Advanced Health Insurance System Using Smart Contract of Ethereum Blockchain

by

Mehedi Hasan Tushar
19101589
Sifat Abdullah
19101384
Aar Rafi Shahriar Shad
22241129
MD. Hisam Bin Hyee
19101553
Sheblee Mohammad Hesham
19101594

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
May 2023

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.

2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.

3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.

4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

[illegible]

Mehedi Hasan Tushar
19101589

[illegible]

Sifat Abdullah
19101384

[illegible]

Aar Rafi Shahriar Shad
22241129

[illegible]

MD. Hisam Bin Hyee
19101553

[illegible]

Sheblee Mohammad Hesham
19101594

Approval

The thesis titled ”Advanced Health Insurance System Using Smart Contract of Ethereum Blockchain” submitted by

1. Mehedi Hasan Tushar(19101589)
2. Sifat Abdullah(19101384)
3. Aar Rafi Shahriar Shad(22241129)
4. MD. Hisam Bin Hyee(19101553)
5. Sheblee Mohammad Hesham(19101594)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on may 22, 2023.

Examining Committee:

Supervisor:
(Member)

Iqbal

Dr.Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr.Md.Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr.Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

Every year, more healthcare insurance frauds are discovered, which is hugely worrying for society. Healthcare insurance fraud takes many forms, including fabricating data, obfuscating third-party responsibility, misrepresenting electronic invoices, etc. Policyholders can deceive general clients with these methods too. The current health insurance system strains resources and labor. In light of the numerous different techniques that are used to digitize healthcare insurance along with security, we will represent a blockchain-based strategy that is built on smart contracts as a means of avoiding insurance data manipulation, resulting in a secured, immutable, and trustworthy system with efficiency in data integrity. This system relies on organizations like judicial bodies. This blockchain system includes organizations like insurance companies as well. Moreover, information about medical costs, prescriptions, inquiry reports, and medical history is used in the operating procedure to create a blockchain for health insurance. In this thesis, we propose third-party treatment activity assessment, extravagant medical behavior look-up and evaluation of data from medical invoices, and liability inspection services.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to our advisor DR. Muhammad Iqbal Hossain sir for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents without their throughout sup-port it may not be possible.
With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

Chapter 1

Introduction

In today's environment, everyone is dependent on health insurance. Our lives wouldn't be the same without it. In the unfortunate circumstance that you find yourself in need of medical assistance, having health insurance can entirely or partially relieve the financial burden that comes along with it. The risk gets split among several parties in a similar manner to other types of insurance. By assessing the overall risk of health-related hazards and expenses within the risk pool, an insurance provider can establish a consistent financial structure, such as a monthly premium or payroll tax, to furnish the necessary funds for the medical benefits outlined in the insurance policy. The administration of the benefit is overseen by a central entity, which may take the form of a governmental body, a profit-driven corporation, or a privately owned enterprise. Every year, more healthcare insurance frauds are discovered, which is hugely worrying for society. Falsifying information, avoiding third-party liability, misrepresenting an electronic invoice, etc. are just a few examples of the many different ways that healthcare insurance fraud can occur. Since it costs a lot of money and effort, the present health insurance system is severely taxed. To eliminate all these problems in the health insurance system, we want to introduce the smart contract of the Ethereum blockchain. Before diving into smart contracts, the idea of blockchain needs to be clear. A blockchain refers to a decentralized database or ledger that is distributed across the nodes that are part of a computer network [13]. And a distributed system enables networked systems to share resources, including software. Finally, a smart contract is a program stored on a blockchain. It only gets issued once the preconditions are met. Blockchain is an immutable and secure way to transfer information. In order to add new blocks to the existing blockchain, it makes use of some validation procedures and mining. We are going to use the Proof of Authority protocol in order to increase the efficiency of the system. The healthcare insurance anti-fraud service's primary objective, according to the authors of [1], is to determine whether or not the patient's claim for medical reimbursement is genuine and complies with all relevant insurance policy requirements. Their system has addressed the implementation of three services. They proposed a system architecture to put the aforementioned three services into practice. The architecture consists of various layers, including a layer for the cloud platform, a layer for the network, a layer for the core, an interface layer, and an application layer. Also, we have introduced DApp, which means decentralized web application, to help facilitate our system even further using truffle and ganache.

1.1 Motivation:

Most people nowadays rely greatly on healthcare insurance. Life without health insurance can hardly be imagined. Health insurance, or medical insurance, is a form of coverage that provides partial or complete financial protection against the risks associated with requiring medical treatment. As with other forms of insurance, the distribution of risk is shared among multiple parties. By assessing the collective health risks and expenses within a risk pool, an insurance provider can establish a standardized financial structure, such as a monthly premium or payroll tax, in order to generate the necessary funds for covering the medical benefits outlined in the insurance contract. The administration of the benefit is carried out by a centrally controlled organization, which may comprise a governmental agency, a profit-oriented enterprise, or a privately owned firm. More healthcare insurance frauds are uncovered each year, which is very concerning for society. Falsifying information, avoiding third-party liability, misrepresenting an electronic invoice, etc. are just a few examples of the many different ways that healthcare insurance fraud can occur. The current health insurance system is heavily taxed since it requires a lot of money and effort. Hence, a need has arisen to reshape the health insurance system for the purpose of having a flawless system. With this view, we have introduced the smart contract of the Ethereum blockchain.

1.2 Problem Statement:

The manual settlement of conventional health insurance policies is a process that is seldom devoid of difficulties. The complexity of the settlement process can be attributed to various factors, such as undisclosed stipulations from the insurer or allegations of deceit on the part of the policyholder. Furthermore, this approach is characterized by a protracted duration, rendering it highly inefficient. The potential integration of blockchain technology and smart contracts within the health insurance sector has the capacity to significantly disrupt the current landscape. Innovative contract technology and blockchain can offer immutable data storage, security, authenticity, transparency, and security when any transaction process is initiated. The entire health insurance procedure, from verification to claim settlement, can be carried out with better security and transparency thanks to the adoption of blockchain technology. A decentralized technology known as a blockchain is a digital chain of data chunks. The decentralized validator entity, rather than a single person, is responsible for each and every transaction as well as any modifications that need to be made to a block. The smart contract is a distinctive attribute that is documented on the blockchain and is triggered upon fulfillment of specific conditions. One of the primary challenges faced by blockchains pertains to the inherent intransitivity of smart contracts. The concept of immutability pertains to the unalterable nature of blockchain-based smart contracts, wherein the established rules of the protocol remain fixed and cannot be modified. The immutability of a blockchain ledger is characterized by its ability to persist in an unmodified, unalterable, and irreversible state. Each block of data, such as transaction details or facts, is processed using a cryptographic principle or hash value. Before diving into the limitations of smart contracts for any digital insurance system, let us know what a hash and hash count are in a blockchain network. A hash value is a fixed-length numerical representation that serves as a unique identifier for a large amount of data. In the realm of digital signatures, it is common to represent large amounts of data using significantly smaller numeric values. Signing a hash value can be more efficient than signing a larger value. When compared to signing with a larger value, a hash value is more effective. The hash value is shared with the next block inside the same network. Here are the limitations of smart contracts (immutability):

1. Difficult to change once it is deployed inside the network: One of the properties of the blockchain that is commonly mentioned is its immutability. A blockchain is technically an immutable database, which means that information that has already been entered cannot be modified. A block's hash value uniquely identifies it. Because it depends on the content of the block, every block contains a unique hash value that is used just to identify that specific block. The four-block is taking a reference to the third one, which is taking a reference to the second, and so on. As a result, each block can refer to or point to the block before it. Therefore, the hash value refers to the previous hash value of that specific block.

2. Any error in the code can be time-consuming and expensive to correct: If any miner wants to add a smart contract to any block inside the network, then after validation, the smart contract cannot be changed. Normally, it needs to change the whole network or delete it later; otherwise, it will show an invalid block. If any miner wants to do a transaction, then it will be more time-consuming [7]. Miners carry out these smart contract transactions in order to add a proper block to the blockchain. Later, the block's smart contract transactions are serially re-executed by the validators.

3. Possibility of loopholes. Couldn't count or keep track of hashes: The ability to accurately enumerate or monitor hash values was not feasible. The mechanism by which blockchain technology operates is through a settlement process that is driven by push-based transactions. This implies that individuals have authority over the resource they intend to authenticate on the blockchain. Land titles, certificate authentication, and digital currency are illustrative instances of this phenomenon. In the context of blockchain technology, a transaction that has undergone authentication and subsequently experienced an error poses a significant challenge as there is no feasible mechanism for its retrieval unless mutual consent is obtained from all involved parties for its reversal. Utilizing a centralized server, such as a bank, However, there is a process in place to resolve disputes over completed trades. In this way, fraudulent smart contracts are detected in the network. And hash values aren't kept on track.

4. Third party chains are not accessible: The blockchain mining method is an innovation that makes use of game theory economics to motivate users to invest computer resources in protecting the network in exchange for money. The drawback of this is that, in general, miners won't care about settling as many transactions as they possibly can; instead, they'll be more interested in identifying and verifying blocks as quickly as they can in order to maximize their profits. As a result, miners have difficulty verifying empty blocks. Selfish Mining is a different issue that arises when a miner or mining pool finds and validates a block but fails to disclose and disseminate a working solution to the

rest of the network. To add to that, if a third-party chain of a small network wants to enter the main network, the miners cannot access the blocks of the third-party chain; otherwise, it will become a hybrid blockchain.

5. Vague terms and protocols are maintained: Pow, PoS, DPoS, PoA etc. are the three most commonly utilized consensus protocols. The aforementioned mechanisms are upheld to enhance scalability and bolster network durability. When considering efficiency and security, it can be observed that PoS and PoW are relatively less efficient. A hash value is a fixed-length numerical representation that serves to uniquely distinguish vast amounts of data. The technique involves representing large amounts of data as significantly smaller numerical values, which are commonly utilized in conjunction with digital signatures. As PoS does not require miners to expend energy in the consensus process, it is possible for someone with malicious intentions to take over the network by acquiring a sizable amount of stake tokens. This could lead to hacks that falsely claim to provide security and fund theft from users of blockchain networks. But in the case of PoA, miners are also counted as validators or authors of each chain. In this case, we can see a lack of decentralization in PoS algorithms. Due to the ease with which larger nodes can overwhelm smaller ones, it may result in a lack of decentralization. This is so that the network can be maintained and transactions can be verified, which is how PoS systems work. Smaller nodes find it more challenging to take part in the consensus process since larger nodes may easily control the delegation choices. As a result, PoS networks may eventually become less decentralized, which is in opposition to one of the fundamental ideas behind blockchain technology.

1.3 Objective and contributions:

Our main goal is to digitalize the whole insurance system. We want to move from paperwork to a totally online system with security enhancements. By doing that, the system will be much more efficient, as we can reduce the administrative cost. Also, it saves time and resources for both patients and providers. We want to create a trusted system where every condition is clear and hassle-free. It will be user-friendly and fraud-free. By lowering the possibility of paper records being lost or stolen, digitizing health insurance systems can increase security. To safeguard patient privacy and confidentiality, digital records can also be encrypted. This system will save time and speed up every transaction. Any transaction can be done very quickly. One of the major features will be that it won't take working days for a transaction. Instead, it can be done anytime and anywhere if it's distributed. People may find it simpler to obtain and manage their health insurance information online with the help of digital health insurance systems. People with mobility challenges or those who live in distant places may find this to be of special benefit. An implementation of a distributed network will make the whole system more secure. With the help of blockchain technology, we can make the settlement process really smooth. In addition, we want to use smart contracts to avoid any hidden conditions and charges by the company, which will attract more customers to avail of the insurance. A handwritten contract can be manipulated or faked. But a smart contract cannot be manipulated, and someone cannot make a false contract. By confirming the validity of claims and guaranteeing that only qualified individuals receive insurance benefits, it can lower the risk of fraud. This could lower costs and strengthen the credibility of the health insurance market. Therefore, the contract will be immutable and transparent to both users and companies. In transactions involving health insurance, smart contracts can increase accountability and transparency. All parties to a transaction have access to the contract's terms and can follow its development. This type of transparency makes a company more trusted by the public. So, we can say that it will be one of the most secure systems to date. Our goal is to attract more consumers to insurance by giving them a smooth and hassle-free automated system where several of the duties related to health insurance, including payment processing, eligibility verification, and claim processing, can be done automatically. Also, we will try to make all the insurance companies adopt a distributed system for their own benefit. A distributed system can facilitate better data management by giving insurers a more thorough understanding of client information across several systems. This can aid insurers in making better selections and enhancing the level of their services. More-over, by dispersing data among several nodes, a distributed system can increase security by lowering the possibility of data breaches and cyberattacks. This can assist insurers in safeguarding private consumer data and lowering the possibility of expensive security issues. We will try to use some of the advanced protocols, like Proof of Authority, which will make the insurance industry move into a totally new era of the digital world of blockchain networks.

Chapter 2

Background:

Blockchain is a distributed system which means every member of the system has access to every other information which is included in the blockchain and if someone tries to manipulate or change anything related to the blockchain then everybody else will get the notification and they will be able to prevent that from happening.[16] It uses some validation processes and mining to include new blocks in the existing blockchain and it takes a lot of time and has a lot of wastage in energy consumption. Therefore they are trying to improve this by adding new and upgraded models to have faster transactions and to prevent excessive energy consumption. POS, POW and POA are some of the protocols which are in use to make that happen. And we are going to try to find a solution by using POA in the health insurance sector to prevent fraud and give people what they deserve. POA stands for Proof of authority .The PoA agreement varies from the PoS agreement in that it employs character instead of the advanced resources each client possesses. Character here alludes to the certainty that validators are who they say they are, as appeared by the correspondence between their individual character on the stage and any formal archives displayed by them. Hence, a person's notoriety is more important than their belonging.[15]Validators are pre-approved by a bunch of "authorities" to confirm exchanges and construct unused pieces. To be trusted, validators must follow a set of necessities. Two of these requests are that they utilize the same character on the stage to enroll within the open public accountant database and take after the rules to guarantee that the network functions appropriately. Moreover, the method for choosing specialists ought to follow acknowledged standards to guarantee that all candidates have an opportunity to be chosen for the advantaged position. Finally, specialists (around 25 substances) must unveil their character to act in their position, for which they get compensated in return. However, using the PoA calculation for a decentralized arrangement is challenging as it were as if a couple of clients hold control. The PoA agreement is in this manner considered a private organized arrangement rather than an open blockchain arrangement.[15]PoA and PoS calculations offer points of interest and impediments comparable to any agreement component. Besides, within the history of blockchain, no designer or stage has howhowever, been able to put forth an agreement instrument that's impenetrable to issues or objections. The PoA calculation diminishes the control required to run the ar- rangement and makes validation easier. On the other hand, staking within the PoS agreement component encourages decentralization by permitting personal support in organized security. The PoA calculation does not require puzzle-solving to ensure the ongoing connection between hubs. In this manner, the validators don't require specialized equipment to preserve the arrangement. In any case, three diverse pieces of software, including an execution client, an agreement client and a validator, are required to take part within the Ethereum staking handle. The speed at which the specialists approve exchanges is quickened by the proof-of-authority calculation.

