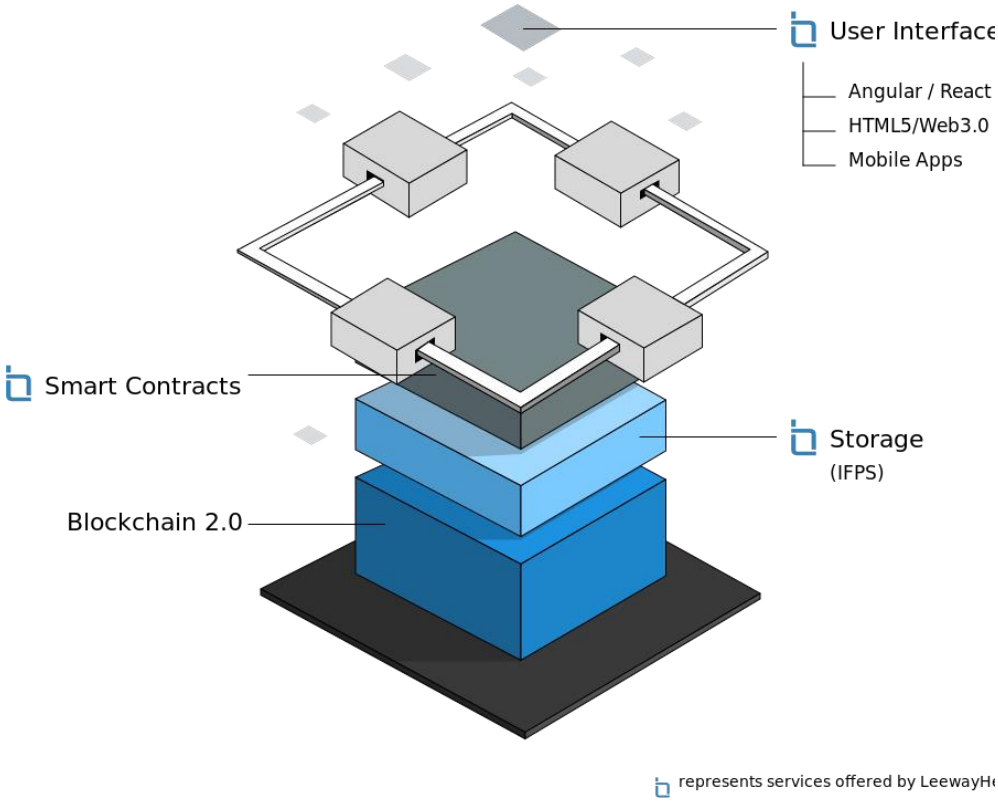

Figure 3.2 The abstraction of layers in decentralized applications

3.3.3 Smart Contracts

A Smart Contract is a computer program that aims to implement a logical sequence of steps according to some clauses and rules. In a conceptual level, Smart Contracts consist of three parts :

- the code of a program that becomes the expression of a contractual logic;
- the set of messages which the program can receive, and which represent the events that activate the contract;
- the set of methods which activate the reactions foreseen by the contractual logic.

Smart Contracts run in a blockchain where contract transactions can be permanently recorded in a transparent environment and are immutable. Once the Smart Contract is deployed into the blockchain its code cannot be modified and the clauses introduced by the parties in the contract will obligatorily be respected because of the computational nature of the system, as for the execution of any software program.

Among all, the most popular is the Ethereum platform, the first blockchain specifically conceived to run Smart Contracts. The most popular programming language

for Smart Contracts in Ethereum is “Solidity”. In this platform it is possible to read some information that characterize each Ethereum transaction.

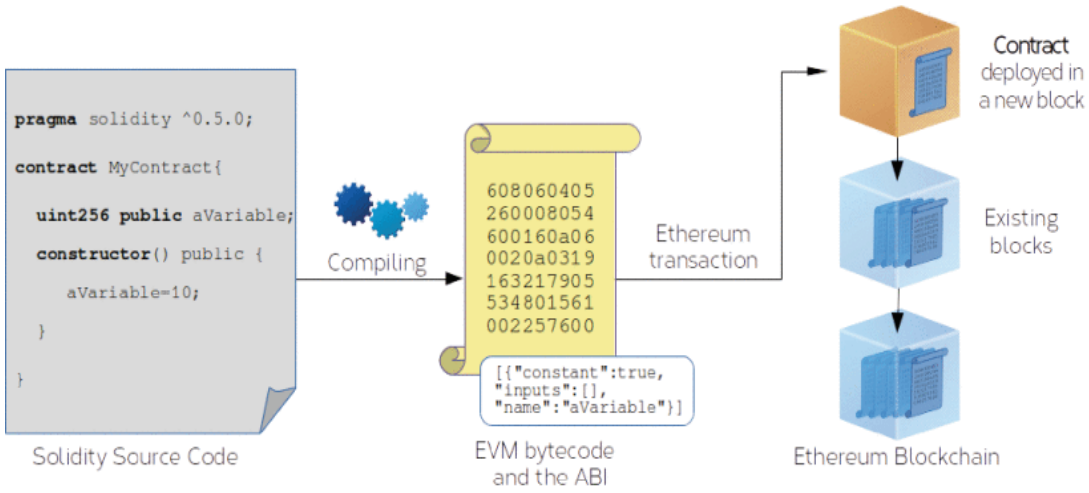

Figure 3.3 Deployment of smart contracts in blockchain network

3.3.4 Solving Public Key Infrastucture using dApp

In blockchain, the shared data is the history of every transaction ever made. The ledger is stored in multiple copies on a network of computers, called nodes. Each time someone submits a transaction to the ledger, the nodes check to make sure the transaction is valid. The transaction is being created in the chained data structure of blockchain, a new timestamp will be recorded at the same time, and any modification of data created before will not be allowed anymore. A block is a data structure which consists of a header and a list of transactions. Each transaction is generally formed as trx:= (M, Signature) where M:= (PK_{Sender}, receiver, data); PK_{Sender} denotes the public key of a sender (address of the sender is derived from the PK_{Sender}), and receiver is the address of a receiver, Signature:=Sign SK_{Sender} (H(M)) where SK_{Sender} is the private key of the sender and H() is a secure hash function. A subset of them competes to package valid transactions into blocks and add them to a chain.

3.4 System Requirement Specification

A System Requirements Specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on how the software product should function. Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

3.4.1 Software Requirement Specification

- **Ganache:** This dependency is a personal blockchain, which is a local development blockchain that can be used to mimic the behavior of a public blockchain. If the user were to create their own personal blockchain network from scratch, or develop the application on a test network, the accounts would have to be created manually and credit each account with ether. Thankfully Ganache already does this for the user, facilitating prototyping for blockchain development..
- **Node.js:** Node.js is an application runtime environment that allows us to write server-side applications in JavaScript. Thanks to its unique I/O model, it excels at the sort of scalable and real-time situations users are increasingly demanding of the servers. It's also lightweight, efficient, and its ability to use JavaScript on both frontend and backend opens new avenues for development. It comes as no surprise that so many big companies have leveraged Node.js in production. Node.js is a good choice for applications that have to process a high volume of short messages requiring low latency. It also comes in with Node Package Manager (**npm)** that offers many dependencies to migrate this model to the public blockchain.
- **Truffle Framework:** Truffle is made for building decentralized applications (dApps) using the Ethereum Virtual Machine (EVM) by providing a development environment, testing framework, and asset pipeline. Some of the functionalities the user gets with Truffle include:
    - **Smart Contract Management** – The user can write smart contracts with the Solidity programming language and compile them down to bytecode that be run on the Ethereum Virtual Machine (EVM).

- **Automated Testing** – User can write tests against smart contracts to ensure that they behave the way user wants them to. These tests can be written in JavaScript or Solidity, and can be run against any network configured by Truffle, including public blockchain networks.
- **Deployment & Migrations** – User can write scripts to migrate and deploy smart contracts to any public Ethereum blockchain network.
- **Network Management** – User can connect to any public Ethereum blockchain network, as well as any personal blockchain network might be used for development purposes.
- **Development Console** – The developers or users may interact with smart contracts inside a JavaScript runtime environment with the Truffle Console. User can connect to any blockchain network that has been specified within network configuration to do this.
- **Script Runner** – The user write custom scripts that can run against a public blockchain network with JavaScript. User can write any arbitrary code inside this file and run it within their project.
- **Client Side Development** – Users can configure their truffle project to host client side applications that talk to their own smart contracts deployed to the blockchain.

- **MetaMask Wallet:** MetaMask is a cryptocurrency wallet which can be used on the Chrome, Firefox and Brave browsers. It's also a browser extension. This means that it works like a bridge between normal browsers and the Ethereum blockchain. It also allows users to browse the Ethereum blockchain from a standard browser. MetaMask requires no login and does not store the private keys in any server, instead they are stored on Chrome and password protected. MetaMask is a dedicated way to allows the user to run Ethereum Apps right in the browser without running a full Ethereum node. This wallet simplifies the Ethereum transaction.

- **Web3.js:** Web3.js is a popular library that allows programmers to interact with the Ethereum blockchain. It represents a JavaScript language binding for Ethereum's

JSON RPC interface, which makes it directly usable in web technology. Web3.js is also commonly used on the server side in Node.js applications and in Electron-based desktop applications. Web3.js can be used to connect to the Ethereum network via any Ethereum node that allows access via HTTP. One common way of integrating a web browser application with Ethereum is to use the MetaMask browser extension in combination with Web3.js.

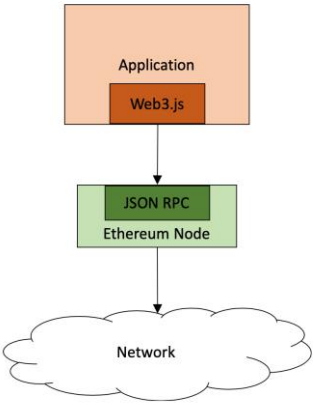

Figure 3.4 Interaction with an Ethereum Node on blockchain

- **React.js:** React.js is a JavaScript library that is being used to build the User Interface. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with rendering data to the DOM, and so creating React applications usually requires the use of additional libraries for state management and routing. Another notable feature is the use of a virtual Document Object Model, or virtual DOM. React creates an in-memory data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently.

3.4.2 Hardware Requirement Specification

- **Processor:** 64-bit Intel i5/AMD Ryzen 3 or higher
- **RAM:** 4GB DDR3 or higher
- **SSD(optional):** 256GB solid state drive for mining blocks faster
- **Memory:** 50GB disk space or higher

3.4.3 Functional Requirement

- User should be able to login to his wallet where his/her Ethers or cryptocurrencies are stored.
- User must be able to send a message securely and anonymously to another user, with the message being delivered tamper-proof.
- User must be able to broadcast the same message to many other accounts simultaneously with one click.
- User must be able to communicate with other users in a chatroom, where all the participating users can send and receive messages to and from each other.
- User must be able to reply to any message that he receives, with a single click.
- User can see the status of his message, while it's being sent and when it is delivered.
- User must be allowed to add any number of recipients in a broadcast, and participants in a chatroom.

3.4.4 Non-functional Requirement

- **Security and Anonymity:** The most outstanding feature of this application is the incorruptible network fabric on which it operates. The communications are secure, reliable and tamper-proof to the highest degree of authentication. Only the person to whom the message is addressed can access it and respond to it.
- **Performance:** The application must be quick to register transactions on the Ethereum network, and must be able to scale accordingly for any number of users.
- **Availability:** The application must be available to securely send and receive individual, group messages as well as broadcasts from other users, irrespective of the time zones its being accessed in, the place of residence, as well as the other processes running on his/her system.
- **Usability:** It is of utmost importance that the user be able to access the features of this application with ease, and with no prior knowledge about the specifics of how the application works. The user must have beginner-level access to every functionality this application aims to deliver.

Chapter 4

SYSTEM DESIGN

4.1 Introduction

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. The purpose of the System Design process is to provide sufficient detailed data and information about the system and its system elements to enable the implementation consistent with architectural entities as defined in models and views of the system architecture.

4.1.1 Structure of Blockchain

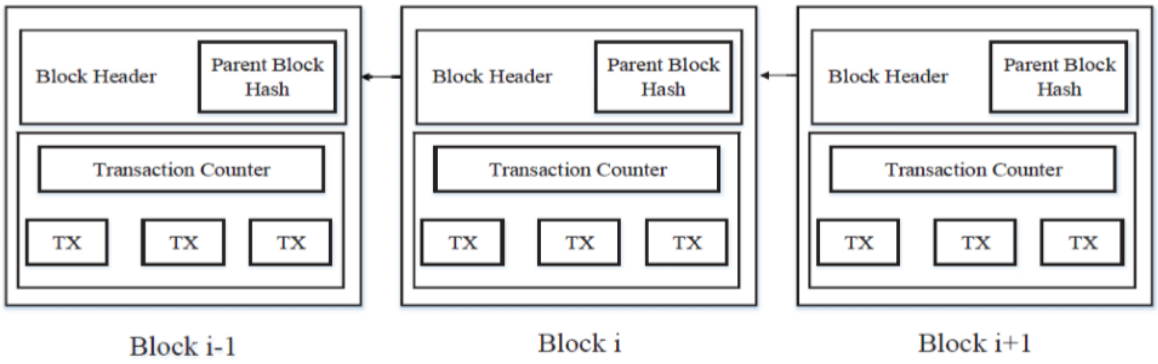

Figure 4.1 Continous sequence of blocks in blockchain structure

The four main components of any blockchain ecosystem are as follows:

1. a node application
2. a shared ledger
3. a consensus algorithm
4. a virtual machine

- Node Application: As seen in Figure 4.1, each Internet-connected computer needs to install and run a computer application specific to the ecosystem they wish to participate in. Using the case of Bitcoin as an example ecosystem, each computer must be running the Bitcoin wallet application.