2.1 Literature Review:

The healthcare industry is constantly reorganizing and embracing new forms in response to technological evolutions and changes [4]. As insurance became more prevalent in modern consumer society, the amount of fraud surely increased. The cornerstone of fraud is information asymmetries. Access to key essential information is commonly restricted to one or a small number of the transacting parties during critical transaction points over the course of an insurance contract. Often, the person with the benefit of information has a strong incentive to commit fraud [6]. Many types of health insurance fraud occur in each nation, with different parties involved. The basic goal of the healthcare insurance anti-fraud service is to determine if a patient's claim for medical reimbursement is legitimate and fits all applicable insurance policy criteria [1]. However, there are many approaches made to prevent fraud and digitalize the health care system based on blockchain, data mining, machine learning, or other analytical methods [9]. Algorithms based on machine learning are ineffective when faced with large amounts of diverse data. Large and complex collections of health data that are difficult or impossible to gather, manage, and analyze using conventional or standard data management tools and procedures are referred to as big data in the healthcare sector [9]. The business community has been interested in the possibility of legally binding smart contracts where contract performance is automatically enforced and supervised by an independent automated party [5]. Blockchain has transaction logic built in that controls the whole health insurance business cycle. The blockchain is a distributed system of nodes that communicate through intelligent contracts, thereby obviating the necessity for an intermediary. Upon the completion of a transaction, smart contracts are executed to carry out the intended operation [7].

The study [10] provides an analytical overview of the smart contracts in Ethereum blockchain. Ethereum, according to the author, is a blockchain with a Turing-complete computer machine embedded. Also, smart contracts are short computer programs that are attached to a specific blockchain address that corresponds to the smart contract software code and are maintained on the Ethereum public ledger or another blockchain. Many blockchains can run applications that implement smart contracts. The Ethereum platform is the first blockchain designed specifically to execute smart contracts and is the most well-known of them [10]. However, once deployed, a smart contract cannot be changed. Because of this, it is necessary to fix the software flaw before deploying the smart contracts on the blockchain network [7]. The author of [5] stated that their goal is to provide a mechanism for the creation and execution of legal smart contracts that may be altered in response to requests from one party only. For creating and deploying legal contracts on the blockchain, they suggest a methodology and architecture in their study. They have used three tiers, which are the presentation tier, the business and data tier, and the blockchain tier, in their system architecture. Also, they have selected a versioning system for concurrent alteration representation and a data separation implementation technique to facilitate efficient contract logic changes.

According to the authors of [1], their system has addressed the implementation of three services to prevent fraud. When information about an injured patient is exchanged across several authorized organizations and agencies, the possibility of discovering healthcare insurance fraud increases. But these organs use different informational systems and can be placed in different places. The author may utilize the cross-cloud platform built on a blockchain service architecture to address this issue. They proposed a system architecture to put the aforementioned three services into practice. The architecture comprises five distinct layers, namely the cloud platform layer, network layer, core layer, interface layer, and application layer. It was stated that three distinct services were deployed using three interconnected blockchains. Firstly, the blockchain of medical processing information The medical process information check service allows the healthcare insurance agency to determine if the medical service process that corresponds to the reimbursement application has been violated in any way. This is based on permission-based blockchain technology. After the smart contract checks the relevant healthcare insurance rule during medical process information querying, the service will detect the suspicious point in line with the common illness diagnosis and treatment procedure. The architecture has a cloud platform, network, core, interface, and application layers. Three connected blockchains implemented three services [1]. Secondly, the third-party responsibility inspection service, which primarily assists the healthcare insurance company in determining if the trauma reimbursement application for insurance is concealing third-party responsibility fraudulent activity, This is based on the blockchain for third-party responsibility inspection [1]. Finally, the third one is based on the healthcare insurance bill's blockchain. Healthcare insurance companies frequently utilize bill inspection services to check for fraud or manipulation in the invoices submitted for reimbursement under healthcare insurance [1]. In addition to the three blockchains outlined above, they have developed a cross-blockchain solution to facilitate cross-blockchain interaction. They stated that the smart contract service supports many channels for cross-chain transactions. Also, they have proposed a data privacy protection scheme with three phases. The phases stated in the permission blockchain data flow are smart contract processing, block file storage, and data upload [1].

In the study [4], the authors want to make a solution based on blockchain to enhance data integrity and security. They have designed a system following MVCA architecture. They have done the whole process by user interaction with the system, storing data in a database, and validating the data using an EDI validator. The documentation is shared on the blockchain only when the validation is successful. The authors have divided the implementation of the process into four phases. They obtained the essential EDI 837 claims standards for phase 1 by researching them online and on the official Medicare website. They also acquired 837 files from Lumeris, Inc. In order to handle various types of validation and enhance speed, four separate levels are executed in four simultaneous threads. In phase 2, each insurance company will be provided with a BigchainDB node. They used the MongoDB database to store the blocks in each BigchainDB node. The third phase involves the creation of the application's user interface, which is utilized by healthcare professionals. A secure (SSH) tunnel is also used to transfer data between the frontend and the backend. Finally, the fourth phase has implemented the deployment of the environment on AWS, which will continue indefinitely to service all incoming requests. The authors claim to use blockchain to guarantee default security for the data contained inside it [4].

A blockchain- and AI-based system was suggested by the author in [7] as a means of combating different security challenges in health insurance fraud detection. This method also promotes transparency and subscriber trust in the health insurance provider. They have used four different layers in their proposed model. The four layers that have been identified are the user layer, the data generation layer, the data analytical layer, and the blockchain layer. The stratum of users is interconnected with a blockchain infrastructure that documents all user engagements. This layer is interconnected with the data generation layer in addition to the blockchain. The data-generating layer will be shared and controlled under smart contract conditions for security reasons. The author believes that using robust ML algorithms and data preparation are the best strategies to manage such data. Several ML techniques are used at this layer to assess if a claim is legitimate or fraudulent. Through the blockchain layer, this prediction result is connected to a smart contract. The blockchain layer ensures the security of all transactions, including payments and authentication [7].
A blockchain-based system with smart contracts is used in [11] to provide security, reliability, availability, durability against hacking and harmful attacks, seamless integration, and simple data administration. Which was named SHealth by the authors. The architecture of the system in [11] has been divided into four main layers. The first is the government layer, which has the most authority in the system. The role of this layer is to manage access to the blockchain by authorizing new nodes and permitting new users to join the network based on their national IDs. The user layer is the second layer that they mentioned. Users in this context are essentially individuals with their own records who interact with the system through the API. Each system user has a unique and private cryptographic key that is controlled via a wallet application installed on their smartphone or tablet. The layer that follows the second tier in the SHealth framework is referred to as the IoT terminals layer. The provider's node contains authorized medical gadgets and equipment that operate within the Internet of Things and perform specific health processes for clients. The last layer is the blockchain layer, which stores the clients' information. They have separated all the nodes into different tiers.

The following system [12] provided an efficient technique for blockchain-based electronic health record authentication: The authentication of blockchain-based electronic health records includes the two cases they described. Case 1 shows that Level 2 users must authenticate any data they supply. Case 2 concerns the accuracy of the block data that Level 1 users have submitted. Here, level refers to the different groups of users. For instance, the EHR server is at level 0, which is the lowest level. Level 1 encompasses diverse organizational structures. The third tier in the

hierarchy is Level 2, which pertains to the workforce of Level 1 users. The typical approach for implementing a blockchain-powered electronic health records (EHR) system involves utilizing an identity-based signature mechanism that involves multiple authorities. The layer that follows the second tier in the SHealth framework is referred to as the IoT terminals layer. The provider's node contains authorized medical gadgets and equipment for the Internet of Things that perform specific health processes for clients. The cited paper [12] contains effective signature and verification methods for the technique.
All The studies mentioned above attempted to improve the entire healthcare system by serving various purposes such as increasing information security, fraud control, and creating a more reliable digital system through the use of different computer technologies, particularly blockchain and smart contract technology.

2.2 Algorithms/Model/Existing Techniques:

We are looking to create an insurance system that includes verification from the validators who are assigned randomly by the server of the bank and check whether the request done by the client is abiding by the rules and limitations of the contract. The smart contract which includes those rules will be available to everyone via the Ethereum blockchain network and it will be written in solidity to prevent problems such as taking too long or endangering the system. It will be a private system which means only specific people will have access to this blockchain network. We will be trying to include the use of a Specific consensus algorithm POA(proof of Authority) in the system to try to prevent the delay which was available in POS and POW. An adjusted form of proof-of-stake is proof-of-authority, in which a validator's recognizable proof serves as the stake instead of a financial one. Moreover, due to the PoA consensus's effortlessness, it is basic to ensure validators' autonomy and give them with the devices vital to defend their hubs. These issues can be settled, in spite of the fact that. For case, the identity-at-stake PoA plan sets up an motivation show in which a validator's ideal course of action is to function within the best interface of the arrangement. Such a development is a charming worldview for blockchain agreement due to its cost-efficiency. Similarly, the reality that proof-of-stake may be utilized for much more than fair cash is what makes it so captivating. For occurrence, PoS calculations can be utilized in decentralized anti-spam frameworks, advancement of decentralized applications (DApps), security and adaptability of cryptocurrencies and possibly different other utilize cases that we haven't indeed seen yet.[15] in our model the client will send his request to the blockchain via the insurance agent who has access to that blockchain network. After getting validation from the validators the request will be accepted or denied. Here the validators will be less because of POA and the transaction speed will also be much faster and the wastages will decrease. The diagram of 2.1 demonstrates it as it were of the individuals and clients of the company. The blockchain in this case is the private Ethereum organization. The validators are already chosen and will approve exchanges per the rules of the Verification of Specialist calculation.

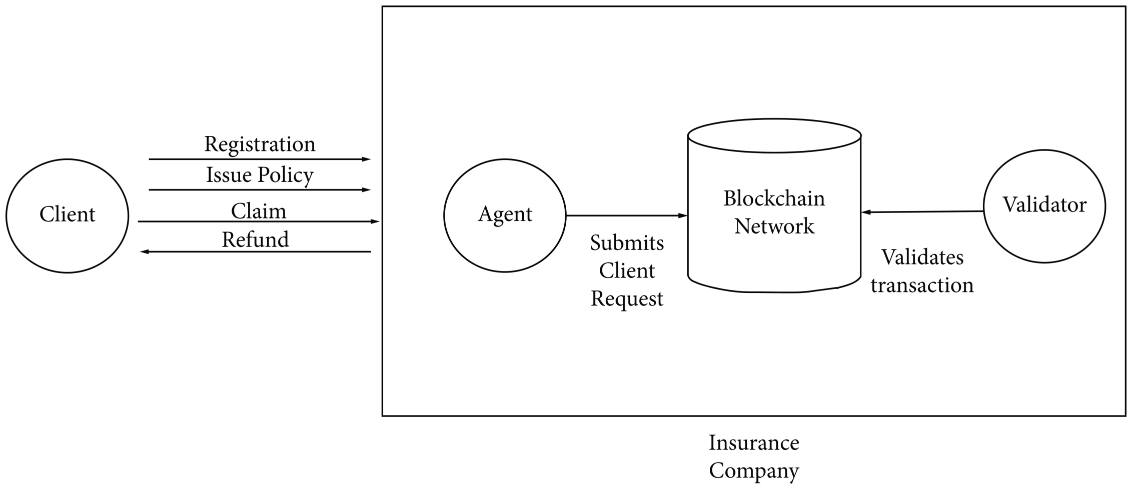


Figure 2.1: Workflow[1]

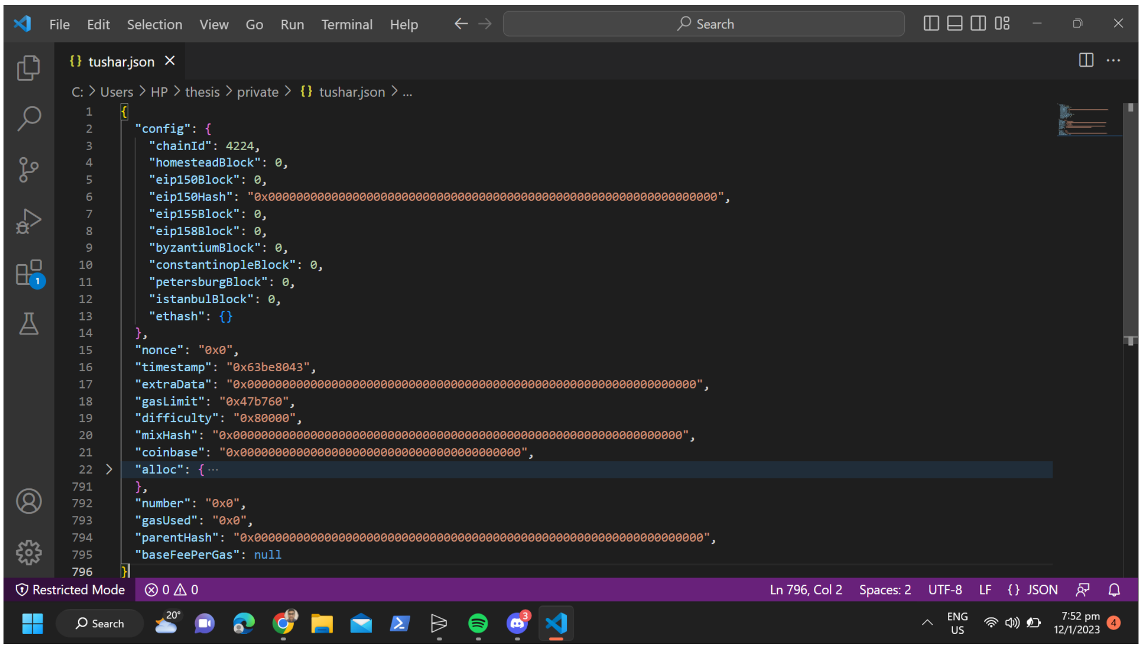
2.3 Information about used models and tools

At first we will collect necessary data from the client by which we can find the best possible insurance for his needs. After creating the Smart contract we will send it to the client to verify if he is pleased with that contract. After getting yes from them the smart contract will be deployed to the Blockchain network. And when the client makes a request in the future then the validators will run that contract and see if the request can be accepted or not if it does not clash with any clauses included in the contract. If the request gets approved from the validators when it will give off to the request and the client will receive what he requested. if the request goes against the contract then the validator will decline the request. This will be fair because the validator doesn't know who the client is and which insurance company will have to pay. Therefore the whole system will not be biased towards anyone. And because of the smart contract the contract can not be changed with others knowing.

2.3.1 Genesis Block

The genesis block is the name given to the first block of a blockchain network. It is the number zero block of each chain. a blockchain contains blocks of data that are connected via the previous hash of the genesis block containing the initial address. it has all zeros as its previous hash.

There are various parts included in the genesis block. gas price, version of the blockchain, model used for the blockchain, and some of the things which can be seen in the genesis block. Every Genesis block has a different Chainid which is unique for that blockChain only. The layers and extensive history of each sequence are among the characteristics that make a block-chain-based cryptocurrency so secure. In essence, blocks are digital containers that store and permanently keep data about network transactions. One or more recent transactions for insurance that haven't yet been included in any prior blocks are listed in a block. Consequently, a block is comparable to a page in a book or record. A block in the block chain "finalizes" when it passes control to the one after it. due to the origin Block, also referred to as Block 0, is the first block to which all subsequent blocks in a chain are added. Furthermore, it's necessary for creating a blockchain. Here the genesis block is named tushar.json with a ChainId of 4224.



```
{
  "config": {
    "chainId": 4224,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip150Hash": "0x0000000000000000000000000000000000000000000000000000000000000000",
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "istanbulBlock": 0,
    "ethash": {}
  },
  "nonce": "0x0",
  "timestamp": "0x63be8043",
  "extraData": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "gasLimit": "0x47b760",
  "difficulty": "0x80000",
  "mixHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "coinbase": "0x0000000000000000000000000000000000000000",
  "alloc": {⋯
  },
  "number": "0x0",
  "gasUsed": "0x0",
  "parentHash": "0x0000000000000000000000000000000000000000000000000000000000000000",
  "baseFeePerGas": null
}
```

Figure 2.2: Genesis Block

2.3.2 Hash

When given an input string, the "Hash" algorithm function will always produce an output with a set length. That is the essence of a hash function, as shown in the illustration below, which I will simplify in a few paragraphs:

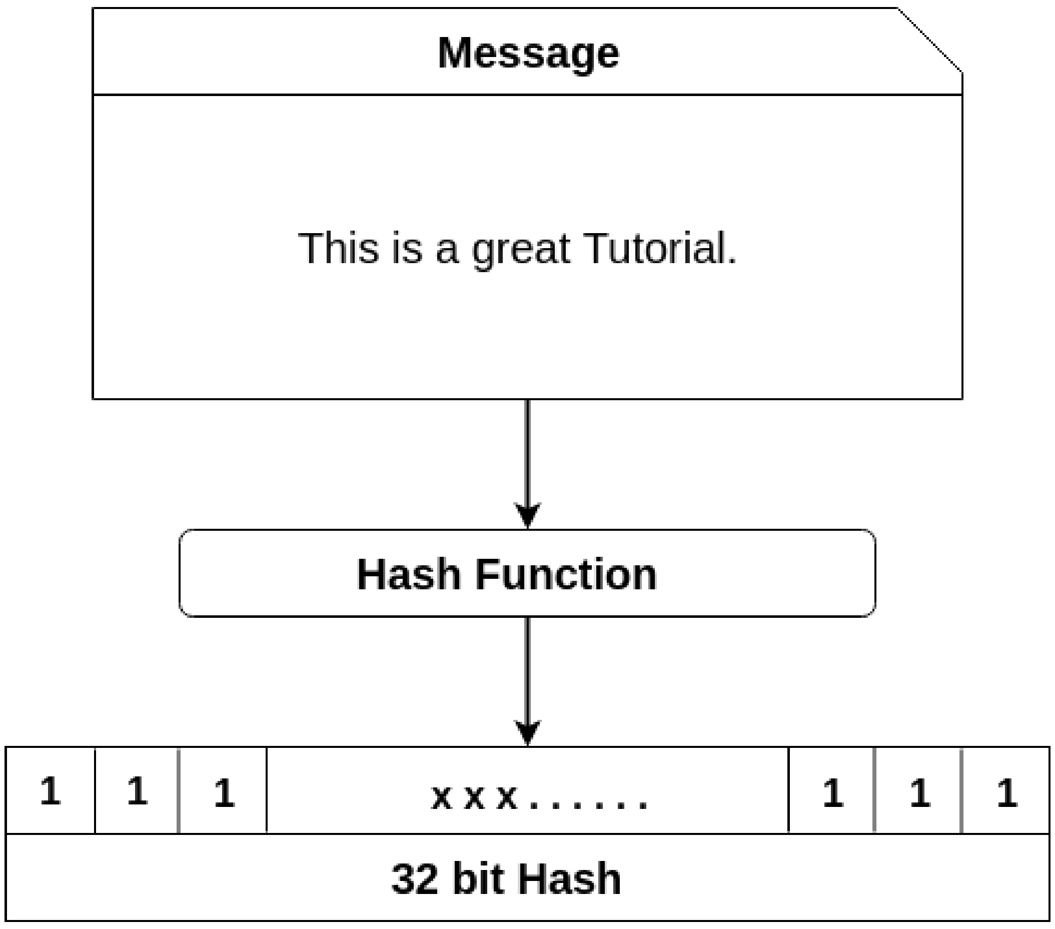


Figure 2.3: Hash

Due to its two primary characteristics—

1. Speed of retrieval, which stays nearly constant regardless of the size of the data;
2. Utilizing less disk space to keep the dataset—hashing offers certain advantages over alternative methods like lists or arrays. Here is a great resource that explains the benefits of hashing.

In our proposed model, we will use the Keccak-256 hash function.

2.3.3 Public key

Cryptocurrency transactions can be received if a public key is there. It consists of a cryptographic code and a private key. While anyone can send transactions to the public key, only the owner of the cryptocurrency that was received in the transaction can "unlock" such transactions and show they were sent by them. A public key's address, which is merely a condensed version of the public key, is often the public key that can accept transactions. As a result, public keys can be distributed without restriction.

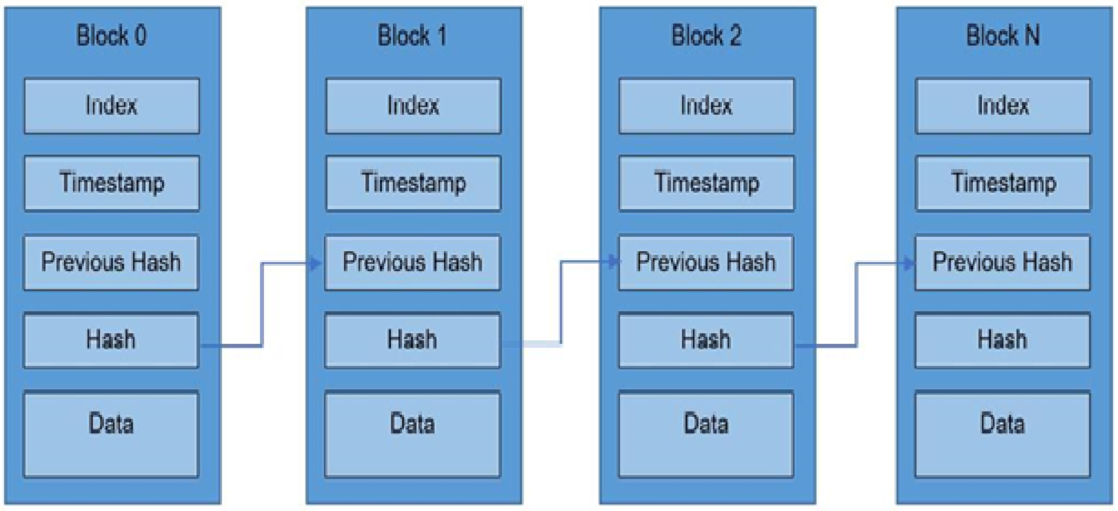


Figure 2.4: Hashing and connection in Blockchain

2.3.4 Private key

In our system ownership can be proved or the money can be used which is linked to the public address by using a private key. Several formats for private keys to exist:

1. binary code with 256 characters
2. Hexadecimal code, 64 bits.
3. QR code
4. Symbolic language

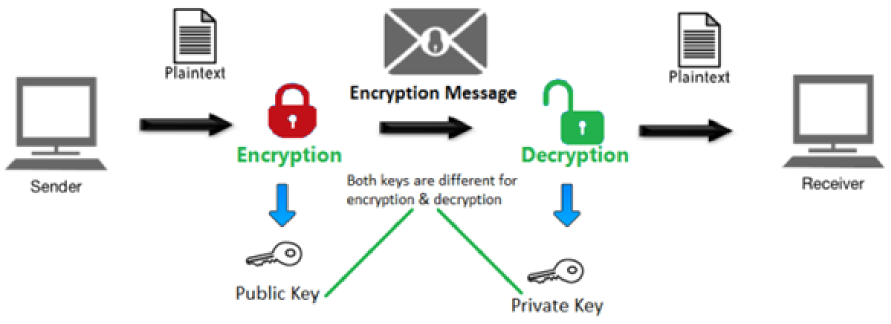


Figure 2.5: Encryption of keys in our network

The private key, regardless of its form, has a significantly large value, and its size is justified by its purpose. The generation of a public key can be facilitated by a private key. However, due to the presence of a one-way "trapdoor" function, achieving the opposite is a challenging task. A single private key can be associated with multiple public keys. Each participant, including the employee's node within the network, will be allocated two keys for individual encryption purposes. Through a process of key comparison, it is possible for an individual to determine the precise key utilized to hash the keys and subsequently gain entry to the corresponding account. The private key was generated using the same method as the public key. The Gmail account, National Identification Number, birth date, security number, and password provided by the individual were employed in the process.

2.3.5 Verifiable Credentials

For verification process in our application request system, we are using verifiable credentials for ultra-security purpose. Verifiable credentials as a digital alternative to physical documents [2]. As we know in real world insurance are done through paper works but, in our system, we will use verifiable credentials to add to their digital wallet through the company website. The necessity to ensure the accuracy of the transferred information and accompanying documents has arisen as a result of the digitization of several administrative and professional exchanges and procedures. Verifiable credentials enable you to claim ownership of your data, simplifying and protecting your privacy during data exchange. Sharing valid credentials in our system could streamline the insurance application process. Checking bank statements and identity documents serves the objective of demonstrating your identity and your reliability as a rent payer. Currently, each vendor you apply to requires you to present identity and supporting documentation. Instead, picture having access to credentials in a digital wallet that could guarantee your identity and dependability in order to pay for insurance. It also unifies the format of information, en- abling you to own your data and share it with the appropriate parties online which are mainly the employees of the insurance company in a controlled manner. VCs return control to the participants by hosting information outside of silos that only communicate via API connections.

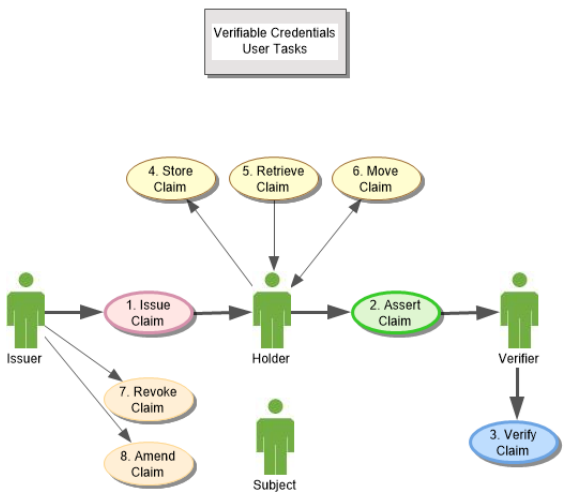


Figure 2.6: Verifiable credentials process

Here, the issuer is the employee or it can be the subject itself. The subject is the client or the insurance owner. Lastly, the verifiers are the author nodes of the system or the validators who are assigned for node validation. The issuer in our system gains a more efficient process by verifying service member information or the clients. Also, they don't bear the risk of storing the personal data of the subject. That data stays with the subject.

2.3.6 Mining

Mining is a peer-to-peer process where, nodes (miners or the authenticator who plays as a miner) add transaction records and the changes that may arrives to the decentralized ledger of the blockchain by solving a puzzle through different protocols of mining. Here we are using proof of authority protocol.

1. The transactions are added to the blocks, and the blocks are secured and linked after mining in the form of a chain [7]

2. Mining requires computational power and effort from miners. And basically, it is the process of adding blocks [7].

3. In this way, the transactions get confirmed and money and assets move from one account to another [7].

2.3.7 Proof of Authority

Proof-of-authority is a consensus technique that provides blockchains, particularly private blockchains, with an effective method for solving problems. Because the PoA algorithm places a high value on identity, people who aspire to hold the role of "authorities" are required to voluntarily divulge their identities. However, there are some other blockchain consensus algorithms like PoW, PoS etc. According to Anwar, H. (2018) PoW is the first introduced blockchain algorithm. The Proof of Work algorithm is good because it is very safe and has a high degree of decentralization. But the main problem with it is that it uses more energy and resources. The Proof of Stake (PoS) consensus technique was suggested by researchers as a solution to the PoW's issue with high processing power consumption. Both PoA and PoS algorithms have pros and cons, just like any other consensus mechanism. The PoA algorithm lowers the amount of power needed to run the network and makes it easier to check if something is correct. On the other hand, staking in the PoS consensus mechanism makes decentralization easier by letting each person help keep the network safe. PoA consensus differs from PoS consensus in that it relies on the user's identification rather than their digital assets. This implies that it depends on the reputation of trusted parties in a blockchain network [19]. While PoS has security weaknesses and PoW is energy intensive, PoA is the most secure and energy-efficient choice. Proof of Authority is a modified form of Proof of Stake that is a more centralized method for maintaining consensus on a blockchain network. The network relies on a number of pre-approved validators, often known as "authorities," for confirming transactions and producing new blocks. Validators are supposed to conform to a set of standards in order to be regarded trustworthy. Proof-of-authority is advantageous over real Byzantine fault tolerance because it requires fewer message exchanges and less overhead. This is one of the reasons why proof-of-authority outperforms Byzantine fault tolerance in practice [20]. If validators desire to be seen as trustworthy, they are expected to adhere to a set of rules. As this is one of the criteria, they must register in the public notary database using the same name they use on the website. People need to adhere to more norms for the network to be functioning. Becoming a validator is not a simple process. Candidates are expected to undergo a screening process in which they must show the level of commitment they will retain to the network during their employment there. In addition to putting their reputation at risk, they must also be ready to spend money on the selection process. Last but not least, the method for picking officials should comply to the previously set guidelines; this ensures that all applicants have an equal opportunity of obtaining the post. Validators are awarded power and advantages in return for their identification and presentation of official papers confirming their identity. Validators are granted authority and benefits [18]. Moreover, this algorithm verifies transactions more quickly. Blocks are created sequentially by authorized network nodes at regular intervals, resulting in a faster transaction rate than PoW and PoS.

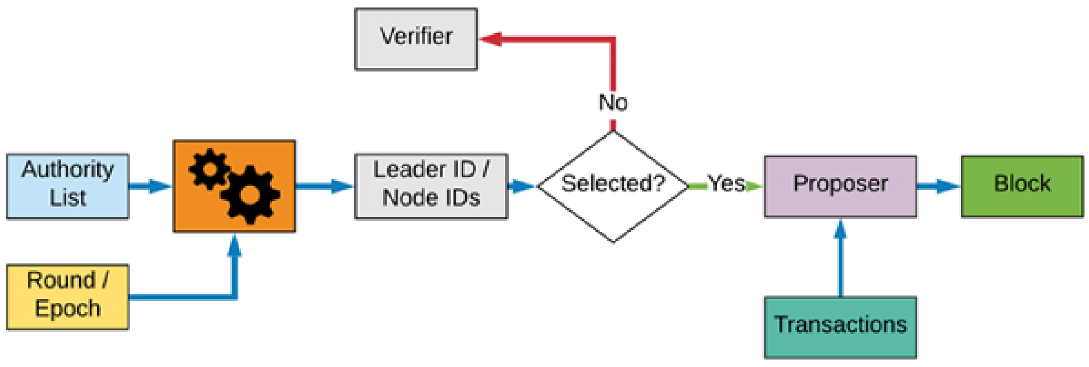


Figure 2.7: Proof of authority mechanism

2.3.8 On-chain and Off-chain

On chain blockchain refers inside the network where the network participants are the validators of settlement. The validators recheck the block through the smart contract where the company and insurance policy are assigned. Transaction occurs both on chain and off chain . A transaction is only valid once when all participants (only few are chosen randomly through PoA mechanism) have verified it and reached an agreement on its validity. The transaction is recorded at the block and dispatched to the blockchain. Once a transaction gets sufficient confirmations from network individuals primarily based totally at the network's consensus mechanism, it will become nearly irreversible, relying at the proof of authority network protocol [21].