- Shared Ledger: This is a logical component. The distributed ledger is a data structure managed inside the node application. Once the node application gets up and running, the respective ledger (or blockchain) contents can be viewed for that ecosystem.
- Consensus Algorithm: This, too, is a logical component of the ecosystem. The consensus algorithm is implemented as part of the node application, providing the rules of the game for how the ecosystem will arrive at a single view of the ledger.
- Virtual Machine: The virtual machine is the final logical component implemented as part of the node application that every participant in the ecosystem runs.

4.1.2 Structure of a block

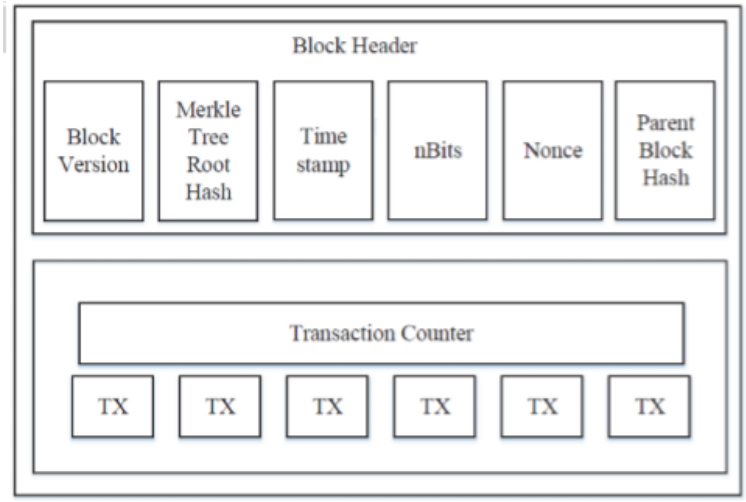

Figure 4.2 Block Structure

The block body is composed of a transaction counter and transactions, as described in Figure 4.2. The maximum number of transactions that a block can contain depends on the block size and the size of each transaction. Blockchain uses an asymmetric cryptography mechanism to validate the authentication of transactions. Digital signature based on asymmetric cryptography is used in an untrustworthy environment. Next, the digital signature is briefly illustrated.

A block consists of the block header and the block body. In particular, the block header includes:

i. **Block version:** indicates which set of block validation rules to follow.

ii. **Merkle tree root hash:** the hash value of all the transactions in the block.

iii. **Timestamp:** current time as seconds in universal time since January 1, 1970.

iv. **nBits:** target threshold of a valid block hash.

v. **Nonce:** a 4-byte field; usually starts with 0, increases for every hash calculation.

vi. **Parent block hash:** a 256-bit hash values that points to the previous block.

4.1.3 Ethereum Stack

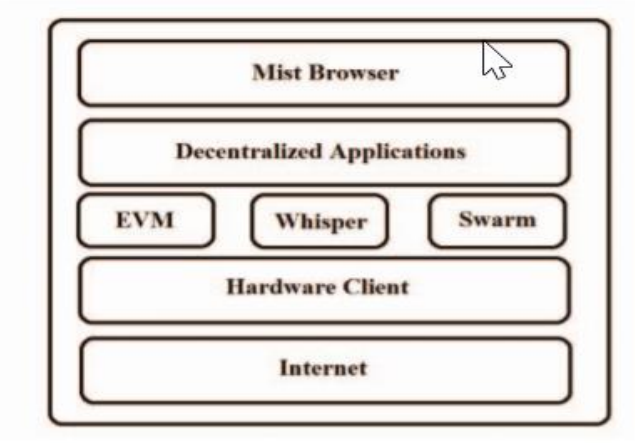

Figure 4.3 Ethereum technology stack

The Ethereum technology stack is given in Figure 4.3:

- Mist Browser: An interface to access various dApps. MetaMask plugin can also be used for Chrome browser support.
- Decentralized Applications: Decentralized Application consists of multiple nodes connected to each other in a mesh topology type network. They are connected to each other in a P2P fashion.
- Ethereum Virtual Machine (EVM): The EVM is an abstraction layer which 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.
- Whisper: It is Ethereum's P2P communication protocol for decentralized applications. P2P Communication between nodes in the Whisper network utilizes

the ĐΞVp2p Wire Protocol. A decentralized application instance can create an identity within a node that is connected to Whisper. All messages on Whisper are initially encrypted and sent through a basic wire-protocol called ĐEVp2p which supports various sub-protocols such as Whisper. The message is further encrypted by the ĐEVp2p protocol. Currently, all Whisper messages are required to be either asymmetrically encrypted using the Elliptic Curve Integrated Encryption Scheme (ECIES) together with the SECP-256k1 public key or symmetrically encrypted using the Advanced Encryption Standard Galios/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.

- Swarm: a distributed platform for storage and a content distribution service.

4.2 System Architecture

In this section, the proposed system is described that uses a client-side application-Ganache, an Ethereum client, to run as a node and to serve as an interface to interact with

the Truffle framework. The front end consists of a web application built on Node.js and was chosen due to the abundant support for web3.js, the Ethereum JavaScript API. The front end UI is rendered using React.js. In the system design, it is proposed that a smart contract owner/developer maintains the integrity of the privatised blockchain network, also validating and executing the broadcasted transaction request.

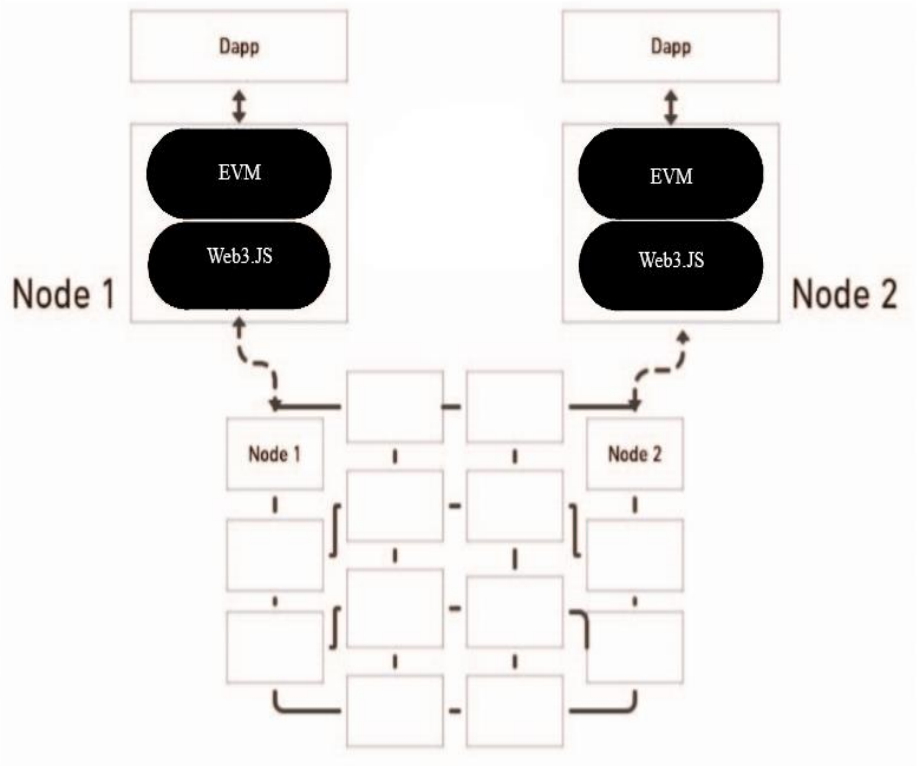

Figure 4.4 System Architecture

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 Ethereum, 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 Ganache client that serves as an Ethereum node, it can communicate with other nodes that are its peers on the Whisper Network

using the ‘messaging function’. The ‘messaging function’ utilizes the Ethereum JavaScript API web3.js library and is capable of transmitting signed encrypted messages to peers on the Whisper network. Since the Ganache instance handles Whisper identities it would not be in the user’s interest for the Ganache instance to be run by a third-party.

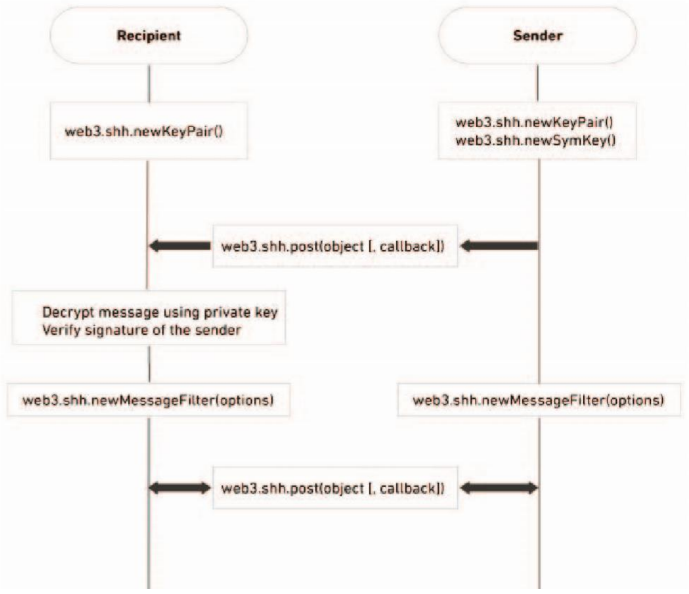

Figure 4.5 Message Transmission between two nodes

As shown in Figure 4.5, 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 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. In addition, message number is included in each outgoing message in order 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.

4.3 Detailed Design

A typical implementation of Ethereum blockchain network begins with setting up of nodes. The number of nodes depend upon number of organizations (separate entities) participating. Each organization has a certificate authority and each node of the organization may or may not have a copy of ledger depending on the type of node. Apart from all these, a specific set of ordering nodes has to be deployed to take care of communication between nodes. All deployments including certificate authority and databases for each node take place in separate docker containers.

Key Components:

Ledger	**Chain Code**	**Peer Network**	**Membership Service**
Events	**Wallet**	**Consensus**	**Systems Management**

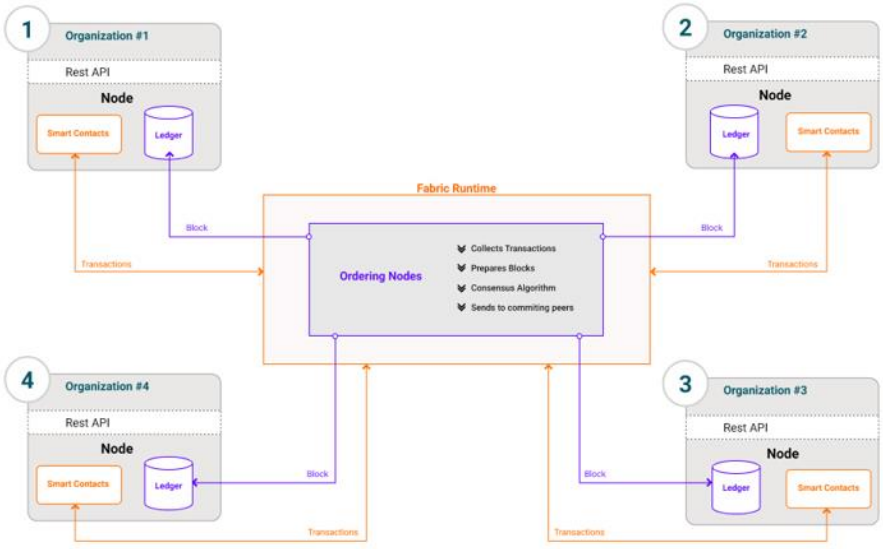

Figure 4.6 The nodes comprising blockchain fabric

The Three Fundamental Types of Nodes:

- **Committing Node**: Maintains ledger and state, commits transactions, and may hold smart contract (chain code).
- **Endorsing Node**: Specialized committing peer that receives a transaction proposal for endorsement, responds granting or denying endorsement. It must hold smart contract.
- **Ordering Nodes (service)**: Approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes. It does not hold smart contract or a ledger.