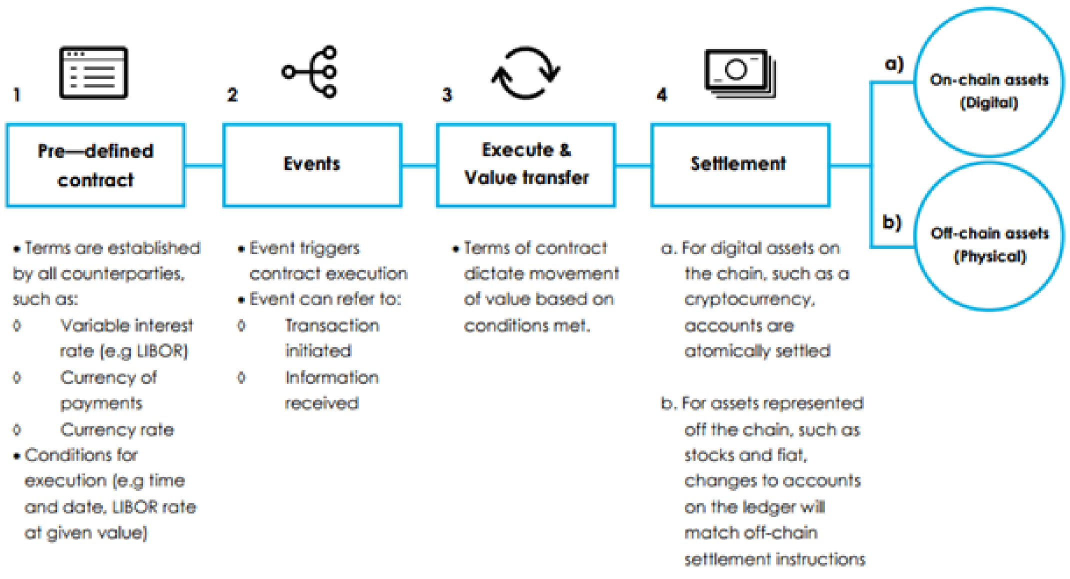

1
Pre—defined contract
• Terms are established by all counterparties, such as:
◊ Variable interest rate (e.g LIBOR)
◊ Currency of payments
◊ Currency rate
• Conditions for execution (e.g time and date, LIBOR rate at given value)
2
Events
• Event triggers contract execution
• Event can refer to:
◊ Transaction initiated
◊ Information received
3
Execute & Value transfer
• Terms of contract dictate movement of value based on conditions met.
4
Settlement
a. For digital assets on the chain, such as a cryptocurrency, accounts are atomically settled
b. For assets represented off the chain, such as stocks and fiat, changes to accounts on the ledger will match off-chain settlement instructions
a)
On-chain assets (Digital)
b)
Off-chain assets (Physical)

Figure 2.8: Use case diagram of chain system [6]

2.3.9 Smart Contract

For our system we will be using public smart contracts through EVM (Ethereum Virtual Machine). The EVM exists as a single entity maintained by thousands of connected computers running an Ethereum client, but its physical instantiation cannot be described in the same way that one might point to a cloud or an ocean wave. The client that we will use for creating the smart contract is geth. The Go version of Ethereum, known as Geth (go-Ethereum), serves as a portal to the decentralized web. Since the beginning of Ethereum, Geth has been a fundamental component. Geth is the most robust and tested client because it was one of the first Ethereum implementations. Geth is an Ethereum execution client, which means it manages transactions, smart contract deployment and execution, and has an Ethereum Virtual Machine embedded in it. The sole aim of the Ethereum protocol is to maintain the constant, unbroken, and immutable operation of this unique state machine. All Ethereum accounts and smart contracts are housed within it. Ethereum only has one ”canonical” state at any one point in the chain, and the EVM establishes the procedures for calculating a new valid state from block to block.

Chapter 3

System Architecture

3.1 Work Plan

At first we will collect necessary data from the client by which we can find the best possible insurance for his needs. After creating the Smart contract we will send it to the client to verify if he is pleased with that contract. After getting yes from them the smart contract will be deployed to the Blockchain network. And when the client makes a request in the future then the validators will run that contract and see if the request can be accepted or not if it does not clash with any clauses included in the contract. If the request gets approved from the validators when it will give off to the request and the client will receive what he requested. if the request goes against the contract then the validator will decline the request. This will be fair because the validator doesn't know who the client is and which insurance company will have to pay. Therefore the whole system will not be biased towards anyone. And because of the smart contract the contract can not be changed with others knowing.

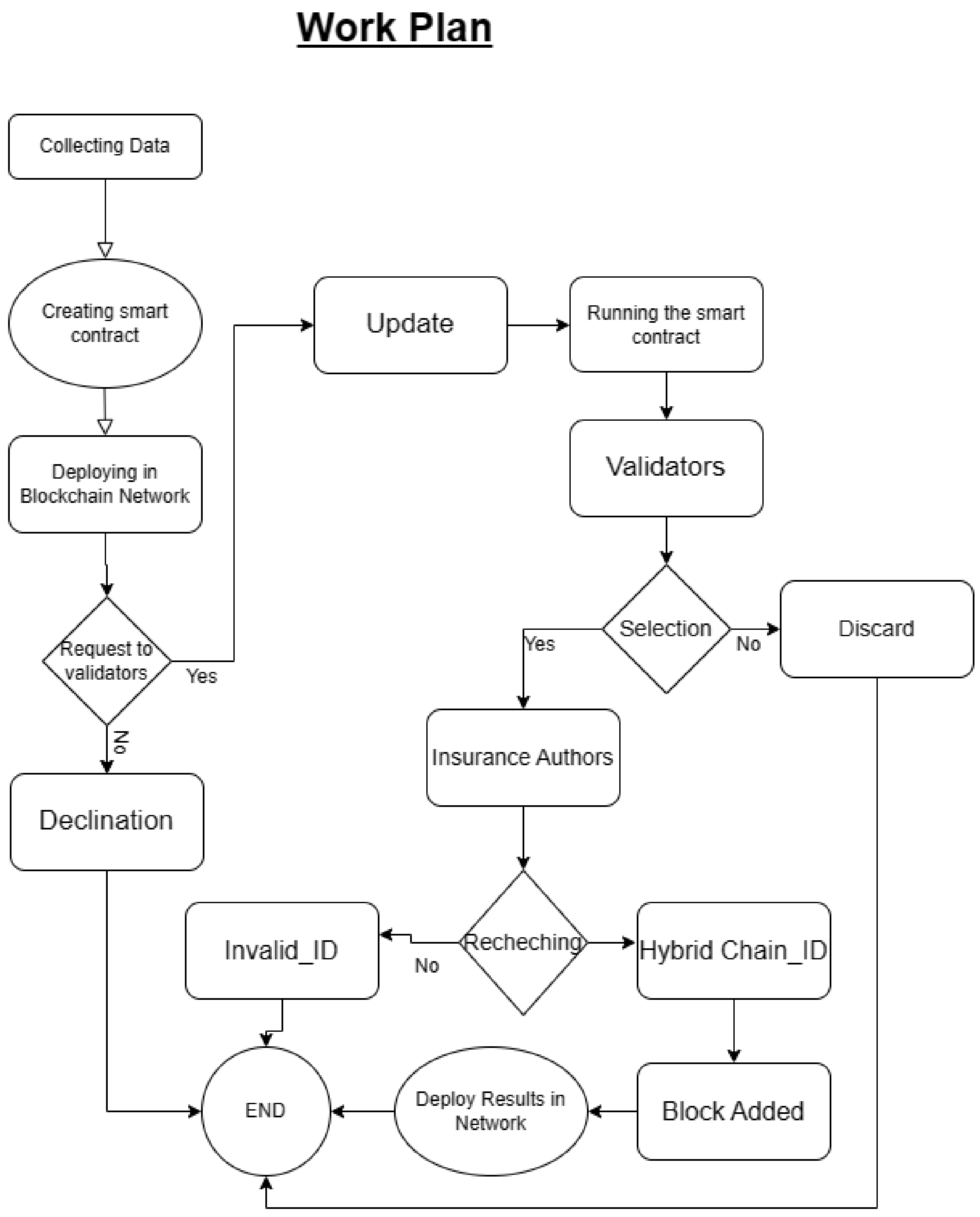


Figure 3.1: Work plan

3.2 Proposed Model

In our proposed Blockchain-based Advance Insurance system, the company's policy maker is the main authority. Here, Policy maker also gives authorities to clients(user) linking institution to join the network so that they issue their(clients) credentials to the system through the website for insurance registration. The Company Authorized Validators owns the smart contract. Here the authorized validators are the client agents. The smart contract consists of different function written in solidity language. Some of the functions that the smart contract will provide for the system are mentioned below :

1. Function policy (); //this function will provide insurance duration, insurance types and terms of company rules.
2. Function credentialAuthentication (); //this function allows company employees to validate credentials for making the insurance.
3. Function identityAuthentication (); //this function identifies the insurance holder using the public and private keys.
4. Function policyChecker (); // this function upholds the credentials of clients checking with the policy brief function.
5. Function insuranceValidator (); // validates the insurance using public key of the holder.
6. Function renewal();
7. Function insurance-eligibility();

3.3 Proposed Architecture

3.3.1 Insurance system application

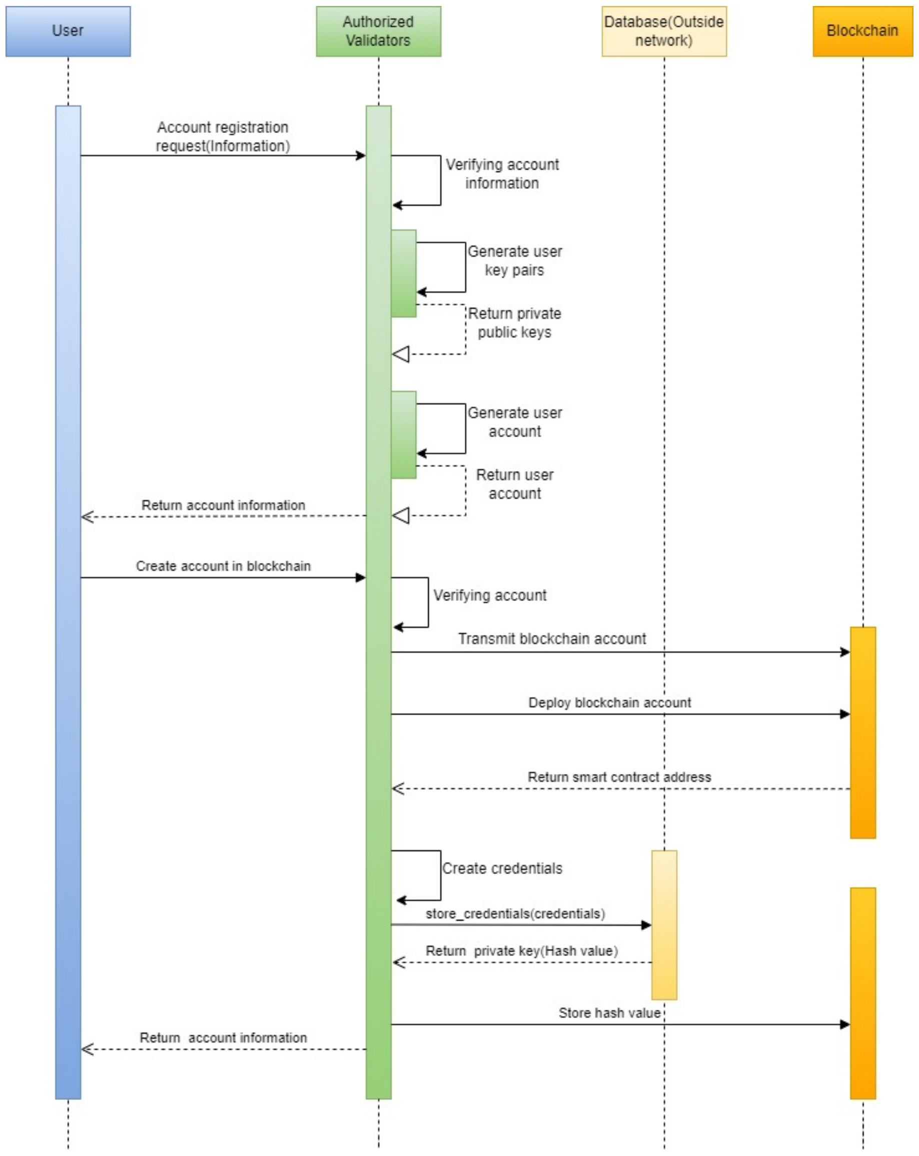

User
Authorized Validators
Database(Outside network)
Blockchain
Account registration request(Information)
Verifying account information
Generate user key pairs
Return private public keys
Generate user account
Return user account
Return account information
Create account in blockchain
Verifying account
Transmit blockchain account
Deploy blockchain account
Return smart contract address
Create credentials
store_credentials(credentials)
Return private key(Hash value)
Store hash value
Return account information

Figure 3.2: Architecture of Ethereum Blockchain based Insurance system application

After giving necessary information the employee will go through them and after confirming that info they will use the mined block and give the private key to the user and generate an account by which the users can make requests in the future. Using that account the user taking the insurance worker's help will make a smart contract and after rechecking it the contract will be deployed to the private BlockChain . The address of the block will be provided and that will be included in the users 1st account which was created previously. The main data Storage will include the users unique information and the hash of the blocks which has the smart contract.

3.3.2 Insurance system Policy

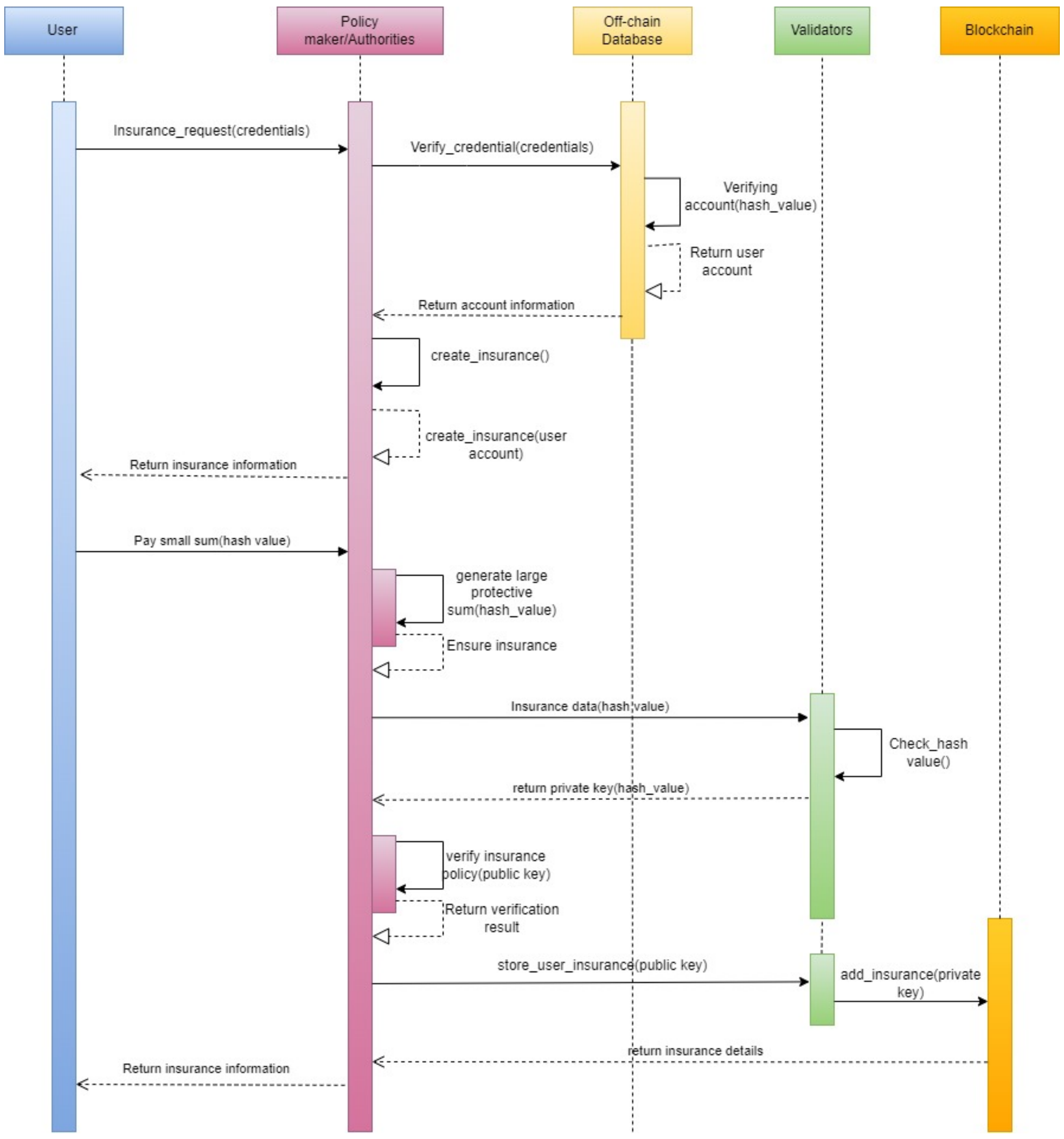

User
Policy maker/Authorities
Off-chain Database
Validators
Blockchain
Insurance_request(credentials)
Verify_credential(credentials)
Verifying account(hash_value)
Return user account
Return account information
create_insurance()
create_insurance(user account)
Return insurance information
Pay small sum(hash value)
generate large protective sum(hash_value)
Ensure insurance
Insurance data(hash value)
Check_hash value()
return private key(hash_value)
verify insurance policy(public key)
Return verification result
store_user_insurance(public key)
add_insurance(private key)
return insurance details
Return insurance information

Figure 3.3: Architecture of Ethereum Blockchain based Insurance system Policy

This here shows the steps which will take place after the insurance request will be send the use who will get the insurance. after agreeeing on the terms and rules the policy maker will send the data to validators who will accept the block and make the block a part of the private BlockChain.he will only have access to the blocks public key the insurance holder will have both public and private key. Which will insure the privacy of the block and the users insurance.

3.3.3 Insurance system renewal

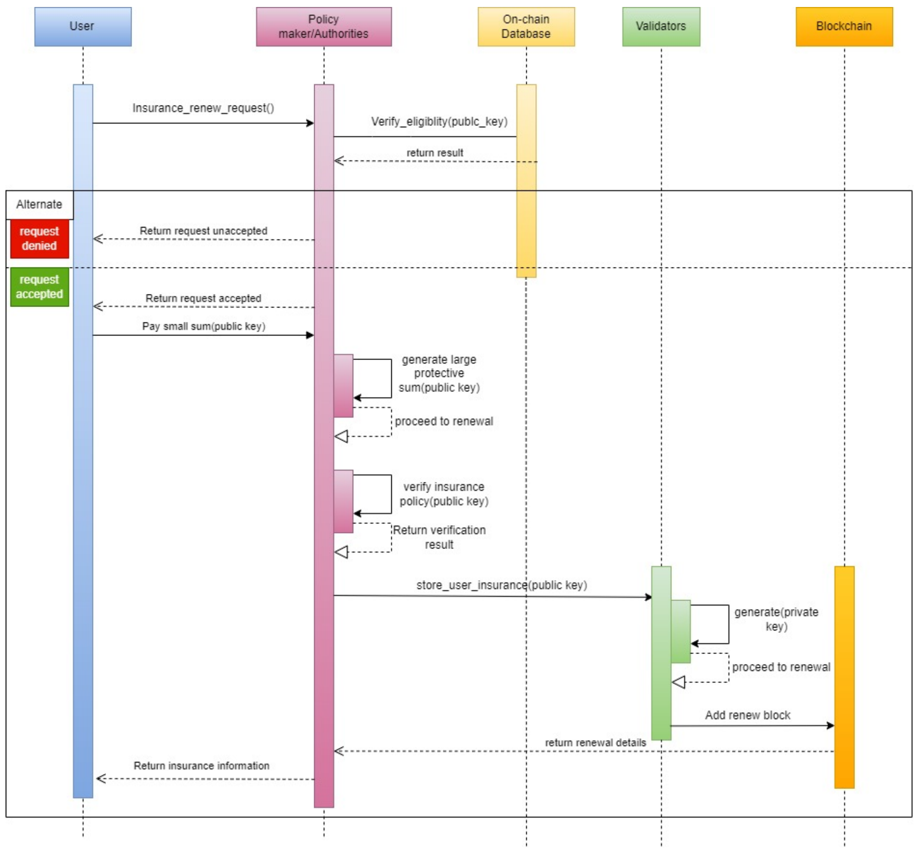

User
Policy maker/Authorities
On-chain Database
Validators
Blockchain
Insurance_renew_request()
Verify_eligiblity(publc_key)
return result
Alternate
request denied
Return request unaccepted
request accepted
Return request accepted
Pay small sum(public key)
generate large protective sum(public key)
proceed to renewal
verify insurance policy(public key)
Return verification result
store_user_insurance(public key)
generate(private key)
proceed to renewal
Add renew block
return renewal details
Return insurance information

Figure 3.4: Architecture of Ethereum Blockchain based Insurance system renewal

This will be done in 2 Steps:

1. First, the user makes a renewal request to the authorities, who are also the company's policymakers and are picked at random across the network. The function eligibility receives the request using the user's public key. The insurance is then verified using the insurance block's timestamp, and the findings are returned to the client. This phase is carried out in the on-chain database.

2. Furthermore, the results reflect whether or not the user is qualified for renewal. The function renewal defines the new transactions here. The client can now receive cryptocurrency transactions thanks to the public key. Following the generation of the renewal, it is a cryptographic code that is matched with a private key by network validators. While authorities can transmit transactions to the public key, the private key is required to "unlock" them and establish that the owner of the transaction matches the client's public key. Finally, the insurance data is returned, and the renewal transaction is added to the blockchain by creating a new block through the authorities (miners).

3.3.4 Use Case of the system

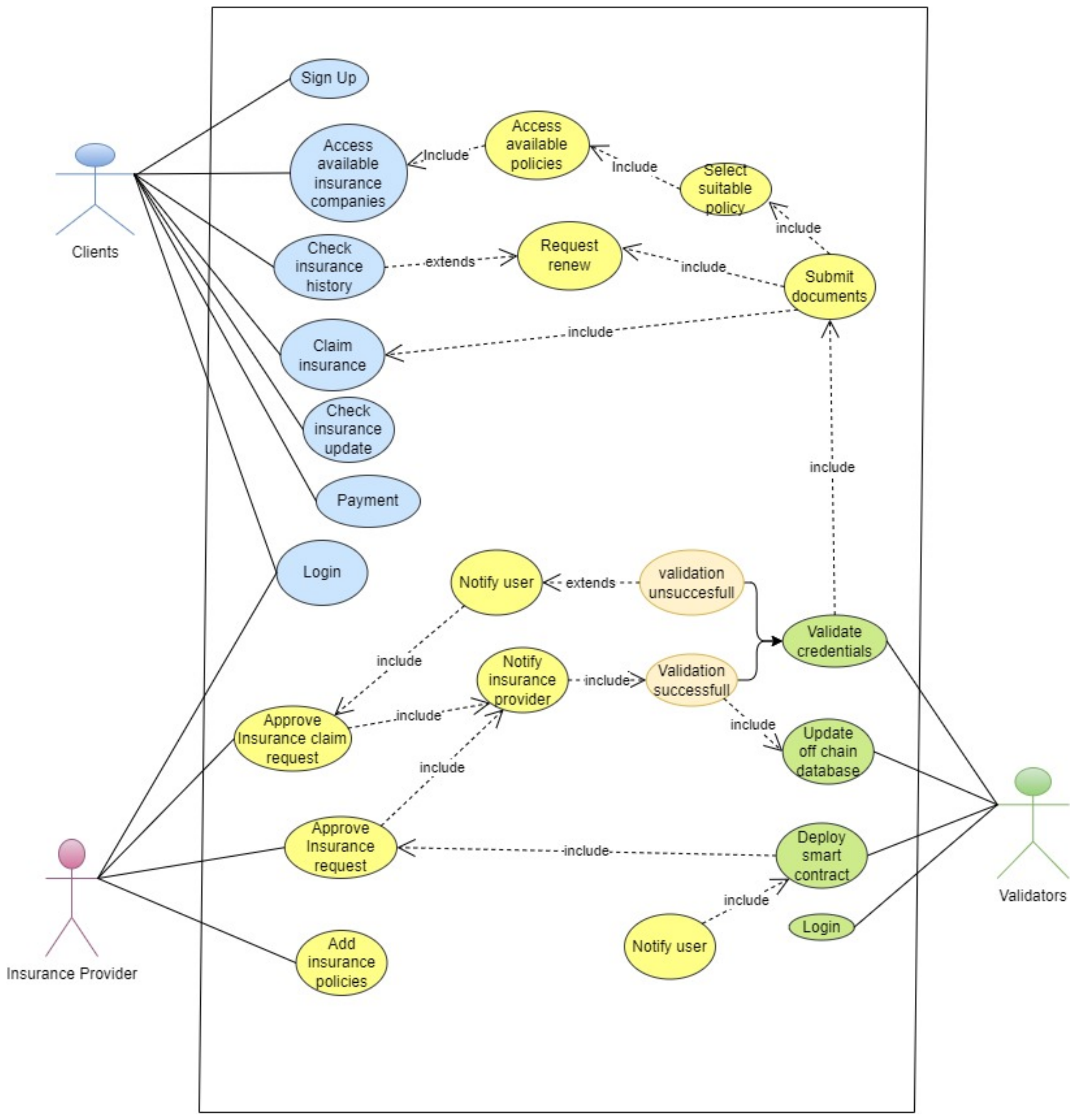


Figure 3.5: Use Case of Ethereum Blockchain based Health Insurance system

The interaction between different user categories and the system is shown in the diagram above. What users can do within the system is demonstrated by how they interact with it as a whole. The main features of the diagram are how insurance requests must be sent and how they are processed by the system in conjunction with users, a validator, and an insurance service provider.

3.3.5 User activity of the system

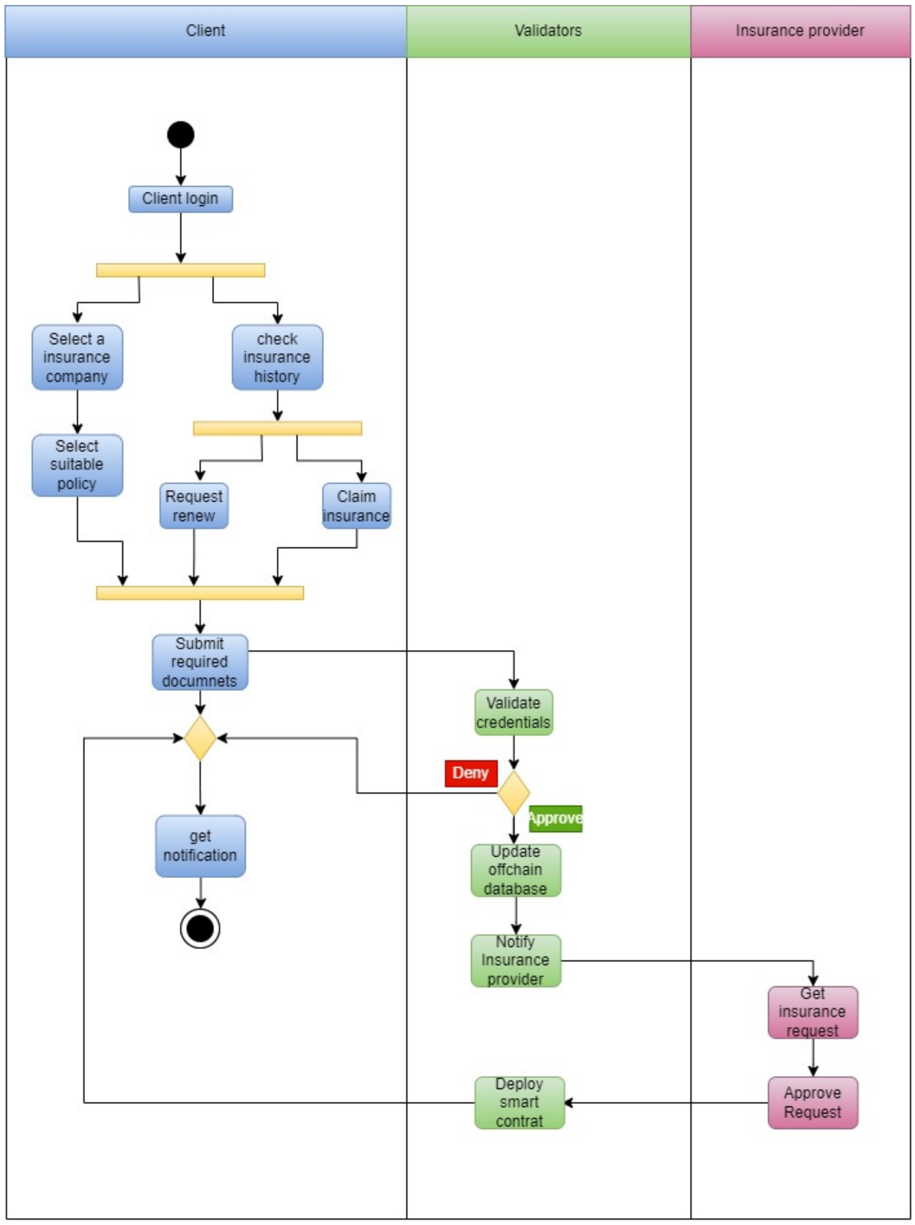


Figure 3.6: User activity of Ethereum Blockchain based Health Insurance system

This diagram 3.6 shows the workflow of the system, which is based on the user's activity. The main focus here is on how the system process runs for various action types, beginning with user log-in. It demonstrates how the system will respond to any user activity. Initially, a user appears to have the option of selecting any suitable plan for him or renewing any of his existing insurance services. After providing all required papers, the user's activity is initially paused, and the validator's results will have an impact on the insurance provider's decision. When the data has been processed, the user will be notified.