4.3.1 High-level Design

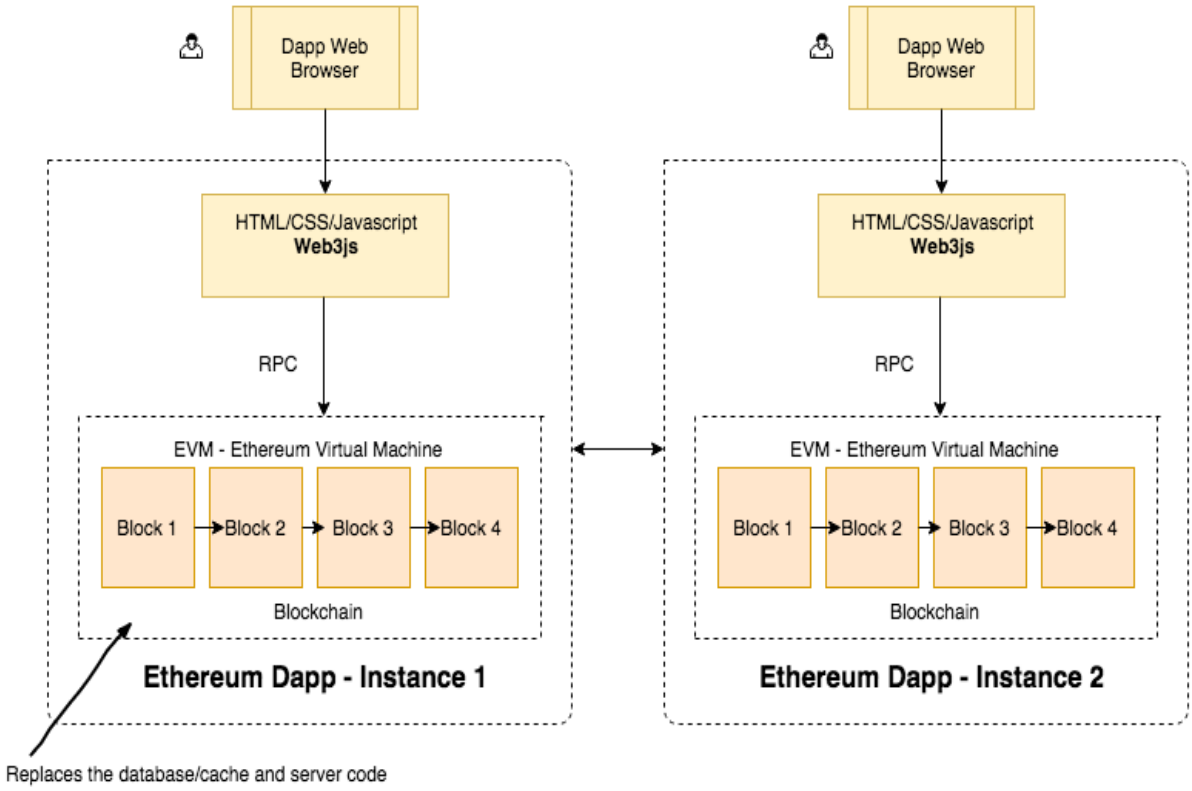

Figure 4.7 High-level dApp Architecture

Analogous to the JavaScript Engine, the Ethereum Virtual Machine (EVM) resides in every node and is responsible for executing the bytecodes. As the EVM runs, it maintains a stack of opcodes that can write to the contract's storage tree. This stack can have up to 1024 elements, and each element is 32 bytes. Thus, recursion is not recommended because there's a good chance it might cause a stack overflow. In essence, the EVM is simply a state machine, and the opcodes specify how state transitions are applied to the next block state.

The ABI is the developer's portal to the opcodes in the contract instance. Its function is to translate the bytecodes into JSON RPC calls to the network. To interact with the user's instance on the blockchain, we can use the ABI to instruct the EVM which functions and with which arguments we want to call. The most popular JavaScript library for this is web3, so named to remind us that we're building the future of Web 3.0.

Each web3 invocation is either a transaction (send) or a call. Transactions are sent to the network and potentially state-changing. Calls are read-only and fast. Contracts can call other contracts (aka, message), but it's always a transaction that gets things started. In

other words, message calls in themselves are never state-changing, but they can be part of a state-changing transaction.

With web3, four types of transactions can be submitted:

- Send ethers from one external account to another external account (like a Bitcoin transaction)
- Send a "blank" transaction to deploy a smart contract (becomes a contract account)
- Send ethers from an external account to a contract account
- Send a transaction to execute a method within the contract account (to update or retrieve the contract state, or call other contracts)

Transactions and calls are executed by the EVM as opcodes, which equate to gas. Transactions that update the blockchain or contract state require payment of ethers. These will take time and will always return the transaction hash. Calls that simply retrieve data from the blockchain or contract are "instant" and free. Any values that are required, can be returned from these calls.

4.3.2 Low-level Design

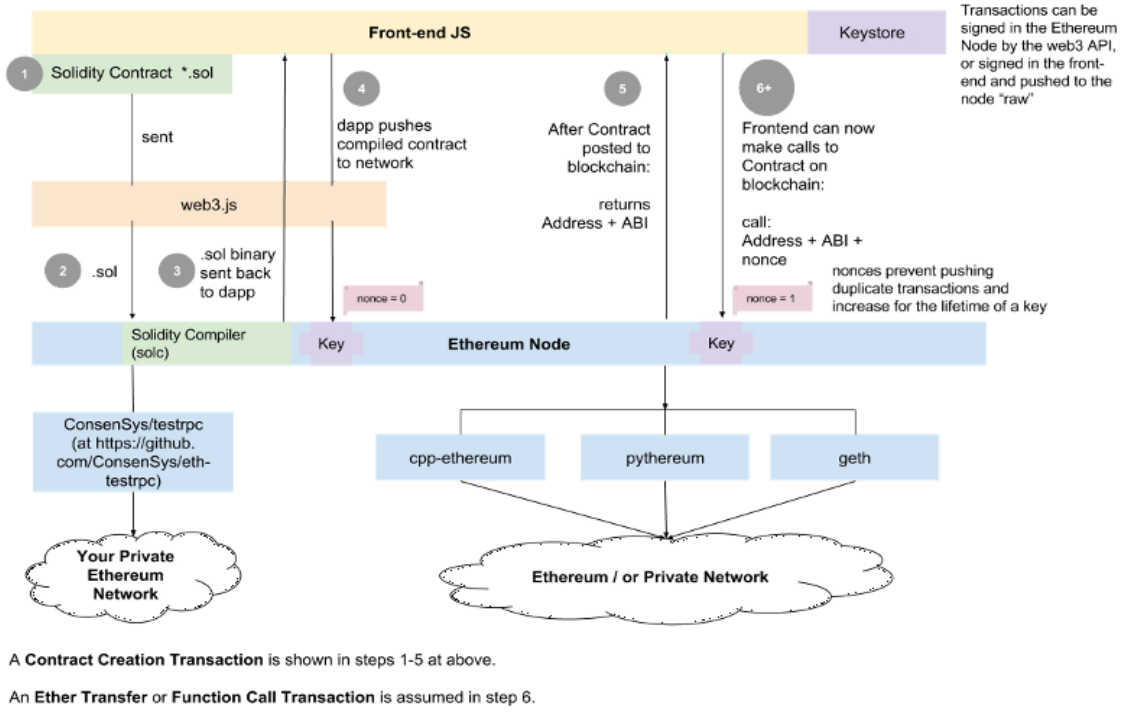

Figure 4.8 Low level architecture of a decentralized application

Performing operations or handling requests on data locally is the same as doing it remotely. The user and everyone else are the server and the client. This sounds more complicated than it actually is. IPFS is our decentralized storage solution of choice because it has gotten farther than any of the competitors in the space and synthesizes great ideas from years of research in the space with proven practices.

When dApps like these are built, it won't run on a server; rather, it will run locally on all the users' computers. Decentralized computation still hasn't been solved, and uploading the compute to a centralized virtual machine (VM) like Heroku would defeat the purpose of decentralization, so the right way to deploy a dApp is as a downloadable binary. Users can download it to their desktops and then access the dApp using either a web browser or directly within a client interface—for example, Spotify or Skype.

dApps will require data storage in some form or another and as such they will double as IPFS-distributed file storage nodes. An alternative would be to just use a third-party IPFS node on a server to store the data, but then that cloud provider would be a central point of failure. Inevitably someone is going to buy some Amazon EC2 space, host a node there, and offer IPFS-node-as-a-service to make it easier for beginners to get started with using it. The data would be replicated from there as people request files on a case-by-case basis. An IPFS cloud node would also be great for mobile dApps, given that running an IPFS node takes a good chunk of processing power, and that correlates to losing a good chunk of battery life for laptop users.

4.4 Data Flow Diagram

As described in Figure 4.9, when data is added to IPFS, it is essentially just broadcasting to the network that the user has the data; the user isn't actually sending it to someone's computer. That only happens when someone requests the data. And because the data lives on the network, manipulation is a result of commands to the network, as well.

IPNS (the naming layer on top of IPFS) gives the appearance that updating and deleting are possible through mutable names. With IPNS a DAG of data can be published under the immutable peer ID, and then whenever someone resolves this peer ID, that person can retrieve the DAG hash. IPNS can only store a single DAG entry per peerID, so if it is required to update or delete data, the user can just publish a new DAG to this peerID.

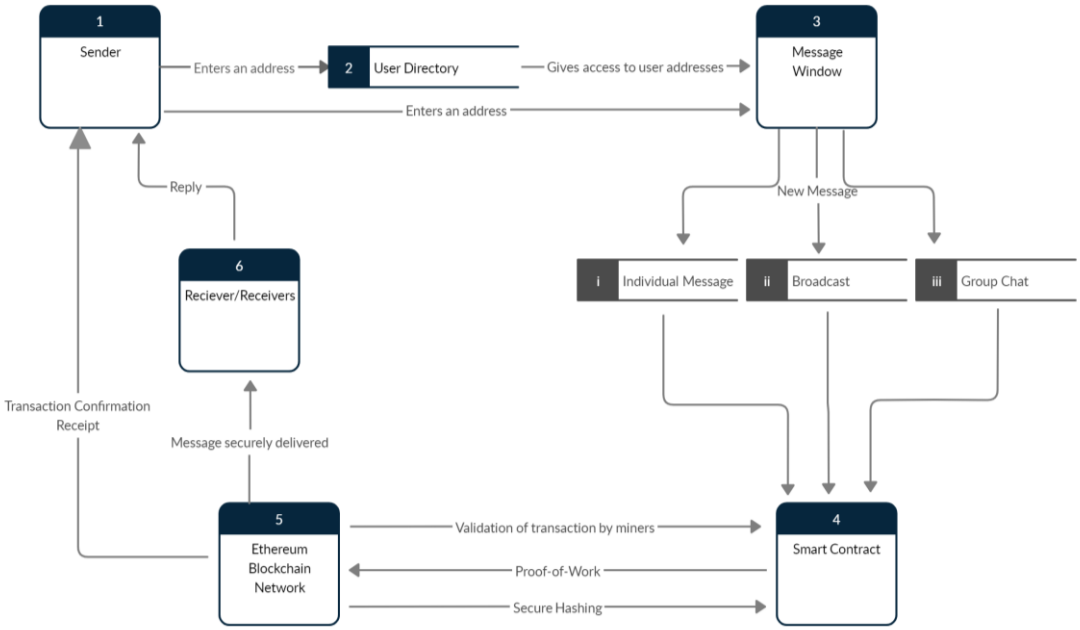

Figure 4.9 Data Flow Diagram for the proposed model

4.5 Use Case Diagram

As shown in Figure 4.10, the actor is single point of operation in this application. These actors have many viable features using which they can communicate to any other party that can operate the same application on their own setup. The only actor in the model is a user, which can be either a sender, or a receiver or both. The user can send a secure message, receive a message, reply to a message, create a broadcast where he can send the same message to multiple users at once, or participate in a group chat where all the participant users can communicate within themselves. The recipients of a message can also choose to reply to that message, view details of the message, and also clear their inboxes.

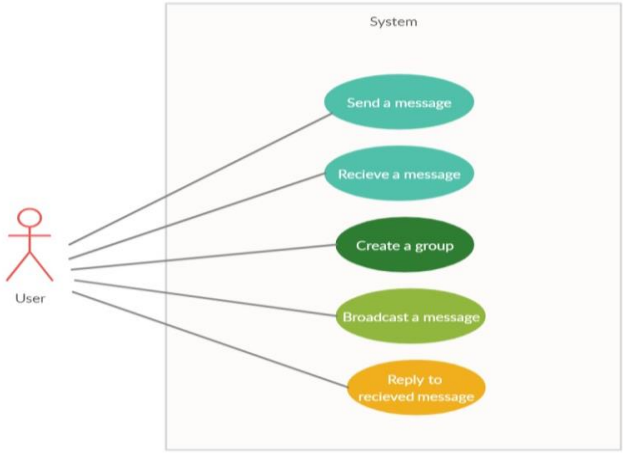

Figure 4.10 Use Case Diagram

4.6 Sequence Diagram

The smart contract periodically checks user status, traversing the state machines, transactions, and trigger conditions contained in each contract. The transactions satisfying the trigger condition is sent to the queue to be verified, waiting for consensus. Transactions that do not meet the trigger condition continue to be stored in the blockchain system.

The sequence diagram in figure 4.11 shows that the transaction entering the queue to be verified will be broadcasted to each verification node, and they will verify the transaction. The qualified transaction will enter the consensus set. When the consensus moment is reached, the transaction will be executed and notified to the user. When the transaction is executed successfully, the smart contract determines the status of the contract. When all the transactions included in the contract have been executed, the contract status would be marked as completed and removed from the latest block. Conversely, the contract is marked as in progress and continues to be saved in the latest block pending processing until the entire contract is processed. Transaction and state processing are automatically done by the smart contract system built in the bottom of the blockchain, which is transparent and cannot be tampered with.