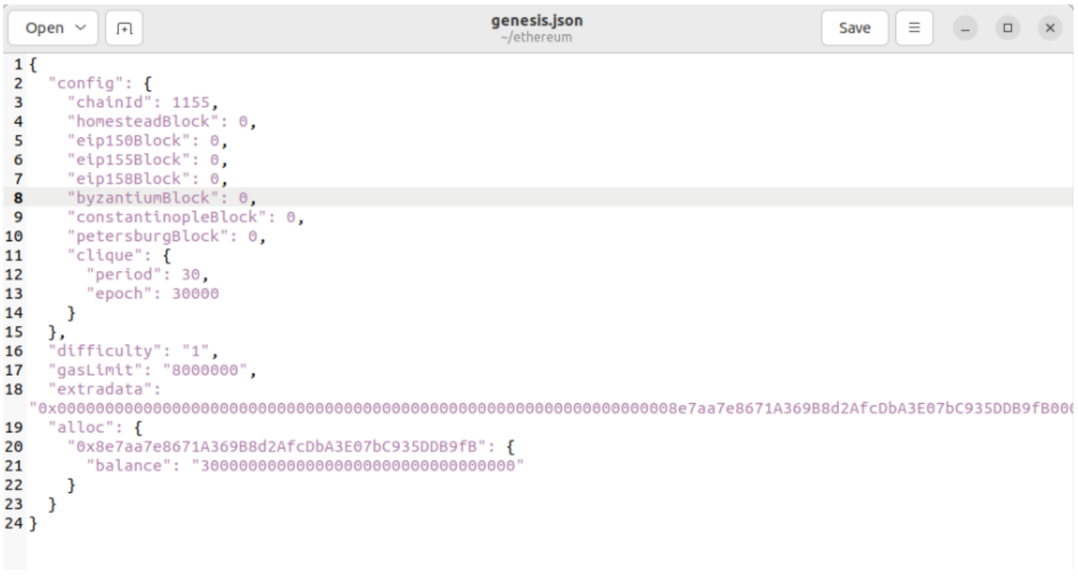
Chapter 4

Implementation

For our implementation we have used Ganache and Go-Ethereum both for analysis which are the local private blockchain network environment in our machine.

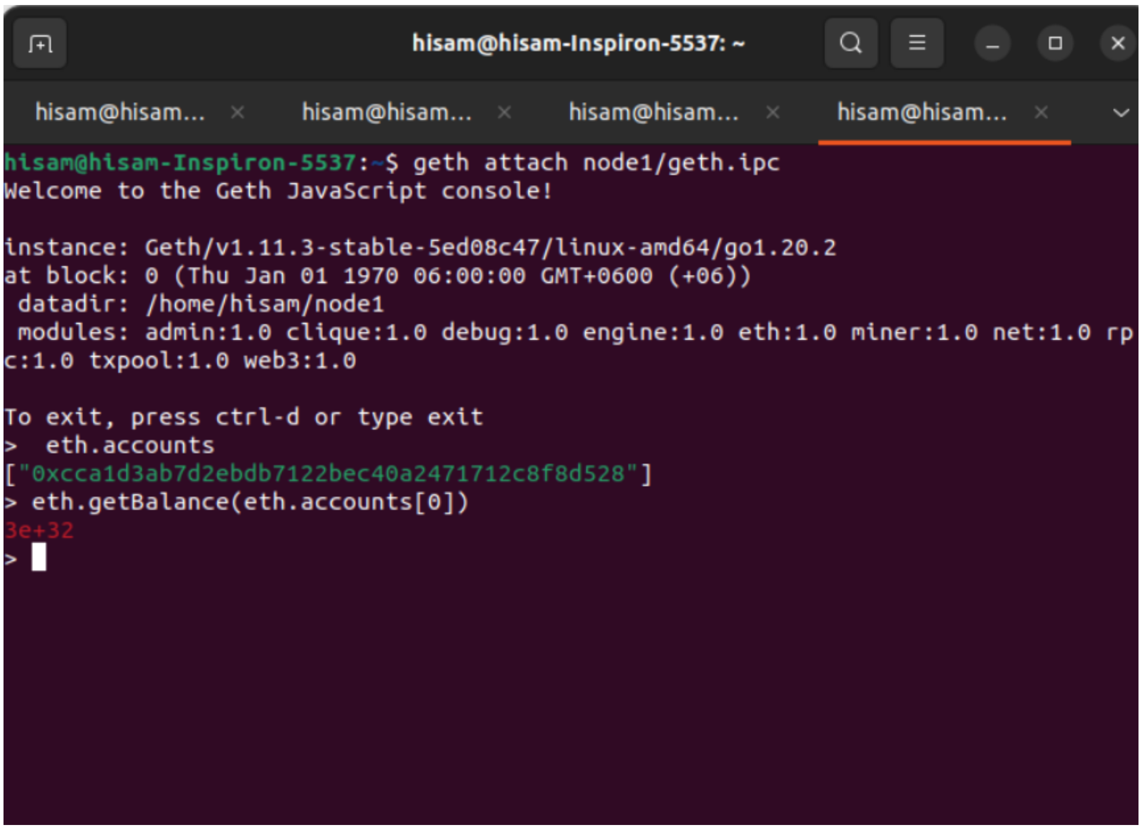
4.1 Using Geth Client

Linux Ubuntu version 23.0.1 is more effective and suitable for ethereum, so we have taken the first step by installing Go-Ethereum where we will be the development environment setup. After that, setting up the Ethereum Private network by creating two directories for two nodes in a root folder named ethereumNetwork.



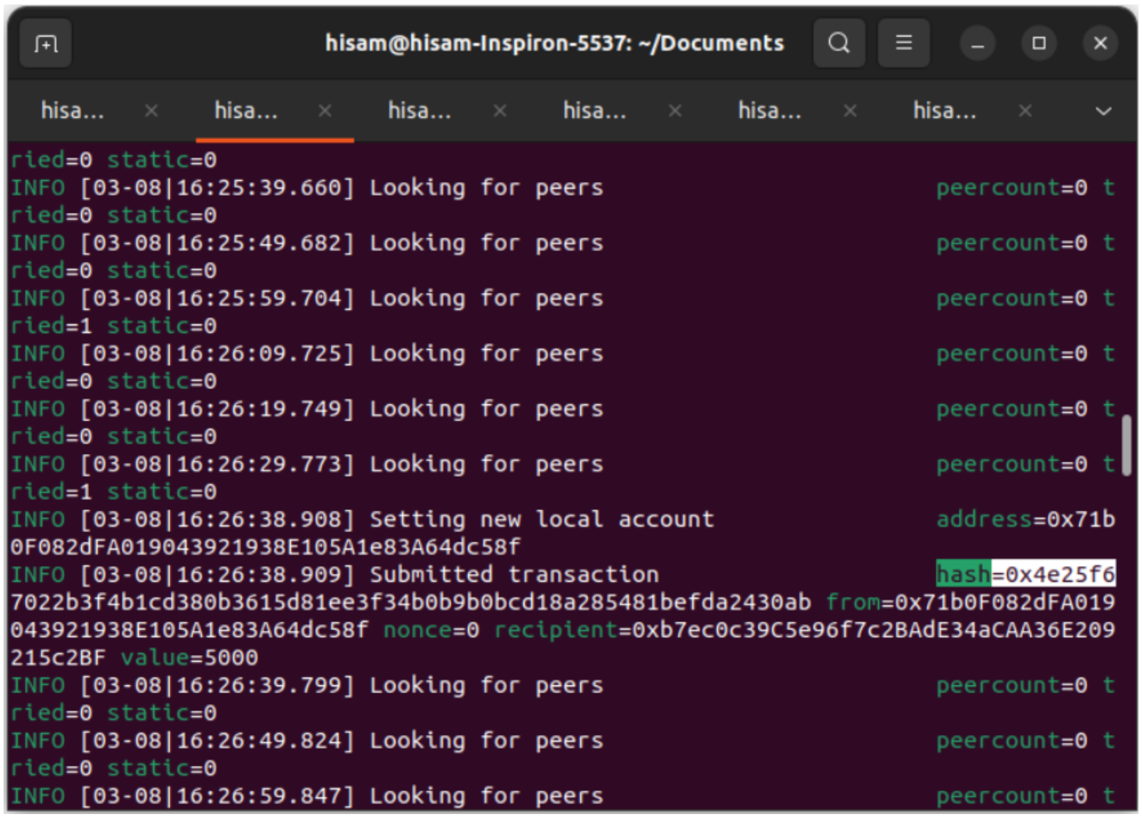
```json
{
  "config": {
    "chainId": 1155,
    "homesteadBlock": 0,
    "eip150Block": 0,
    "eip155Block": 0,
    "eip158Block": 0,
    "byzantiumBlock": 0,
    "constantinopleBlock": 0,
    "petersburgBlock": 0,
    "clique": {
      "period": 30,
      "epoch": 30000
    }
  },
  "difficulty": "1",
  "gasLimit": "8000000",
  "extradata":
"0x00000000000000000000000000000000000000000000000000000000000000008e7aa7e8671A369B8d2AfcDbA3E07bC935DDB9fB00
  "alloc": {
    "0x8e7aa7e8671A369B8d2AfcDbA3E07bC935DDB9fB": {
      "balance": "300000000000000000000000000000000"
    }
  }
}
```

Figure 4.1: Code of genesisblock.json



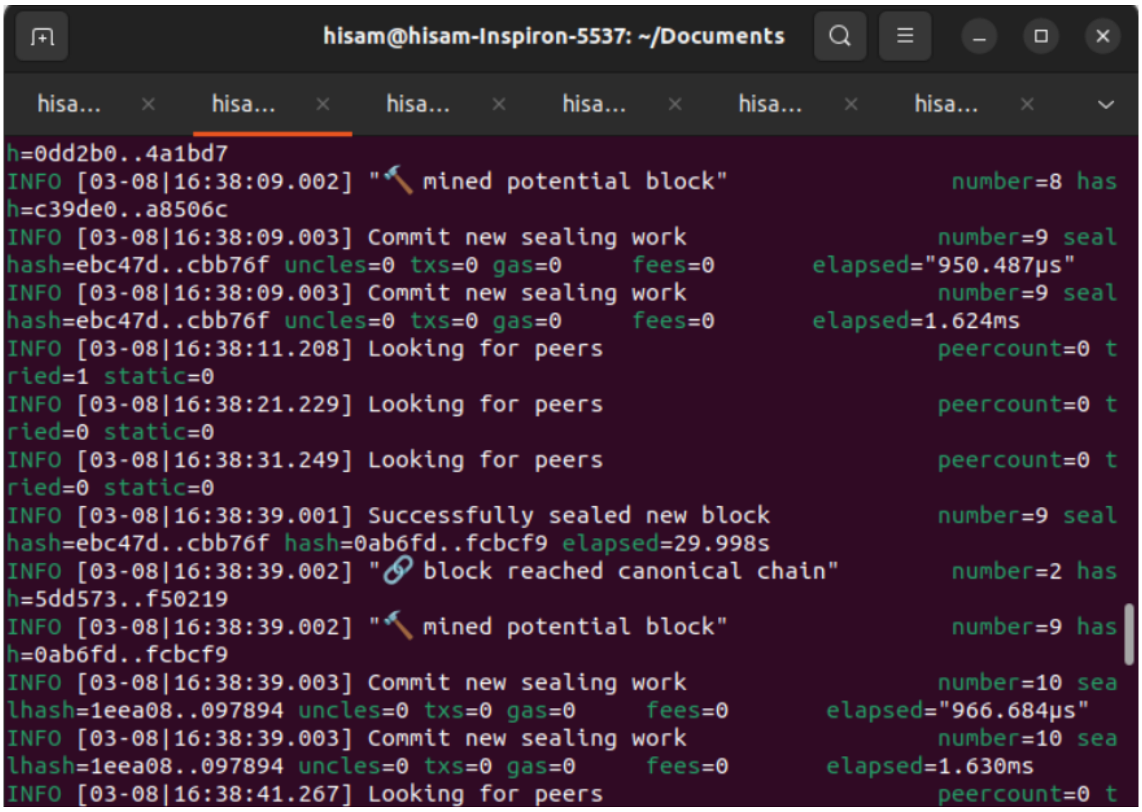
```
hisam@hisam-Inspiron-5537:~$ geth attach node1/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.11.3-stable-5ed08c47/linux-amd64/go1.20.2
at block: 0 (Thu Jan 01 1970 06:00:00 GMT+0600 (+06))
 datadir: /home/hisam/node1
 modules: admin:1.0 clique:1.0 debug:1.0 engine:1.0 eth:1.0 miner:1.0 net:1.0 rp
c:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
>  eth.accounts
["0xcca1d3ab7d2ebdb7122bec40a2471712c8f8d528"]
> eth.getBalance(eth.accounts[0])
3e+32
> 
```

Figure 4.2: Creating ethereum account and connecting with its genesis block



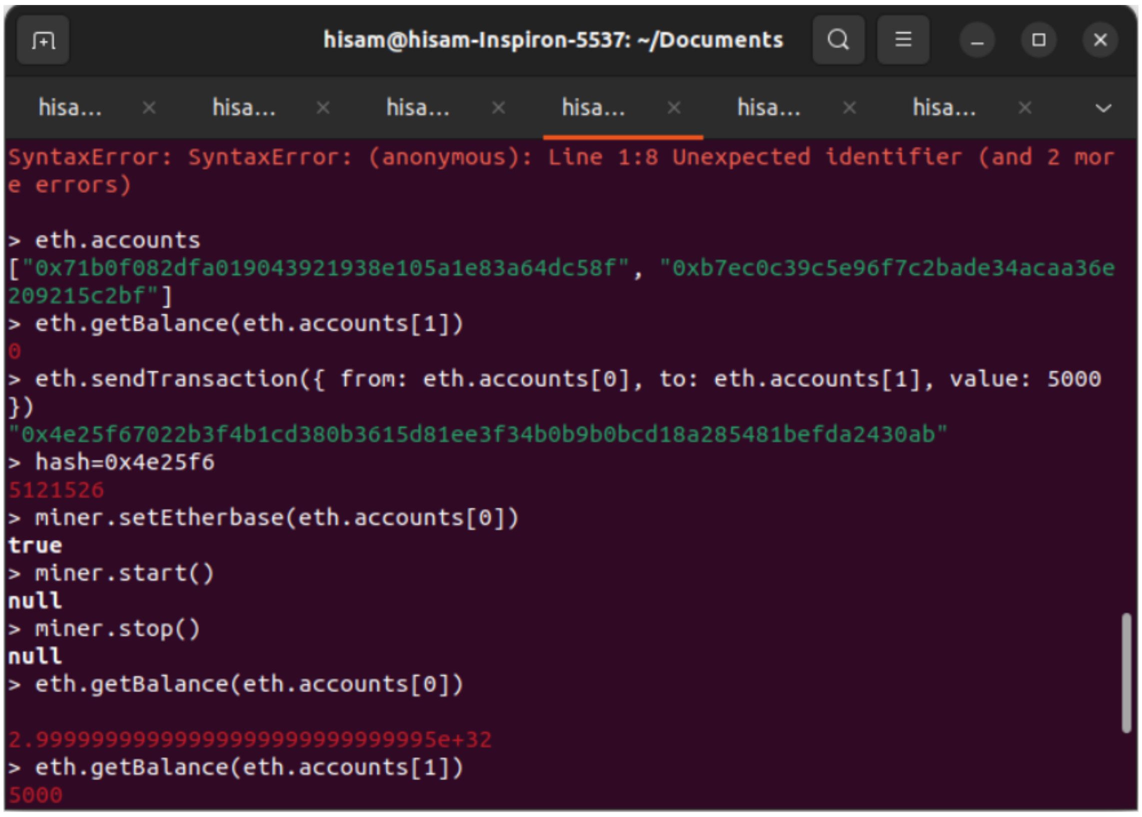
```
ried=0 static=0
INFO [03-08|16:25:39.660] Looking for peers                        peercount=0 t
ried=0 static=0
INFO [03-08|16:25:49.682] Looking for peers                        peercount=0 t
ried=0 static=0
INFO [03-08|16:25:59.704] Looking for peers                        peercount=0 t
ried=1 static=0
INFO [03-08|16:26:09.725] Looking for peers                        peercount=0 t
ried=0 static=0
INFO [03-08|16:26:19.749] Looking for peers                        peercount=0 t
ried=0 static=0
INFO [03-08|16:26:29.773] Looking for peers                        peercount=0 t
ried=1 static=0
INFO [03-08|16:26:38.908] Setting new local account                address=0x71b
0F082dFA019043921938E105A1e83A64dc58f
INFO [03-08|16:26:38.909] Submitted transaction                    hash=0x4e25f6
7022b3f4b1cd380b3615d81ee3f34b0b9b0bcd18a285481befda2430ab from=0x71b0F082dFA019
043921938E105A1e83A64dc58f nonce=0 recipient=0xb7ec0c39C5e96f7c2BAdE34aCAA36E209
215c2BF value=5000
INFO [03-08|16:26:39.799] Looking for peers                        peercount=0 t
ried=0 static=0
INFO [03-08|16:26:49.824] Looking for peers                        peercount=0 t
ried=0 static=0
INFO [03-08|16:26:59.847] Looking for peers                        peercount=0 t
```

Figure 4.3: Setting new local account to the network



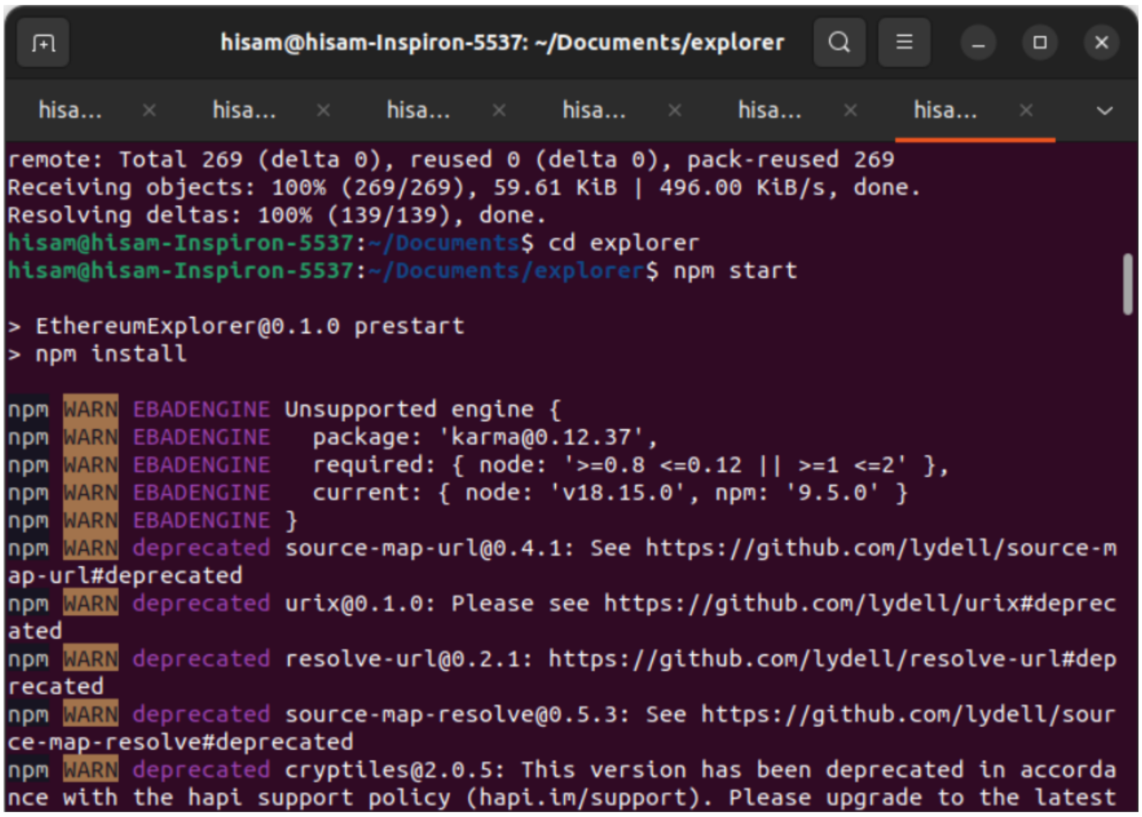
```
h=0dd2b0..4a1bd7
INFO [03-08|16:38:09.002] "🔨 mined potential block"                  number=8 has
h=c39de0..a8506c
INFO [03-08|16:38:09.003] Commit new sealing work                    number=9 seal
hash=ebc47d..cbb76f uncles=0 txs=0 gas=0      fees=0       elapsed="950.487μs"
INFO [03-08|16:38:09.003] Commit new sealing work                    number=9 seal
hash=ebc47d..cbb76f uncles=0 txs=0 gas=0      fees=0       elapsed=1.624ms
INFO [03-08|16:38:11.208] Looking for peers                          peercount=0 t
ried=1 static=0
INFO [03-08|16:38:21.229] Looking for peers                          peercount=0 t
ried=0 static=0
INFO [03-08|16:38:31.249] Looking for peers                          peercount=0 t
ried=0 static=0
INFO [03-08|16:38:39.001] Successfully sealed new block              number=9 seal
hash=ebc47d..cbb76f hash=0ab6fd..fcbcf9 elapsed=29.998s
INFO [03-08|16:38:39.002] "🔗 block reached canonical chain"          number=2 has
h=5dd573..f50219
INFO [03-08|16:38:39.002] "🔨 mined potential block"                  number=9 has
h=0ab6fd..fcbcf9
INFO [03-08|16:38:39.003] Commit new sealing work                    number=10 sea
lhash=1eea08..097894 uncles=0 txs=0 gas=0      fees=0       elapsed="966.684μs"
INFO [03-08|16:38:39.003] Commit new sealing work                    number=10 sea
lhash=1eea08..097894 uncles=0 txs=0 gas=0      fees=0       elapsed=1.630ms
INFO [03-08|16:38:41.267] Looking for peers                          peercount=0 t
```

Figure 4.4: Mining new block to the blockchain network



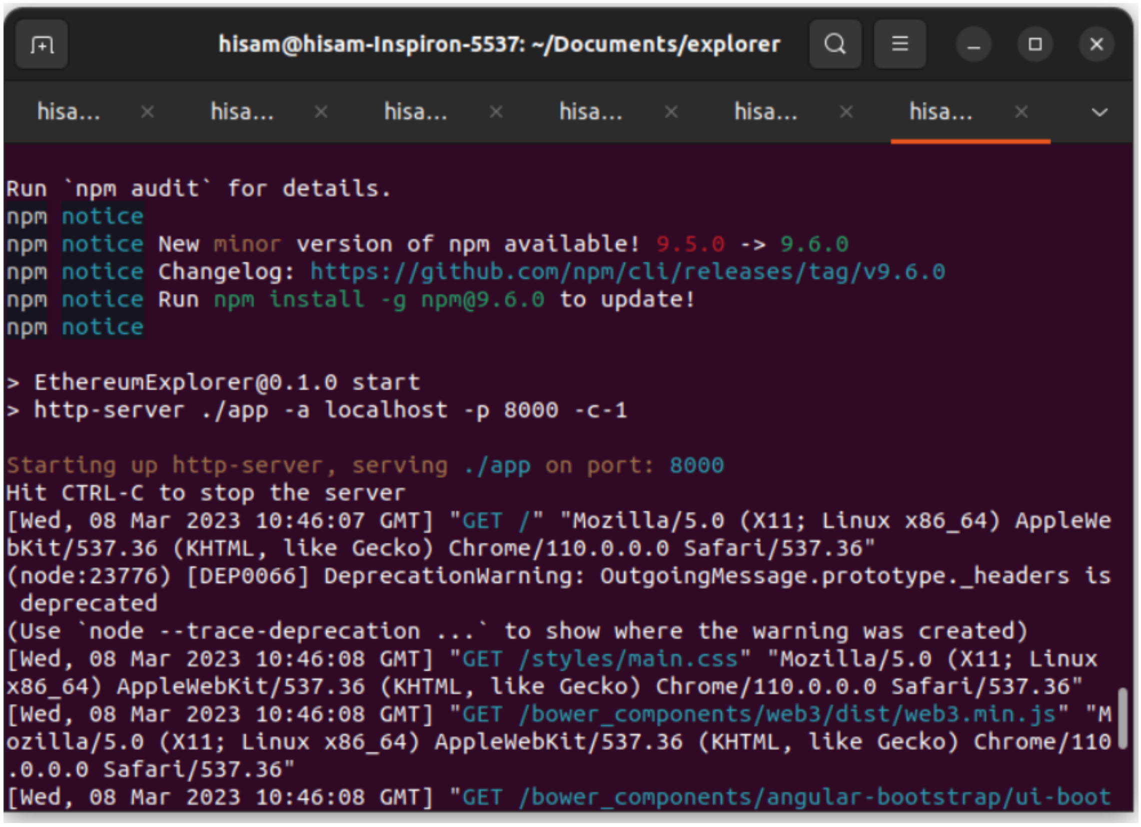
```
SyntaxError: SyntaxError: (anonymous): Line 1:8 Unexpected identifier (and 2 mor
e errors)

> eth.accounts
["0x71b0f082dfa019043921938e105a1e83a64dc58f", "0xb7ec0c39c5e96f7c2bade34acaa36e
209215c2bf"]
> eth.getBalance(eth.accounts[1])
0
> eth.sendTransaction({ from: eth.accounts[0], to: eth.accounts[1], value: 5000
})
"0x4e25f67022b3f4b1cd380b3615d81ee3f34b0b9b0bcd18a285481befda2430ab"
> hash=0x4e25f6
5121526
> miner.setEtherbase(eth.accounts[0])
true
> miner.start()
null
> miner.stop()
null
> eth.getBalance(eth.accounts[0])

2.99999999999999999999999999995e+32
> eth.getBalance(eth.accounts[1])
5000
```

Figure 4.5: Checking the hash value and account details



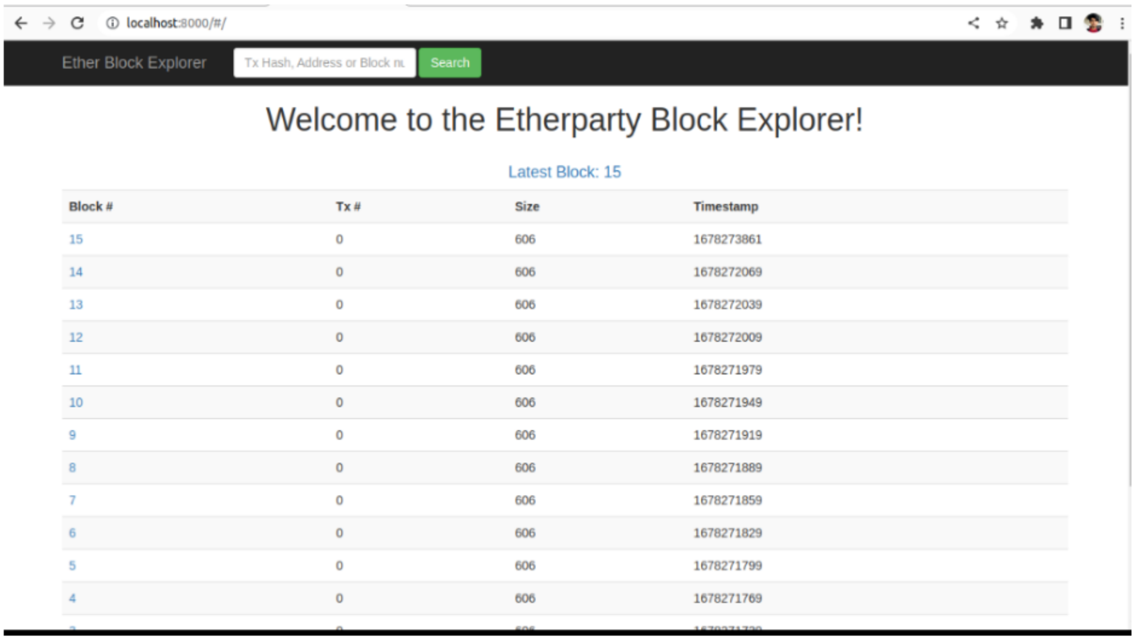
```
remote: Total 269 (delta 0), reused 0 (delta 0), pack-reused 269
Receiving objects: 100% (269/269), 59.61 KiB | 496.00 KiB/s, done.
Resolving deltas: 100% (139/139), done.
hisam@hisam-Inspiron-5537:~/Documents$ cd explorer
hisam@hisam-Inspiron-5537:~/Documents/explorer$ npm start

> EthereumExplorer@0.1.0 prestart
> npm install

npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'karma@0.12.37',
npm WARN EBADENGINE   required: { node: '>=0.8 <=0.12 || >=1 <=2' },
npm WARN EBADENGINE   current: { node: 'v18.15.0', npm: '9.5.0' }
npm WARN EBADENGINE }
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-m
ap-url#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprec
ated
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#dep
recated
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/sour
ce-map-resolve#deprecated
npm WARN deprecated cryptiles@2.0.5: This version has been deprecated in accorda
nce with the hapi support policy (hapi.im/support). Please upgrade to the latest
```

Figure 4.6: Running the node through the browser installing npm



```
Run `npm audit` for details.
npm notice
npm notice New minor version of npm available! 9.5.0 -> 9.6.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v9.6.0
npm notice Run npm install -g npm@9.6.0 to update!
npm notice

> EthereumExplorer@0.1.0 start
> http-server ./app -a localhost -p 8000 -c-1

Starting up http-server, serving ./app on port: 8000
Hit CTRL-C to stop the server
[Wed, 08 Mar 2023 10:46:07 GMT] "GET /" "Mozilla/5.0 (X11; Linux x86_64) AppleWe
bKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36"
(node:23776) [DEP0066] DeprecationWarning: OutgoingMessage.prototype._headers is
 deprecated
(Use `node --trace-deprecation ...` to show where the warning was created)
[Wed, 08 Mar 2023 10:46:08 GMT] "GET /styles/main.css" "Mozilla/5.0 (X11; Linux
x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36"
[Wed, 08 Mar 2023 10:46:08 GMT] "GET /bower_components/web3/dist/web3.min.js" "M
ozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110
.0.0.0 Safari/537.36"
[Wed, 08 Mar 2023 10:46:08 GMT] "GET /bower_components/angular-bootstrap/ui-boot
```

Figure 4.7: Connecting to the http local server with the help of EthereumExplorer



Welcome to the Etherparty Block Explorer!