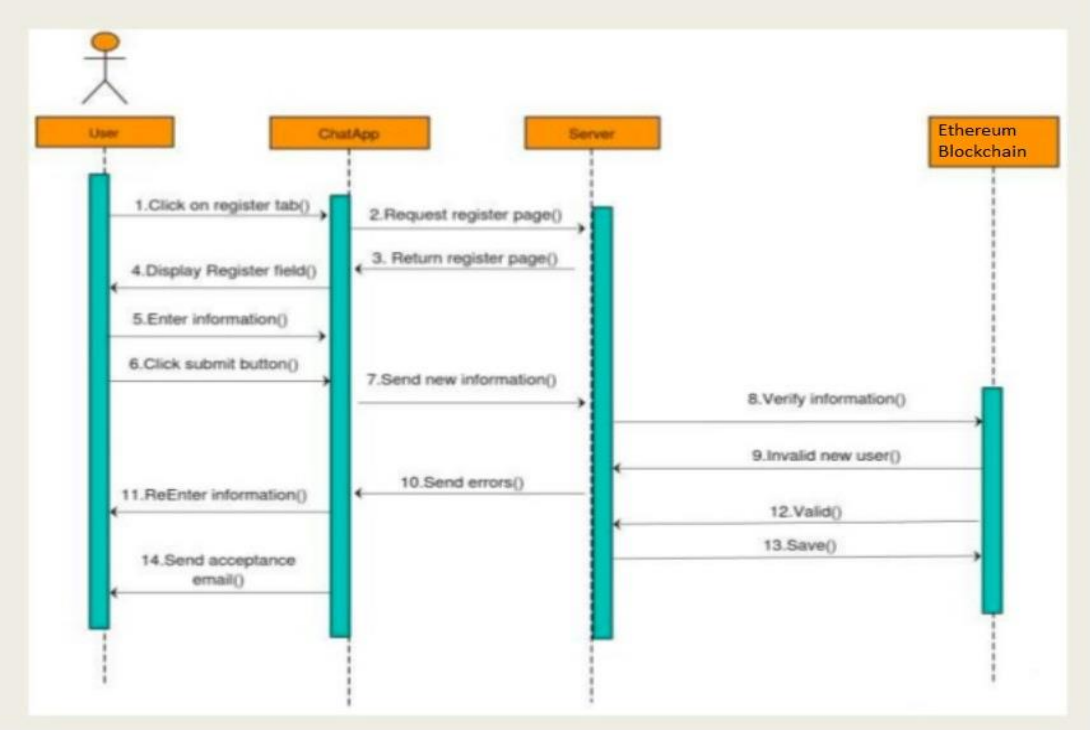

Figure 4.11 Sequence diagram

Chapter 5

IMPLEMENTATION

Software implementation begins with the effort of software fabrication. Fabrication is an act of making something. Software fabrication involves programmatic design, source code editing or programming, and testing of each software unit. This series of technical tasks represents how software procedures, routines, modules, objects, or graphical models are produced. Each software unit is presumed to be suitable for their intended purpose or role in the overall architectural context. The result of software fabrication should be a documented unit of source code that has been tested against its structural unit specification. This source code (software unit) is then available to be assembled, integrated, and compiled with other fabricated software elements to craft larger software components. These integrated software components are tested against structural component specifications to ensure their correctness. This assembly, integration, and testing series of events continue to generate larger, more complex software components. Software integration progresses until a completely integrated and tested software configuration item is realized and available for acceptance testing.

A special case occurs in object-oriented programming, when a concrete class implements an interface; in this case the concrete class is an implementation of the interface and it includes methods which are implementations of those methods specified by the interface.

5.1 Overview

The Blockchain technology enables direct transactions between two parties without any intermediary such as a bank or a governing body. It is essentially a database of all transactions happening in the network. The database is public and therefore, not owned by any one party, it is distributed that is, it is not stored on a single computer. Instead, it is stored on many computers across the world. The database is constantly synchronized to keep the transactions up to date and is secured by the art of cryptography making it hacker proof.

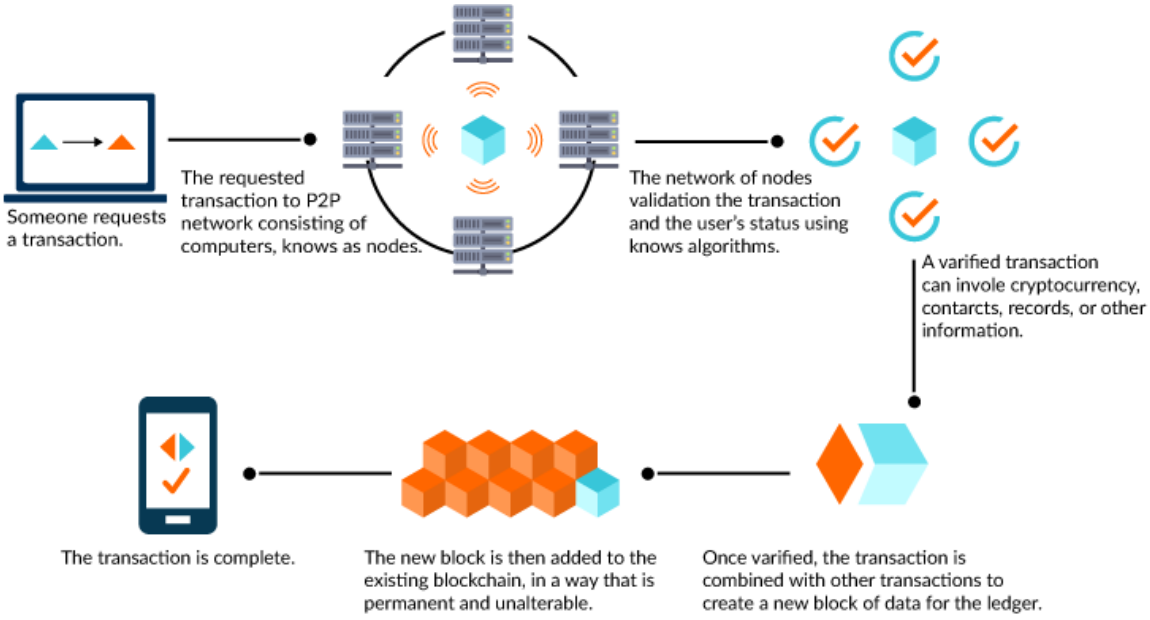

Figure 5.1 Overview of blockchain transaction

The basic framework on which the whole blockchain technology works is actually two-fold; first is gathering data (transaction records) and the second is putting these blocks together securely in a chain with the help of cryptography. Say a transaction happens, this transaction information is shared with everybody on the blockchain network. These transactions are individually timestamped and once these transactions are put together in a block, they have timestamped again as a whole block. Now, this complete block is appended to the chain in the blockchain network. Other participants might also be adding to the network at the same time, but the timestamps added to each block takes care of the order in which the blocks are appended to the network.

Each block's information is taken and a hash function is applied to it. The value computed from this is then stored in the next block and so on. So in this way, each block's hash function value is being carried by the next block in the chain which makes tampering with the contents of the block very difficult. Even if some changes are made to the block, one could easily find out because that block's hash value would not be the same as the already calculated value of the hash function that was stored in the next block of the chain.

Blockchain works as a network of computers. Ethereum photography is used to keep transactions secure and also shared among those in the network after the transaction

is validated, the details of the transfer are recorded on a public ledger that anyone on the network and sees in the existing financial system essentially ledger maintained by the institution access the custodian of the information. But on a blockchain the information is transparently held in a shared database and no one party access the movement, thus increasing the trust among parties.

In essence, a one-to-one, one-to-many and many-to-many messaging service is discussed, on a distributed network system in which no manager or administrator exists. A message is sent by a sender, then the receiver receives the message via an intermediary. The permission of send a message is the source of the incentive. This messaging is recorded in the blockchain to develop an autonomous distributed network system. Encapsulated information in a message is needed to chain a block to a blockchain. When the receiver receives a message, the intermediary obtains the confirmation of receipt. The reward is permission to send a new message. Therefore, relaying communications makes incentive.

5.1.1 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.

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. In addition, message number is included in each outgoing message in order 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.

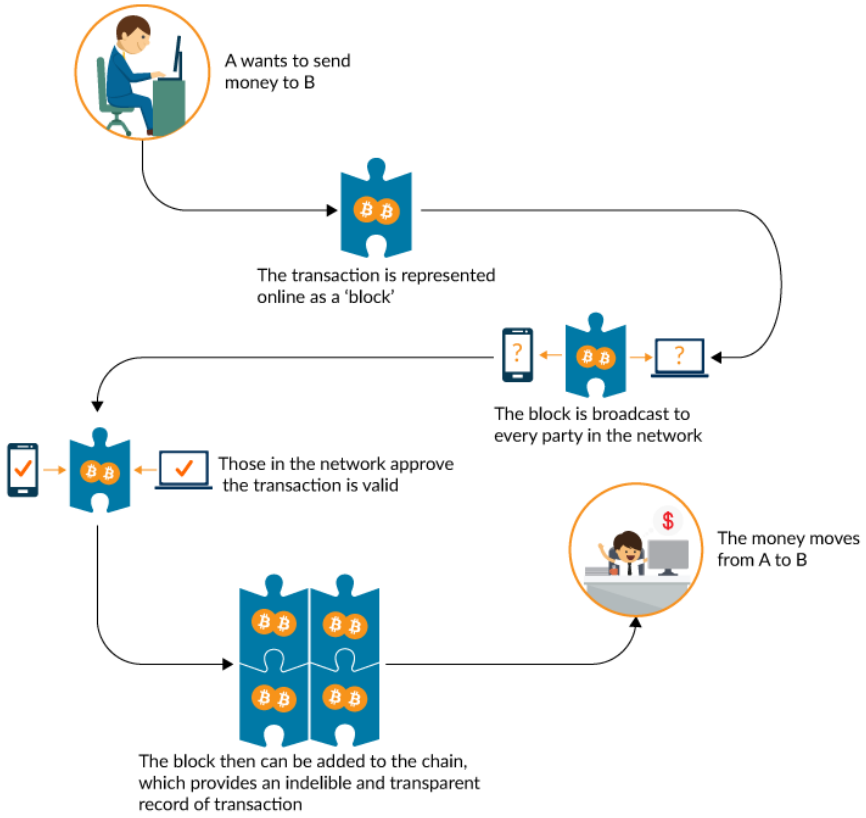

Figure 5.2 Simplified flow of transaction in blockchain network

The recipient can be certain of the identity of the sender since the symmetric key was signed by the sender using his private key. However, the messages cannot be used as conclusive evidence as they could have been sent by anyone with access to the symmetric key. This feature is responsible for the high degree of anonymity possible in blockchain communication.

5.2 Algorithms

An algorithm is a procedure or formula for solving a problem, based on conducting a sequence of specified actions. Algorithms are widely used throughout all areas of IT (information technology). A search engine algorithm, for example, takes search strings of keywords and operators as input, searches its associated database for relevant web pages, and returns results. Technically, computers use algorithms to list the detailed instructions for carrying out an operation. For example, to compute an employee's paycheck, the computer uses an algorithm. To accomplish this task, appropriate data must be entered into the system.

An encryption algorithm transforms data according to specified actions to protect it. A secret key algorithm such as the U.S. Department of Defense's Data Encryption Standard (DES), for example, uses the same key to encrypt and decrypt data. As long as the algorithm is sufficiently sophisticated, no one lacking the key can decrypt the data.

5.2.1 Proof-of-Work (PoW)

The purpose of a consensus mechanism is to bring all the nodes in agreement, that is, trust one another, in an environment where the nodes don't trust each other. Proof of Work consensus is the mechanism of choice for the majority of blockchain networks currently in circulation.

In Blockchain, this consensus algorithm is used to confirm transactions and produce new blocks to the chain. With PoW, miners compete against each other to complete transactions on the network and get rewarded. All the transactions in the new block are then validated and the new block is then added to the blockchain. Note that, the block will get added to the chain which has the longest block height. Miners, which are special computers on the network, perform computation work in solving a complex mathematical problem to add the block to the network, hence named, Proof-of-Work. With time, the mathematical problem becomes more complex.

Miners solve the puzzle, form the new block and confirm the transactions.

How complex a puzzle is depends on the number of users, the current power and the network load. The hash of each block contains the hash of the previous block, which increases security and prevents any block violation.

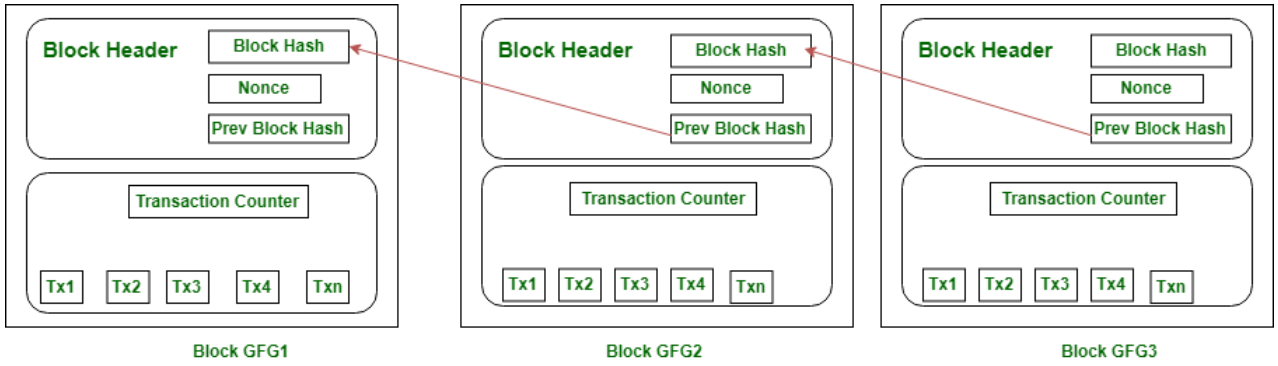

Figure 5.3 Proof-of-Work consensus algorithm

5.2.2 Secure Hashing Algorithm (SHA-256)

A cryptographic hash is a kind of 'signature' for a text or a data file. SHA-256 generates an almost-unique 256-bit (32-byte) signature for a text. **SHA-256** is a cryptographic hashing algorithm, which provides one of the strongest hash functions available.

- A hash is not 'encryption' – it cannot be decrypted back to the original text (it is a 'one-way' cryptographic function, and is a fixed size for any size of source text). This makes it suitable when it is appropriate to compare 'hashed' versions of texts, as opposed to decrypting the text to obtain the original version.

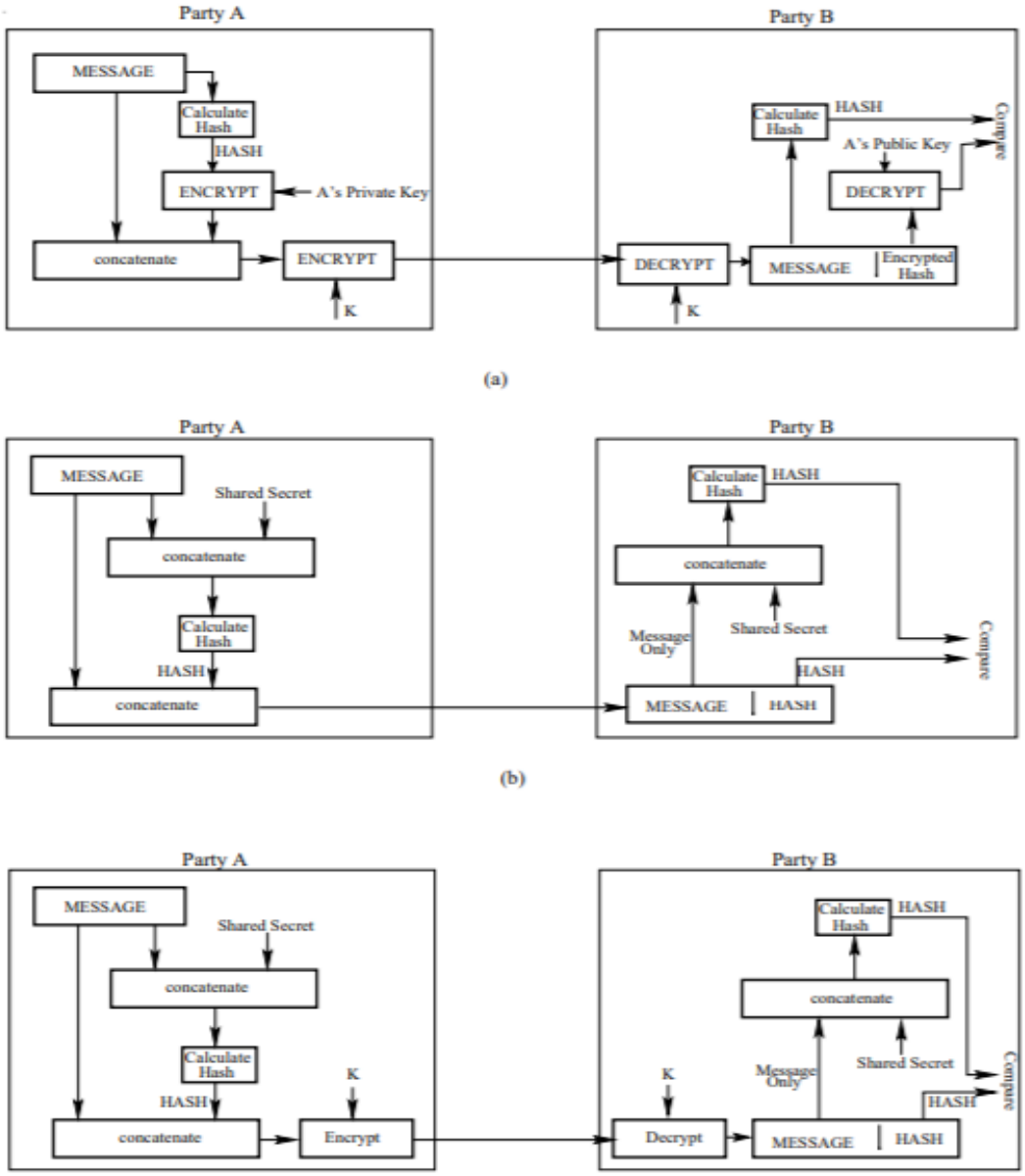

Figure 5.5 Different ways of incorporating hashing in P2P communication

5.2.3 Message Relaying Protocol

There are many nodes on a network, and one node can directly communicate with only some of the others. A node communicates with another node that cannot directly communicate via an intermediary. Unicast messaging, broadcasts as well as group messaging are considered. A message is sent by a sender, relayed by an intermediary, and received by a receiver constantly. A participating node is allowed to participate but cannot leave. The communication to synchronize blocks is called "broadcast". Broadcast is spread without incentive, and each node helps in the broadcast of communication. All blocks are shared by all nodes in the system by broadcast. However, detailed methods of broadcast have not considered. A nonce is replaced from Bitcoin with proof communication relay between participants. In this section, the recipient of a message conveys the value of confirmation to the intermediary.

Initial State

The initial state is from the first node creating the network to an existing relay node. The steady state is from an existing relay node. This is an example of begging the network to steady state. The first node: N1 creates the genesis block. When a node does not have permission to send a message, any node cannot send a message nor obtain, so a certain amount of permissions must be given unconditionally. Function **S(x)** defines the amount of giving permissions of sending a message for a participating node, where x is a natural order of participation. The terms n is a natural number and a is positive constant.

S(x) = a (if x ≤ n) or

$$\mathbf{S(x) = 0 \text{ (otherwise)}} \quad (1)$$

Two definitions are given for parameter n in Eq. (1), i.e., (i) n = ∞ and (ii) n is any natural number. The term ***a*** is the amount of receiving permissions. Permissions are received from the 1st participating node (N_1) to the nth participating node (N_n). For simplicity, (ii) is used in this section and defines **a = 1** and **n = 1**. Therefore, N_1 unconditionally obtains one permission to send one message. Then node N_2, which has direct connection to N_1, joins the network. Direct communications such as N_1 to N_2 messages, are not subject of considering of generating incentives. The joining the network of N_2 is completed when all blocks are shared by all nodes. Next, node N_3, which has direct connection to only N_2, i.e., it does not have direct connection to N_1, joins the

network. The N_2 sends N_3 all holding blocks. The joining of N_3 is completed when all blocks are shared by all nodes.

Joining node at steady state

While the system is in steady state, joining of a new node is completed when all blocks are shared by all nodes. In this model, the addition of an edge node between nodes already participating isn't considered, so once the network reaches a steady state, the network eternally satisfies the condition of a steady state.

Message sending process

While the system is in steady state, joining of a new node is completed when all blocks are shared by all nodes. The sender S generates a message M, then S sends M to a destinator D, the system transfers it. Specifically, the intermediary I relays M in the system. This protocol enables M to be relayed by I and received by D, through relaying, provides incentive for relaying a message, and prevents or detects various types of fraud.

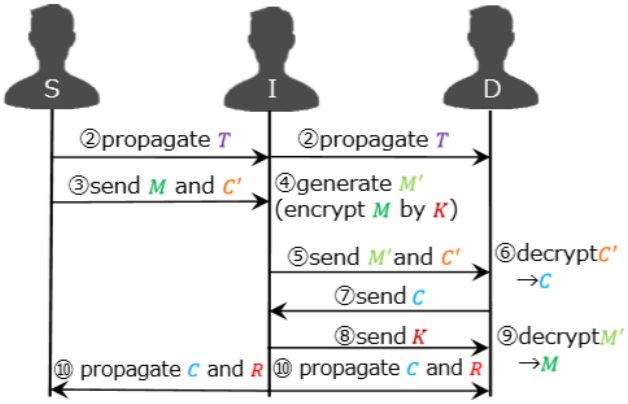

Figure 5.6 Sequence diagram of message relaying protocol

(i) The S writes M for D, generates a random number: C, generates C_0 by encrypting C with D's public key, and generates T which is the digest of M, sender and receiver information, and signature of the hash of C with S's private key.

(ii) S broadcasts T throughout the entire system.

(iii) S sends M and C_0 to I.

(iv) I generates K, generates M_0, and encrypts M by K which is generated by I.

(v) I sends M_0 and C_0 to D.

(vi) D decrypts C0 with D's private key and receives C.

(vii) D sends C to I.

(viii) I sends K to D.

(ix) D decrypts M_0, then D receives M.

(x) I generates R and broadcasts R and C, then I obtains a reward.

5.3 Pseudocode

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading. Pseudocode typically omits details that are essential for machine understanding of the algorithm, such as variable declarations, system-specific code and some subroutines. The programming language is augmented with natural language description details, where convenient, or with compact mathematical notation. The purpose of using pseudocode is that it is easier for people to understand than conventional programming language code, and that it is an efficient and environment-independent description of the key principles of an algorithm.

5.3.1 Registering user address

Input: Receiver's hash address

Output: Boolean value denoting success or failure.

Steps:

1. Read the address of receiver passed as parameter to the function.
2. If the receiver address is new, create a new instance of datatype **Inbox** for the passed address. Else skip to step 5.
3. Set the registered flag for this address to true.
4. Push the receiver address in the user directory so it is quickly accessible.
5. Copy this address in the receiver field of '**New Message'.**
6. Return success if new address was registered.

5.3.2 Sending Group message

Input: Array of addresses, message string

Output: Success/Failure

Steps:

1. Read the message string. If it is empty, then display failure and go to step 9.
2. Declare a variable NewMessage with attributes timestamp, content and sender.
3. Set the content of this variable as message string.
4. Set the timestamp to the current datetime.
5. Assign the sender to whichever user invokes the function.
6. Update the sender's inbox to reflect his own message.
7. For **n** number of receivers do:
    - Update n receivers' inbox with the new message.
    - Increment the number of received messages by 1.
8. Display success.
9. Return the control to invoking object and exit.

5.3.3 Receiving Group Message

Input: null

Output: message string, group name, timestamp, sender

Steps:

1. Create target arrays of **n** length for content, timestamp, groupname and sender, where n denotes number of recipients.
2. For **n** users do:
    - Initialize each content array element with received message string.
    - Initialize timestamp array elements with the datetime of reception.
    - Initialize all groupname array elements with name of group.

- Initialize sender array elements with address of the sender.

3. Return **n** tuples comprising of elements from each of 4 arrays; one tuple for one recipient.

4. Exit and return control to invoking object.

5.4 Implementation Support and Features

Along with the discussed modules in this proposed model, this ethereum messenger has the robustness to scale for any type of user bases. In this section, the various additional features are discussed. These are implemented to make this prototype stand out from the various decentralized applications that exist today. Also, the syntaxes and code snippets that bring along the implementation of these features are discussed.

5.4.1 User Directory

```
  $('#registeredGroups').html(s);
  var meta;
  App.contracts.BlockChat.deployed().then(function(instance) {
    meta = instance;
    return meta.checkUserRegistration.call({from: account});
  }).then(function(value) {
    if (value) {
      console.log(value);
      self.setStatus("User is registered...ready");
    } else {
      if (confirm("New user: we need to setup an inbox for you on the Ethereum blockchain.")) {
        App.registerUser();
      } else {
        return null;
      }
    }
  }).catch(function(e) {
    console.log(e);
    self.setStatus("Error checking user registration; see log");
  });
  return App.getMyInboxSize();
},
```

Figure 5.7 Implementing User Directory

This feature gives us a go-to mechanism to access all the recent addresses the user has communicated with. Whenever the user sends a new message or a broadcast, the system model checks whether the address and the user is already registered. If yes, it goes ahead and sends the message. Otherwise, it registers the address in the form of a new user in a data structure which is the User Directory. This implementation is somewhat similar in essence to the speed dial feature used in day-to-day lives. The user can quickly copy an address from the user directory to the contact field, making the communication a lot easier for anybody.

5.4.2 Reply to a Message

```
replyToMessage: function() {
  document.getElementById("message").focus();
  document.getElementById("message").select();
  sendsingle=replyToAddress;
  if(usermap.has(replyToAddress)) {
  document.getElementById("receiver").value = usermap.get(replyToAddress);
  }
  else {
    document.getElementById("receiver").value = "abcde";
  }
},
```

Figure 5.8 Replying feature

Essentially, when this feature is used, the contact field is automatically filled with the address of the person who sent the message earlier. The sender of the message becomes the recipient of the reply.

5.4.3 Copy Address to Send

```
copyAddressToSend: function() {
  var sel = document.getElementById("registeredUsersAddressMenu");
  var copyText = sel.options[sel.selectedIndex];
  sendsingle=copyText.innerHTML;
  console.log(copyText.innerHTML);
  console.log(usermap.has(copyText.outerHTML));
  if(usermap.has(copyText.innerHTML)) {
  document.getElementById("receiver").value = usermap.get(copyText.innerHTML);
  }
  else{
  document.getElementById("receiver").value = "abcde";
  }
  document.getElementById("message").focus();
  document.getElementById("message").select();
},
```

Figure 5.9 Copying Address feature

This feature can be used when the user desires to either send a message to a user from user directory, or wishes to add them to a group chat. In either ways, the user may copy the recipient's address to the contact field by selecting the Copy Address feature. It is notable that the application copies the selected address from the user directory, and pastes it in the contact address field, making the communication easier.

5.4.4 Clearing Inbox

```
},
clearInbox: function() {
  var self = this;
  var meta;
  this.setStatus("Clearing inbox:(open MetaMask->submit->wait)");
  App.contracts.BlockChat.deployed().then(function(instance) {
    meta = instance;
    return meta.clearInbox({}, {
      from: account,
      gas: 6385876,
      gasPrice: 20000000000
    });
  }).then(function(value) {
    var clearInboxButton = document.getElementById("clearInboxButton");
    clearInboxButton.parentNode.removeChild(clearInboxButton);
    $("#mytable tr").remove();
    document.getElementById("receivedTable").style.display = "none";
    alert("Your inbox was cleared");
    self.setStatus("Inbox cleared");
```

Figure 5.10 Clearing Inbox

Sometimes, when the user has received a large number of messages, he may wish to clear out the app inbox to refresh the application. In this case, the Clear Inbox option has been implemented, which enables the user to get rid of the messages he has already seen. From the point of time he clears his inbox, all the old messages will disappear and only the new messages from that point on will be displayed on the user's app window.

5.4.5 Creating a group

```
},
createGroup: function() {
  var self=this;
  var groupname=document.getElementById("groupname").value;
  sessionStorage.setItem(groupname, JSON.stringify(sendgroup));
  self.setStatus("group created:"+groupname);
  for (var i = 0; i < sessionStorage.length; i++){
    //validating if user belongs in that group
  var user=JSON.parse(sessionStorage.getItem(sessionStorage.key(i)));
  var user=user.replace(/,\s*$/, "");
  console.log("all users:"+user);
  var userarray=user.split(",");
  console.log("user array:"+userarray);
  console.log("isarray:"+Array.isArray(userarray));
  var curruseraddress=currentUser;
  console.log("curruseraddress:"+curruseraddress)
  console.log("real test:"+userarray.includes(curruseraddress));
  var select='';
  if(userarray.includes(curruseraddress)) {
    select += '<option val=' + i + '>' +sessionStorage.key(i)+ '</option>';
    }
  }
  $('#registeredGroups').html(select);
}

};
```

Figure 5.11 Creating a chatroom for group chat

This is one of the most essential features that makes the proposed model applicable and usable to many real-world scenarios where privacy is paramount. If a group of users wish to communicate with each other and with the admissible feature of blockchain security within the chatroom, they may do so with this group chat feature. The user may either manually add participants to create a new group, or can copy any number of addresses from the user directory. The user also has the option to assign a name to this group within which the chatroom will be hosted. The array of addresses is passed to the smart contract, which then initiates a secure transmission between these users for a specific session in which these messages are stored and only the pariticipants can view these messages with a shared key.

Chapter 6

TESTING

Software testing is conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. It involves the execution of a software component or system component to evaluate one or more properties of interest (a program or application), with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.

6.1 Introduction

A system based on machine learning is primarily been associated with building models that could do numerical or class-related predictions. This is unlike conventional software development, which is associated with both development and "testing" the software. The usage of the word "testing" in relation to decentralized application models is primarily used for testing the model performance in terms of accuracy/precision of the model. It can be noted that the word, "testing" means different for conventional software development and dApp model development.

Decentalized application models would also need to be tested as conventional software development from the quality assurance perspective. Techniques such as Blackbox and white box testing would, thus, apply to Machine Learning models as well for performing quality control checks on Machine Learning models.

6.1.1 Blackbox testing

Blackbox testing is testing the functionality of an application without knowing the details of its implementation including internal program structure, data structures, etc. Test cases for Blackbox testing are created based on the requirement specifications. Therefore, it is also called as specification-based testing. The following figure 6.1 represents the Blackbox testing. This method attempts to find errors in the form of incorrect or missing functions, interface errors, errors in data structures or external database access, behavior or performance errors, and initialization and termination errors.

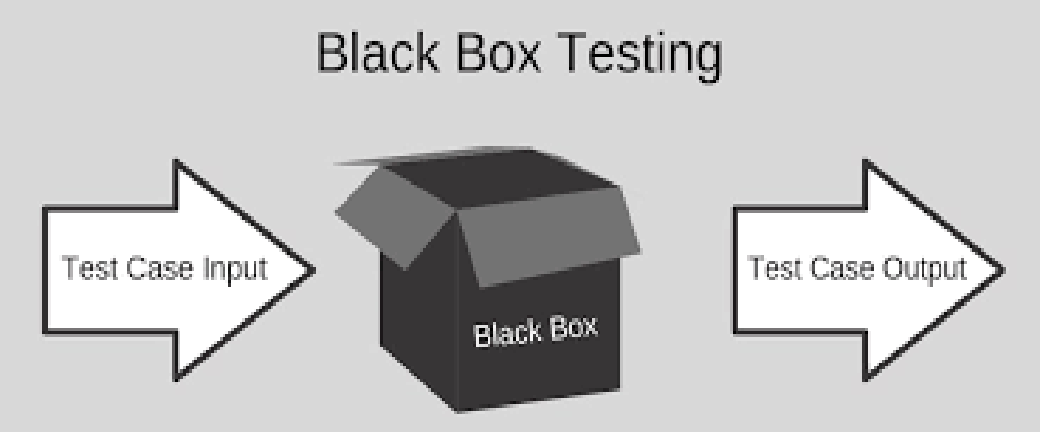

Figure 6.1 Black box testing

6.2 Unit Testing

The purpose of unit testing is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

In this decentralized application, unit testing was enabled for different modules of the system model, to ensure that the message is transmitted to a valid receiver, and that any number of participants can participate in a group chat. Along with that, the addresses for message transmission were validated with a variety of test input classes and the results were classified accordingly.

Table 6.1 Unit Testing

Test Case	Description	Expected Output	Actual Output	Result
1	Check whether all recipients in the group gets the message	Message received in all inboxes	Message received in all inboxes	Pass
2	Check the validity of user sending the message	Message sent from valid user	Message sent from valid user	Pass
3	Check whether message costs less than the ether balance	Cost in ether deducted from balance	Cost in ether deducted from balance	Pass

6.3 Integration Testing

The various modules of the prototype were binded together and the smart contract was written that governs any kind of transmission that occurs from this application to the ethereum network. The different units and features were integrated to work together with the help of the UI. The system was then tested as a whole to check whether the transmission of message from one user to another occurs efficiently, with test addresses and test ether currency as inputs. This application was found to have successfully hold the communication between multiple cases of user groups.

Table 6.2 Integration Testing

Test Case	Description	Expected Output	Actual Output	Result
1	The details of transaction when a message is sent	Cost in ethers is displayed	Cost in ethers is displayed	Pass
2	Messaging to multiple parties simultaneously	The same message is received	All the inboxes receive identical messages	Pass
3	Status of sending a message	Displays waiting while in process and success after confirmation	While transaction is pending, displays Waiting and after transaction confirmed, shows success	Pass

6.4 System Testing

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also

intended to test up to and beyond the bounds defined in the software or hardware requirements specification.

In this system model, the various modes of message transmission between users is tested along with the robustness of this application. It is observed that the speeds of message transmission between different group of peers remain the same, i.e the speed depends on how fast Proof-of-Work algorithm validates the transactions.

Table 6.3 System Testing

Test Case	**Description**	**Expected Output**	**Actual Output**	**Result**
1	One-to-one messaging (Individual chat)	Message received and ether deducted from sender	Costs the sender ether equal to one transaction	Pass
2	One-to-Many messaging (Broadcast)	Message received by all and **n**-times ether deducted for **n** users	Costs the sender ether equal to **n** times the cost of one txn	Pass
3	Many-to-Many messaging (Group Chat)	Message received by **n** users and **n**-times ether deducted for every message	Costs any sender ether equal to **n**-times cost of txn for every message sent	Pass

The proposed system supports asynchronous request, which means the client can send a batch of requests and retrieve the replies together when they are ready. Compared with synchronous pattern, the mean latency of asynchronous request is almost the same while the throughput can achieve a good improvement if the broker has enough workers.

6.5 Validation Testing

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements. Validation Testing ensures that the product actually meets the client's needs. It can also be

defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

In this application, the user requirements are modelled by assigning the top priority to the security feature while communicating with other peers on the ethereum network. The critical scenarios of application being validated revolve around how effectively the user can send and receive messages individually and in a group.

Table 6.4 Validation Testing

Test Case	Description	Expected Output	Actual Output	Result
1	Messaging with anonymity and secure communication	Message received and viewed by intended receiver only	Inbox of receiver is populated with message and details of message	Pass
2	Replying to a message	Receiver can reply to the sender with one click	Reply feature works in messaging as well as group chat option	Pass

One important finding is that the chain verification process is executed faster and consumes less energy when transactions are grouped together in a block. The cryptography function can be separated into three parts, i.e., cryptography-hashes, key-signature, and encode-decode. In the cryptography-hashes, the SHA3-256 algorithm can be used. In particular, as shown in Figure 6.4, when there are 3 or 6 transactions grouped in one block, the total execution time per block can be reduced approximately 30% or 40%, respectively. When the execution time is reduced, the energy consumption will be decreased accordingly. However, in practice, having more number of transactions in a block can cause more delay if the transactions are generated randomly. Clearly, as the number of blocks increases, the execution time rises. However, interestingly, when the number of threads is increased, the execution time is not always reduced. Specifically,

when the number of threads is increased from 1 to 2 or from 2 to 4, the execution time is reduced approximately twice. However, if the number of threads keeps increasing, the execution time reduces insignificantly.

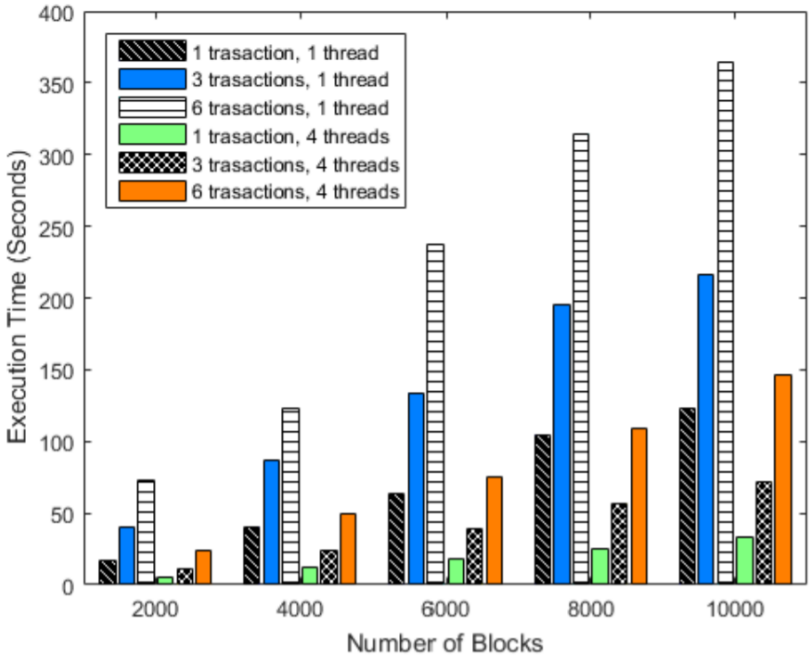

Figure 6.2 Execution Time for verification of transaction

6.6 User Acceptance Testing

User Acceptance Testing (UAT), also known as beta or end-user testing, is defined as testing the software by the user or client to determine whether it can be accepted or not. This is the final testing performed once the functional, system and regression testing are completed.

The main purpose of this testing is to validate the software against the business requirements. **Alpha** and **Beta** testing are different types of acceptance testing. This is typically the last step before the product goes live or before the delivery of the product is accepted. This is performed after the product itself is thoroughly tested (i.e after system testing).

In this case, the user tests all the given functionalities of the application to find out the glitches, latency as well as the efficiency while sending and receiving single and batch messages as well as broadcasts from other users. The intention is to create a fault-proof system designed for the everyday user where he can communicate surveillance-free, safely and anonymously in a trustless environment.

Table 6.5 Acceptance Testing

Test Case	Description	Expected Output	Actual Output	Result
1	Adding a user to user directory	Address added to directory	Any new address read is saved into user directory	Pass
2	Copying address from user directory	Selected address is copied using Copy button	Selected address goes to Contact field on clicking Copy button	Pass
3	Creating and communicating with a group	Chat group is created with any number of users	A group of selected users is created with a group name, in which everyone can send and receive messages	Pass
4	Broadcasting a message	A message is sent to all selected users	The same message is received in all the users' inboxes	Pass
5	Clearing inbox	All the received messages are cleared from inbox	All previous messages disappear and inbox is now empty	Pass

Chapter 7

DISCUSSION OF RESULTS

In this chapter, the final working aspects of this decentralized application are discussed. An attempt is made to apprehend how these functionalities can be best implemented. It is worth noting that this is just a prototype model that has been proposed in this project, to elaborate on what functionalities can be applied to a blockchain messenger. This cannot be viewed as a commercial or marketable product, because it is not industry-ready. Despite the development model being a prototype, it is attempted to achieve the most usability a user can get with this messenger application. The results are discussed briefly in this chapter along with snapshots.

7.1 Sending a message

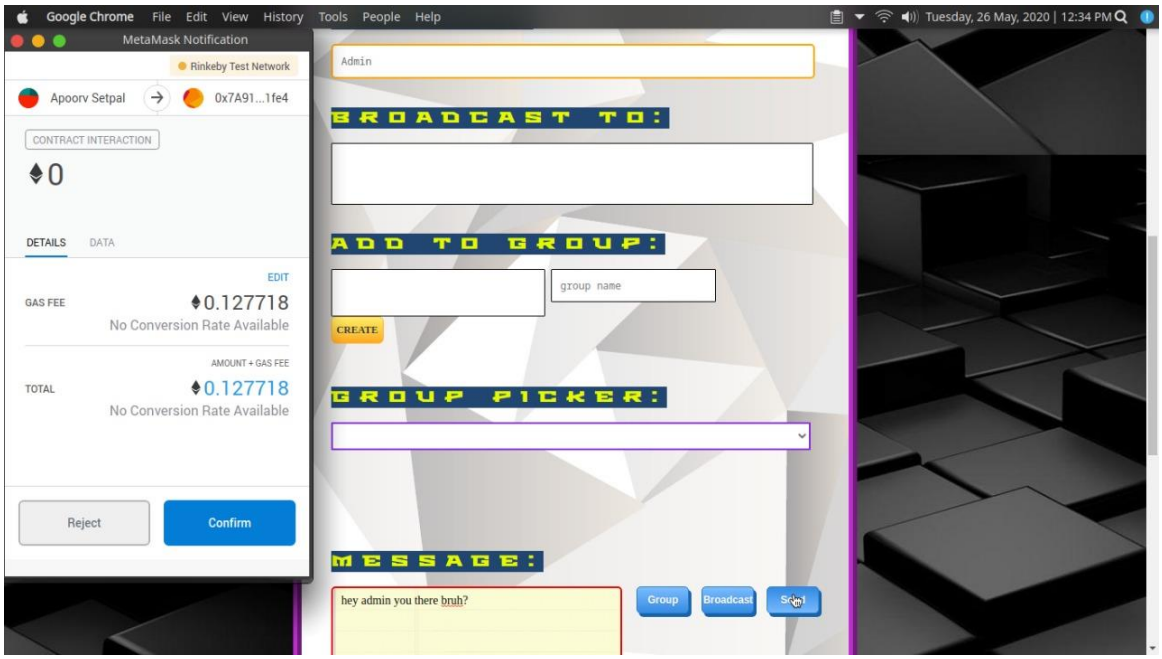

Figure 7.1 Sending a message

The user sends a message to another user named “Admin” by typing message string in message box and clicking Send button. It can be noted here that a MetaMask window pops up that shows the Gas Fee and ether cost for the transaction (every sent message on blockchain is a transaction). Upon confirming, the transaction is verified and message status is displayed as Successfully sent.

7.2 Receiving a message

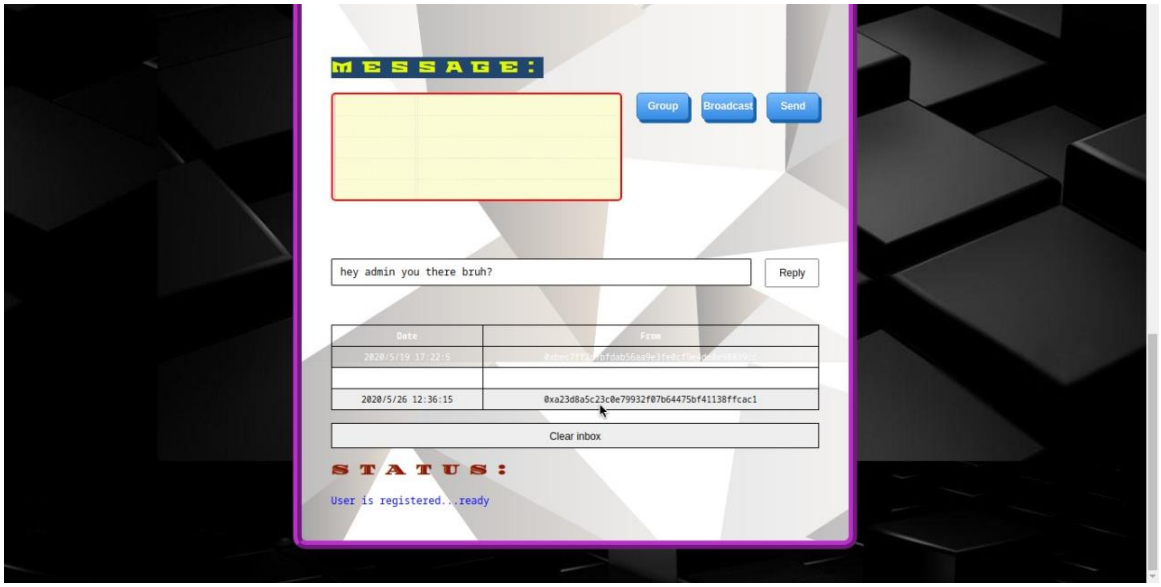

Figure 7.2 Receiving a message

As shown in Figure 7.2, the recipient's inbox is populated with the timestamp and hash address on the received message, while the message string is displayed on a separate field. Right next to the message, the user have a reply button so that the recipient can instantly respond to the message and reply to the sender.

7.3 Replying to a message

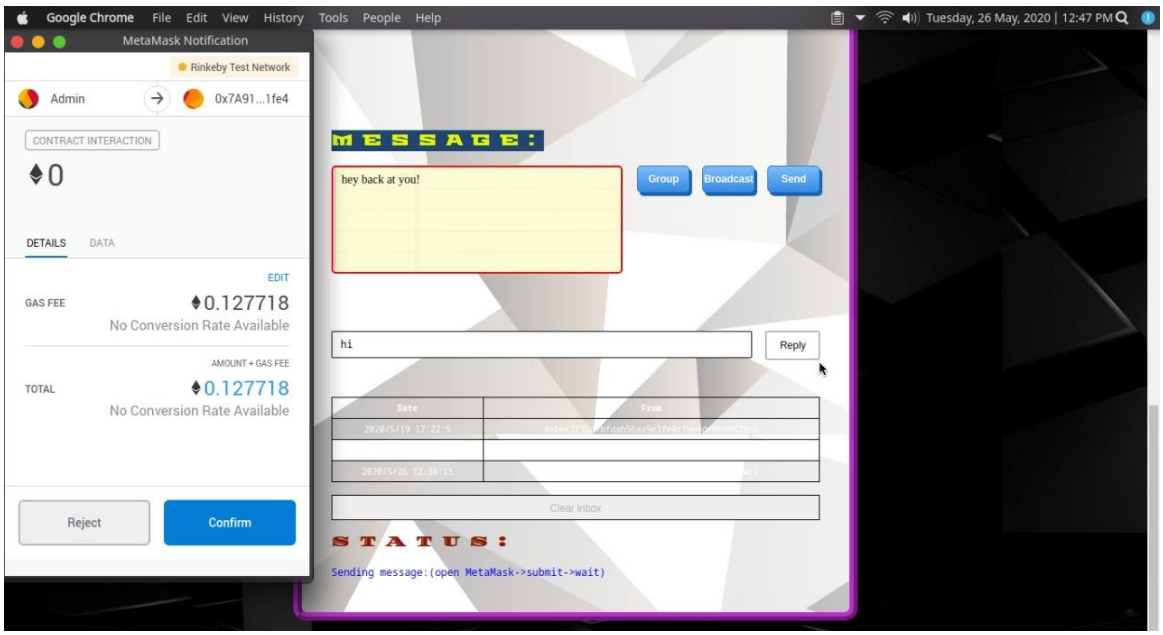

Figure 7.3 Replying to a message

When a user receives a message, he views it on the inbox and next to it is the reply option. On clicking that button, the contact field is automatically set to the address of the initial sender. The user just has to type a reply and hit the Send button, upon which the reply will be communicated back to the sender. The message gets sent only when the user confirms the transaction on MetaMask popup window, which represents an official transaction on the Ethereum blockchain.

7.4 Broadcasting a message

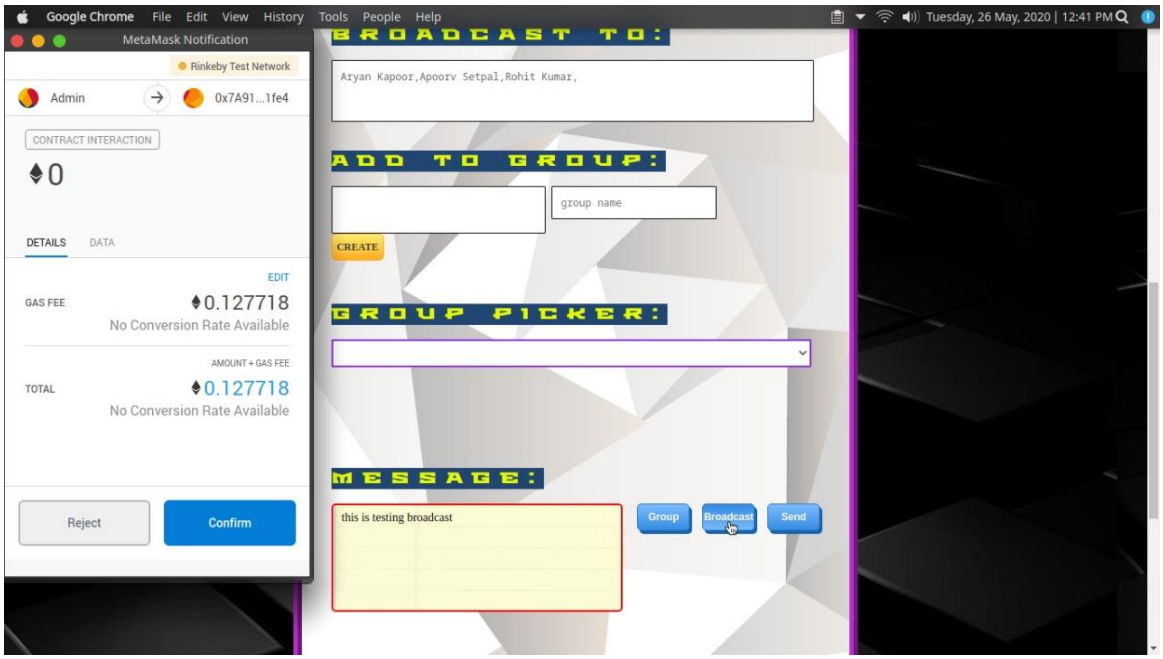

Figure 7.4 Broadcasting a message

If a user wishes to send a single message to multiple users at once, he can do so with this application by using the Broadcast Feature. In the above snapshot, it can be seen that the user is sending a sample message to three users. Upon clicking the Broadcast button, the same message will be identically transmitted to all the three users on their respective addresses simultaneously. The user can also add any number of recipients in a broadcast message with no difference in latency.

It can be noted here that the gas fee for one transaction always remains the same, no matter how many addresses the message is being broadcasted to. That is because one transaction takes up only one thread on the blockchain, which means the execution power required to validate this transaction is the same as validating a transaction with single recipient.

7.5 Group Messaging

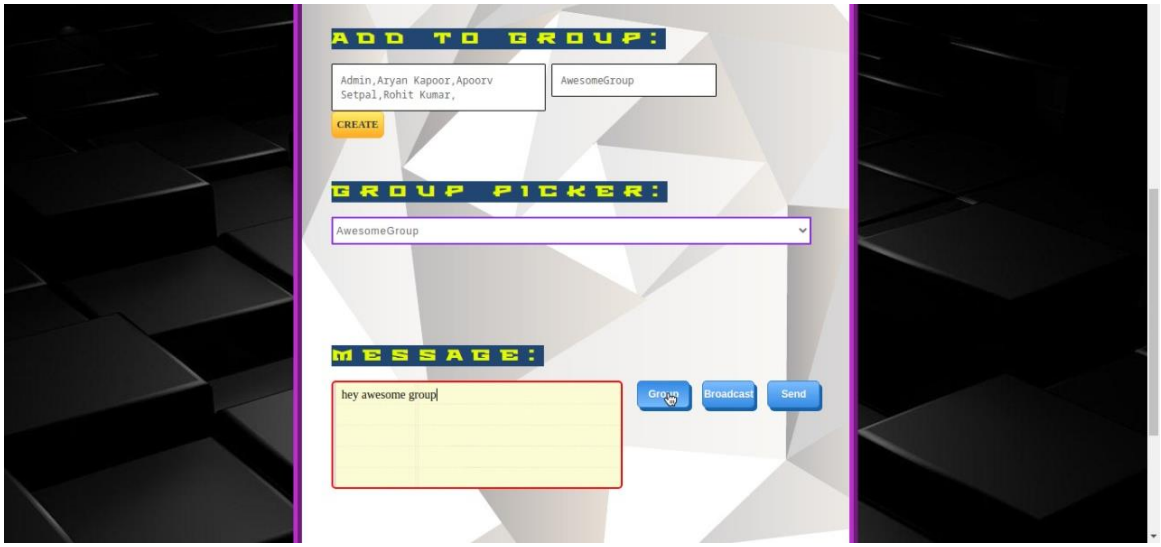

Figure 7.5 Messaging on a group chat

Here, the user adds the desired number of participants to a chat room, as here he has added four of them. Along with that, he may give a name to the group chat so that it is easily identifiable. With the help of group picker, the user can choose from any of the groups he has created before. This acts like a speed-dial feature in the application, providing ease of use for a user. After picking a group or creating one manually, the user types the message in message field and clicks on Group button. Upon confirmation of transaction, the message is received in every participant's inbox along with the name of the group.

7.6 Clearing the inbox

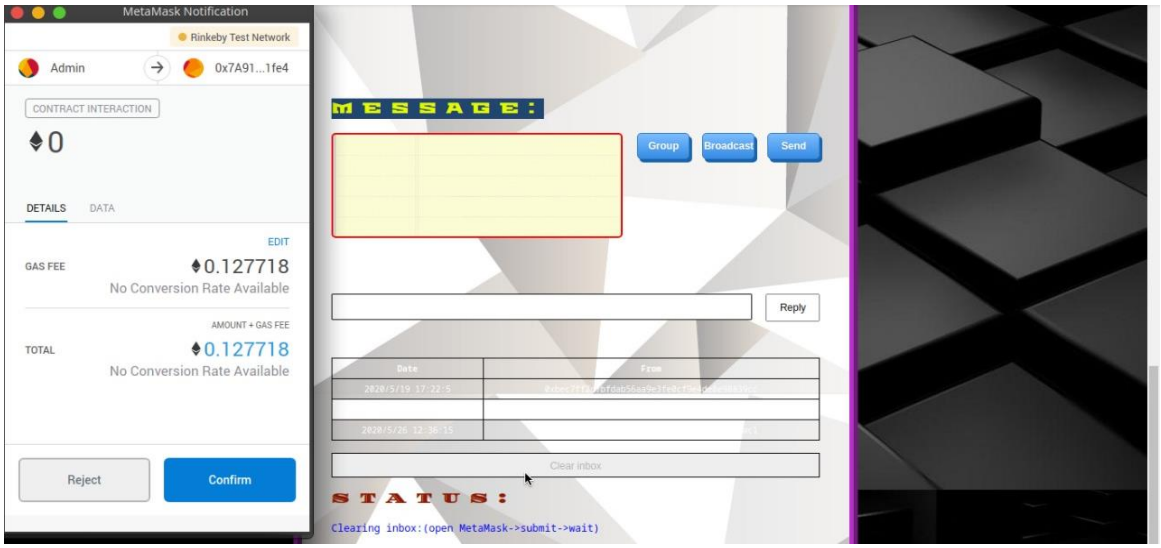

Figure 7.6 Clearing user's inbox

When the user receives a large amount of messages and his inbox is already populated, he may choose to clear his inbox by clicking Clear Inbox button. All the previous messages automatically disappear on clicking that button. This frees the cache and refreshes the user's chat window.

7.7 Cost Analysis

- **Token**: A token is a unit value that exists on an existing blockchain. As a token has a unit value, that value is tradable. Example- Ethereum, Bitcoin etc.
- **Ether**: Like Bitcoin, ether is a digital bearer asset ("token"). It provides "fuel" for the decentralized apps on the network.
- **Gas**: Gas is a unit of cost for a particular operation a computer needs to execute, and it executes this instruction when the user broadcasts a transaction which contains an Ethereum program in order to run a *dApp* (decentralized application).

Significance of Gas over Ether

All the gas prices of all the possible operations the EVM can perform are hard-coded in the Ethereum protocol. If the code listed them in ether, then the code has to be updated every time ether's value fluctuated.

By adding this *gas* layer on top of the costs, and paying for gas with GWei, the user is given the option to alter the amount of gas to use in a transaction and the amount of money to pay for it. It's fully under the control of user, without throwing the system off balance.

- **Gas Limit**: Gas limit is the *maximum amount* of gas the user is willing to spend on a transaction. Most softwares used to broadcast Ethereum transactions has the ability to auto-estimate the amount of gas that'll be necessary to execute a function.
- **Gas Cost**: The spent amount of gas is called gas cost.

Cost In Ether: It is calculated as:

*Cost In Ether=Actual Gas Cost*Gas Price*

Where, GAS PRICE=WEI cost per unit gas.

This fluctuates according to market value. **(1 ether=10^18 Wei)**

Task	Gas Required	Cost (ETH)	Cost (USD)	Ops per ETH	Ops per USD	Ops per Block	Blocks to complete Op
Add or subtract two integers	3	9.00E-08	2.66E-05	111111111.11	37664.78343	1566666.667	6.38E-07
Add or subtract two integers 1 million times	3000000	0.09	26.55	11.11111111	0.03766478343	1.566666667	0.6382978723
Multiply or divide two integers	5	1.50E-07	4.43E-05	6666666.667	22598.87006	940000	1.06E-06
Multiply or divide two integers 1 million times	5000000	0.15	44.25	6.666666667	0.02259887006	0.94	1.063829787
Compare two integers	3	9.00E-08	2.66E-05	111111111.11	37664.78343	1566666.667	6.38E-07
Compare two integers 1 million times	3000000	0.09	26.55	11.11111111	0.03766478343	1.566666667	0.6382978723
Create a new contract	32000	0.00096	0.2832	1041.666667	3.531073446	146.875	0.006808510638
Create 1 million new contracts	3200000000	960	283200	0.001041666667	3.53E-06	0.000146875	6808.510638
Save a 256-bit word to storage	20000	0.0006	0.177	1666.666667	5.649717514	235	0.004255319149
Save 1 MB to storage (31250 256-bit words)	625000000	18.75	5531.25	0.05333333333	0.0001807909605	0.00752	132.9787234
Save 1 GB to storage (1000 MB)	625000000000	18750	5531250	5.33E-05	1.81E-07	7.52E-06	132978.7234
Send 1 transaction	21000	0.00063	0.18585	1587.301587	5.380683347	223.8095238	0.004468085106
Send 1 million transactions	21000000000	630	185850	0.001587301587	5.38E-06	0.0002238095238	4468.085106
Include 1 byte (non-zero) in transaction data	68	2.04E-06	0.0006018	490196.0784	1661.681622	69117.64706	1.45E-05
Include 1 MB in transaction data	68000	0.00204	0.6018	490.1960784	1.661681622	69.11764706	0.01446808511
Include 1 GB in transaction data (1000 MB)	68000000	2.04	601.8	0.4901960784	0.001661681622	0.06911764706	14.46808511

Figure 7.7 Cost analysis of different blockchain operations

7.8 Risk analysis

This section focuses on the review of recorded and probable attacks on the Blockchain network and an analysis on the application's resistance to attacks is made. Only the main probable attack surfaces from malicious users on the Ethereum network are considered.

- **Re-entrancy attacks**: In re-entrancy attack, if the user does not update the balance before sending ether, an attacker can steal all the ether stored in the contract by recursively making calls to the call.value() method in a ERC20 token. As such, a careless user may lose his entire balance in the contract if he forgets to update his balance.

```
// Vulnerable Smart Contract mapping (address -> uint)
private userBalance;
function withdraw()
public
{
uint WithdrawAmount = userBalance[msg.sender];
if (!(msg.sender.call.value(WithdrawAmount)()))
        {
                throw;
        }
// Caller's code executed and it can make recursive
// call to withdraw() again.
userBalance[msg.sender] = 0;
}
```

Figure 7.8 Re-entrancy attack snippet code

- **DoS attacks**: DoS attack in a smart contract allows a malicious actor to keep his own funds and authority. Consider a smart contract auction example where a fraudulent bidder seeks to become an auction chief. The insecure contract forbids the old contract holder from being refunded and makes the new leader the intruder. In Ethereum, the contract agreement fails if the total gas produced by the smart contract after execution reaches the gas cap.

```
// DoS attack
contract Malicious_Auction
{
address presentLeader;
uint maxBid;
function bid() payable
{
require(msg.value > maxBid);
require(presentLeader.send(maxBid));
// Refund the old leader, if it fails then revert
presentLeader = msg.sender;
maxBid = msg.value;
```

Figure 7.9 DoS attack snippet code

- **Overflow attacks**: An overflow in a smart contract happens when the value of the type variable (2256) is exceeded. For instance, in a smart contact of online betting, if someone sends large amount of ether, exceeding (2256), the value of the bet

would be set to 0. Although exchange of an ether value greater than (2256) is unrealistic, but it remains a programming vulnerability in smart contracts written in Solidity.

- **Short address attacks**: The short address attack exploits a flaw in the virtual machine of Ethereum to make additional tokens on transactions that are minimal. For the most part, the short address attack refers to ERC20 tokens. The attacker produces an Ethereum wallet that ends with 0 digit for this attack. Then he makes a transaction by deleting the last 0 on the email. If the contract has an adequate balance, the buy function will not check the address of the sender and the virtual machine of Ethereum will add missing 0 to complete the address.

Chapter 8

CONCLUSIONS AND FUTURE ENHANCEMENTS

In this project work, the proposed idea is 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.

Results of this empirical study provide a global overview of the world of Smart Contracts. This world can be described as very active in the usage of the blockchain, heterogeneous in the typologies and in the code features, and supported by an interactive and reactive development community. The research leads to several outstanding findings that is summarized below.

- It is found that the Smart Contract developers' community follows and constantly adheres to the evolution of the Smart Contract programming language, Solidity. In fact, the update of a Smart Contract for bug fixing and, in parallel, depends upon the disposal of the old one, since there is not possibility to update or modify the source code once the contract is deployed on the Blockchain, as instead occurs for traditional software.
- By analyzing the purposes of various Smart Contracts, it was observed that developers have overtaken the concept of "parties' agreements" that characterizes the first era of Smart Contracts. It is also note-worthy that just a few percent of the total Smart Contracts are used to deposit ether.
- There are strong evidences on the practice and importance of code reuse. Thanks to the availability of thousands Smart Contracts source codes, developers start from already implemented contracts to create new and more efficient applications, or updated and customized versions of former Smart Contracts. In addition, source codes are generally well commented, and this helps new developers to understand their contents.

- It is discovered that some specific names are commonly used even if in general are associated to very different source codes with different purposes. Thus, it can be concluded that contract name is not representative of the contract's purpose and code.
- The interaction of deployed Smart Contracts with the blockchain by means of usage indicators was analyzed and it was discovered that the number of transactions follow a power-law distribution. Indeed, Smart Contracts with high balance may use a low number of transactions and conversely Smart Contracts with very many transactions may have a low balance.

This work presents the setup, the analysis and the results of an empirical study on a set of Ethereum Smart Contracts and on their source code. This empirical study examined the dataset from several points of view, like the use and the evolution of the Solidity compiler version and the related Solidity constructs, the number of interactions and transactions among Smart Contracts.

As long as future works are considered, this contribution benefits from an entirely decentralized architecture offering inherent fault tolerance, redundancy, and transparency. The plan is to implement this proposal to base the results empirically and demonstrate its viability. Our next proposal is to set up an architecture with smart contracts to validate, store and revoke the certificate on a public blockchain. The certificate of the individual will contain, his address and public key, the address of smart contract that issued it, and stored it in Off-chain.

Future works should consider to analyze a higher number of Smart Contracts (taking into account the set of Smart Contracts without available source code), further and specific code metrics (i.e to evaluate eventual code optimization in order to limit the Ethereum *gas* consumption or to measure the use of libraries and the interaction with already deployed contracts), other usage indicators (such us the internal transactions and the interaction between deployed contracts) and a wider analysis of correlation.

Also while concerning future enhancements, 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.

REFERENCES

[1] Rishav Chatterjee and Rajdeep Chatterjee, "*An Overview of the Emerging Technology: Blockchain*", 2017 International Conference on Computational Intelligence and Networks.

[2] Mohammad Dabagg, Mehdi Sookhak and Nader Sohrabi Safa, "*The Evolution of Blockchain: A Bibliometric Study",* IEEE Access 2019 vol7. Date of current version February 22, 2019.

[3] Pinyaphat Tasatanattakool and Chian Techapanupreeda, "*Blockchain: Challenges and Applications*", 2018 IEEE.

[4] Mohamed Abdullaziz, David Culha and Ali Yazichi, "*A Decentralized Application for Secure Messaging in a Trustless Environment*". International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism Ankara, Turkey, 3-4 Dec, 2018.

[5] Harry Halpin, Marta Piekarska, "*Introduction to Security and Privacy on the Blockchain*", EuroS&P 2017 - 2nd IEEE European Symposium on Security and Privacy, Workshops, Apr 2017, Paris, France. IEEE, Security and Privacy Workshops (EuroS&PW), 2017 IEEE European Symposium on, pp.1-3, 2017, 10.1109/EuroSPW.2017.43.

[6] Sachchidanand Singh and Nirmala Singh, "*Blockchain: Future of Financial and Cyber Security*" ,2016 IEEE.

[7] Wei Cai (Member, IEEE), Zehua Wang (Member, IEEE), Jason B. Ernst (Member, IEEE), Zhen Hong (Student Member, IEEE), Chen Feng (Member, IEEE) and Victor C.M. Leung (Fellow, IEEE), "*Decentralized Applications: The Blockchain-Empowered Software System*". IEEE Access VOLUME 6 2016.

[8] Shuai Wang, Yong Yuan, (Corresponding author, Senior Member, IEEE), Xiao Wang, Juanjuan Li1, Rui Qin, Fei-Yue Wang (Fellow, IEEE). "*An Overview of Smart Contract: Architecture, Applications, and Future Trends*". 2018 IEEE Intelligent Vehicles Symposium (IV) Changshu, Suzhou, China, June 26-30, 2018.

[9] Maher Alharby, Amjad Aldweesh and Aad van Moorsel, "*Blockchain-based Smart Contracts: A Systematic Mapping Study of Academic Research (2018)*", IEEE Access VOLUME 5,2017.

[10] Maximilian Wöhrer and Uwe Zdun, "*Smart Contracts: Security Patterns in the Ethereum Ecosystem and Solidity*", IEEE Access VOLUME 3,2018.

[11] Peter Menegay ,Jason Salyers and Griffin College, "*Secure Communications Using Blockchain Technology*", Milcom 2018 Track 3 - Cyber Security and Trusted Computing 2018 IEEE.

[12] Giuseppe Destefanis, Andrea Bracciali, Michele Marcesi, Marco Ortu, Robert Tenelli, Andrea Bracalli and Robert Hierons, "*Smart Contracts Vulnerabilities: A Call for Blockchain Software Engineering?*", IWBOSE 2018,Campabosso Italy.

[13] Christopher G Harris, "*The Risks and Challenges of Implementing Ethereum Smart Contracts*", 2019 IEEE.

[14] Joshua Hannon, "*Introduction to Smart Contracts and dApp*", Medium, Feb 18,2018.

[15] 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.

[16] "Ethereum Homestead 0.1 documentation", Ethdocs.org, 2018. [Online]. Available: http://ethdocs.org/en/latest/introduction/index.html. [Accessed: 12Aug- 2018].

[17] S. Kniesburges and C. Scheideler, "Hashed Patricia Trie: Efficient Longest Prefix Matching in Peer-to-Peer Systems", WALCOM: Algorithms and Computation, pp. 170-181, 2011.