Latest Block: 15

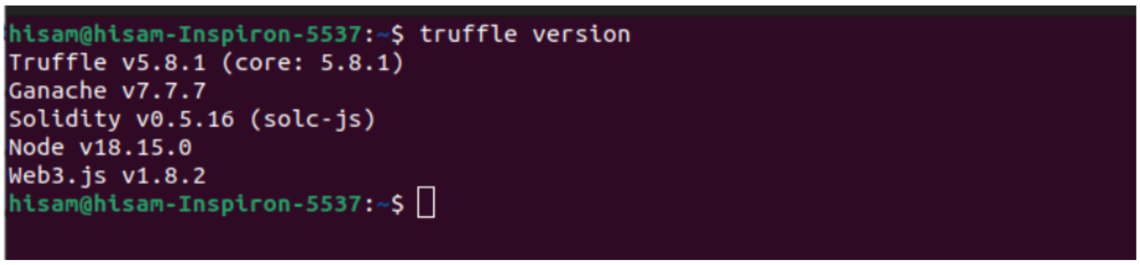
Block #	Tx #	Size	Timestamp
15	0	606	1678273861
14	0	606	1678272069
13	0	606	1678272039
12	0	606	1678272009
11	0	606	1678271979
10	0	606	1678271949
9	0	606	1678271919
8	0	606	1678271889
7	0	606	1678271859
6	0	606	1678271829
5	0	606	1678271799
4	0	606	1678271769

Figure 4.8: Results of blocks mined through browser with accounts timestamp.

4.2 Using Ganache and ethereum wallet(Metamask)

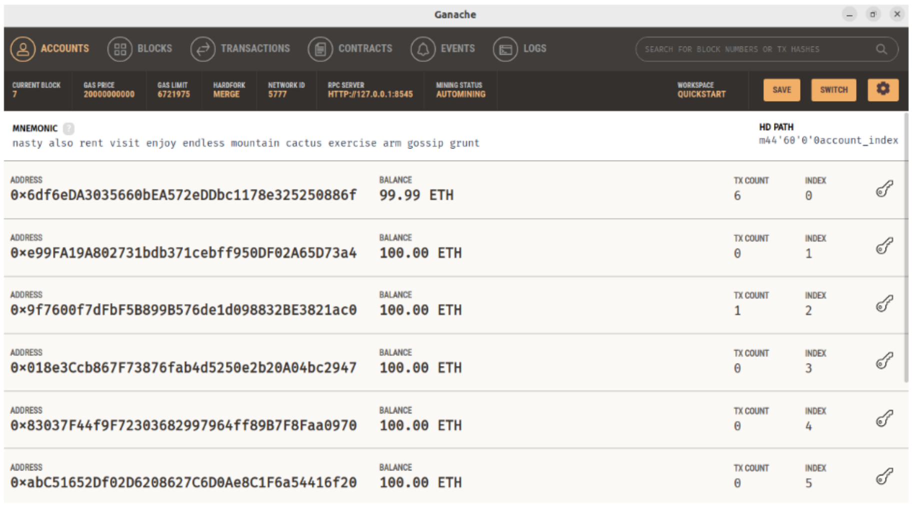
Firstly, for installing Ganache we need to install nvm or we will get an error. We used curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.2/install.sh — bash this cmd for nvm installation. As we are using the linux operating system for

our implementation we have run this command in our terminal main directory.
After successfully installing truffle, next we will install Ganache and Metamask in the web browser that is chrome for this time. Before installing Ganache we had to run this sudo add-apt-repository universe sudo apt install libfuse2 command in our terminal for creating the blockchain network files.
Now, connect the Ganache to the localhost server which we have used post 8585 with metamask. As we will be using two smart contracts for our implementation, we will create two files in the contracts directory by using this command touch contracts/Insurance.sol touch contracts/Health.sol. Finally for deployment configuration we have to create a migration file in the migration folder from where we will call for the deployment.
// requiring the contract
var [smartcontract$_n$ame] $= artifacts.require(”./smartcontract_filename.sol”);$
$//exportingasmodule$
$module.exports = function(deployer)deployer.deploy(smartcontract_name);;$
$Therefore, savingthiscodetoaseparatefileinthemigrationfolderwerunthecommandtrufflemigrat$



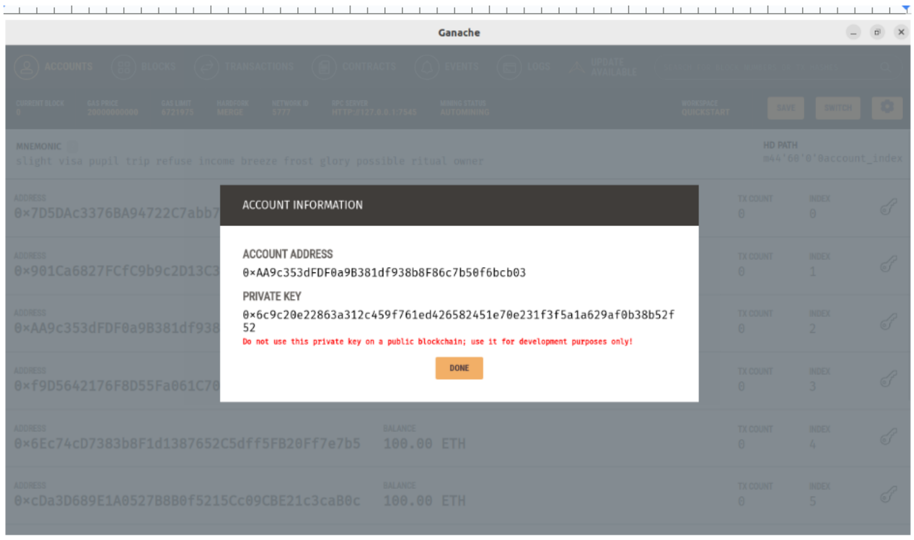
```
hisam@hisam-Inspiron-5537:~$ truffle version
Truffle v5.8.1 (core: 5.8.1)
Ganache v7.7.7
Solidity v0.5.16 (solc-js)
Node v18.15.0
Web3.js v1.8.2
hisam@hisam-Inspiron-5537:~$ 
```

Figure 4.9: Results of blocks mined through browser with accounts timestamp.



```
CURRENT BLOCK  GAS PRICE    GAS LIMIT  HARDFORK  NETWORK ID  RPC SERVER              MINING STATUS   WORKSPACE
7              20000000000  6721975    MERGE     5777        HTTP://127.0.0.1:8545   AUTOMINING      QUICKSTART

MNEMONIC                                                                              HD PATH
nasty also rent visit enjoy endless mountain cactus exercise arm gossip grunt         m44'60'0'0account_index

ADDRESS                                       BALANCE      TX COUNT  INDEX
0×6df6eDA3035660bEA572eDDbc1178e325250886f    99.99 ETH    6         0

ADDRESS                                       BALANCE      TX COUNT  INDEX
0×e99FA19A802731bdb371cebff950DF02A65D73a4    100.00 ETH   0         1

ADDRESS                                       BALANCE      TX COUNT  INDEX
0×9f7600f7dFbF5B899B576de1d098832BE3821ac0    100.00 ETH   1         2

ADDRESS                                       BALANCE      TX COUNT  INDEX
0×018e3Ccb867F73876fab4d5250e2b20A04bc2947    100.00 ETH   0         3

ADDRESS                                       BALANCE      TX COUNT  INDEX
0×83037F44f9F72303682997964ff89B7F8Faa0970    100.00 ETH   0         4

ADDRESS                                       BALANCE      TX COUNT  INDEX
0×abC51652Df02D6208627C6D0Ae8C1F6a54416f20    100.00 ETH   0         5
```

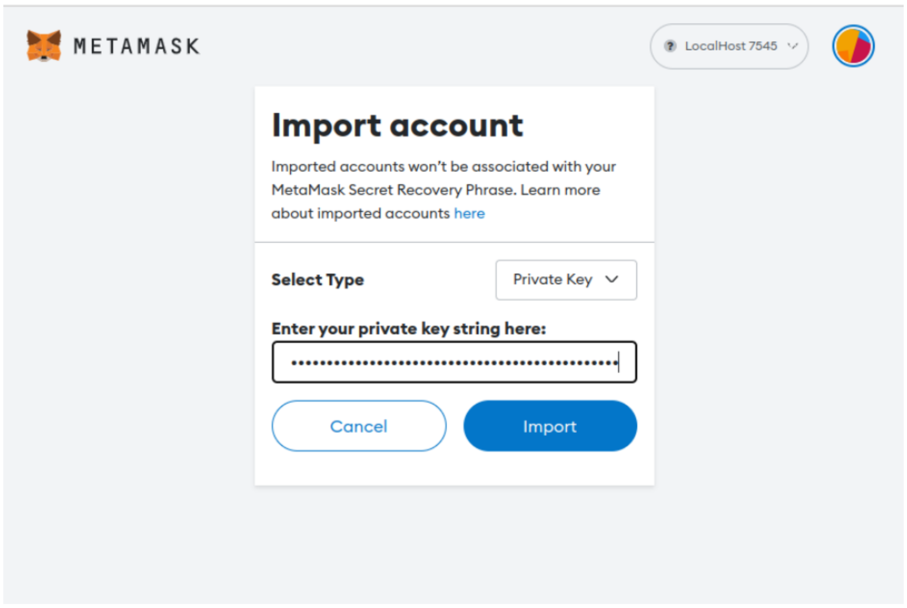
Figure 4.10: Ganache interface with TX count 6 for peer connection.


Ganache

ACCOUNT INFORMATION

ACCOUNT ADDRESS
```
0×AA9c353dFDF0a9B381df938b8F86c7b50f6bcb03
```
PRIVATE KEY
```
0×6c9c20e22863a312c459f761ed426582451e70e231f3f5a1a629af0b38b52f
52
```
Do not use this private key on a public blockchain; use it for development purposes only!

Figure 4.11: Results of blocks mined through browser with accounts timestamp.



Import account

Imported accounts won't be associated with your MetaMask Secret Recovery Phrase. Learn more about imported accounts here

Select Type: Private Key

Enter your private key string here:

Figure 4.12: Importing Account to metamask for Smart Contract Deployment.

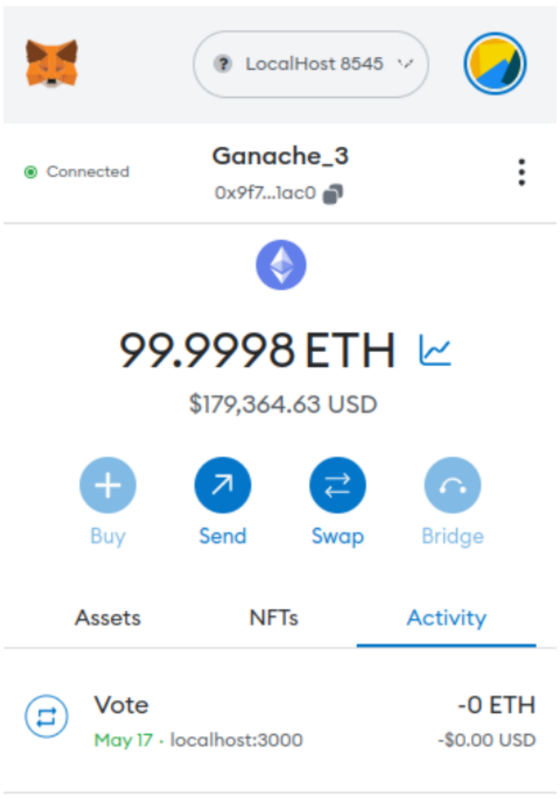
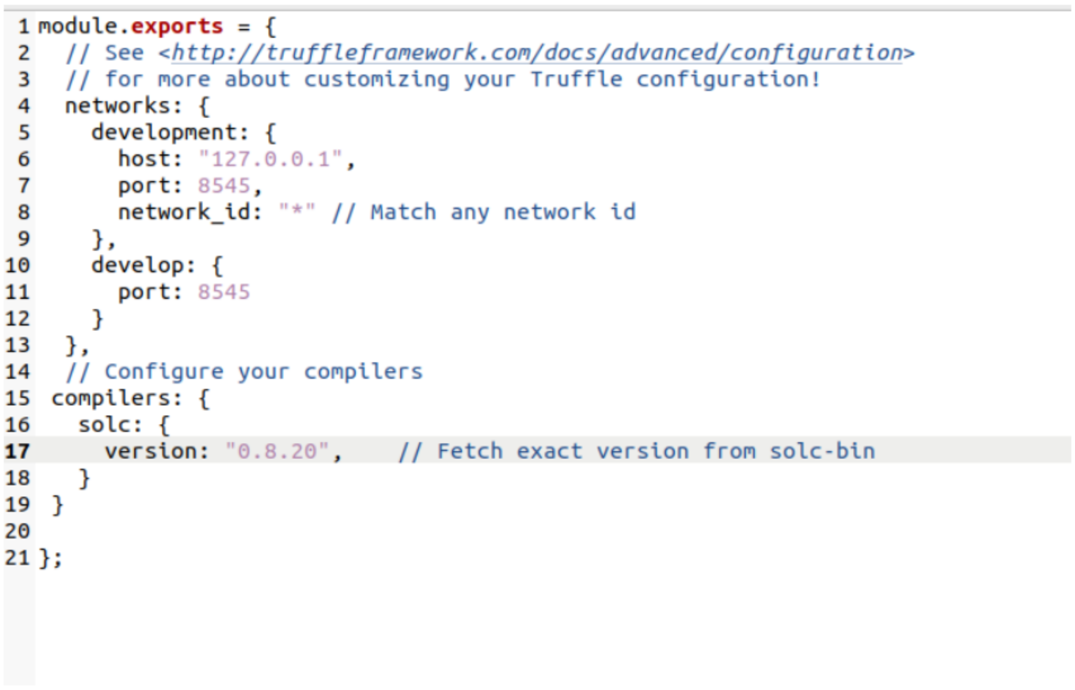
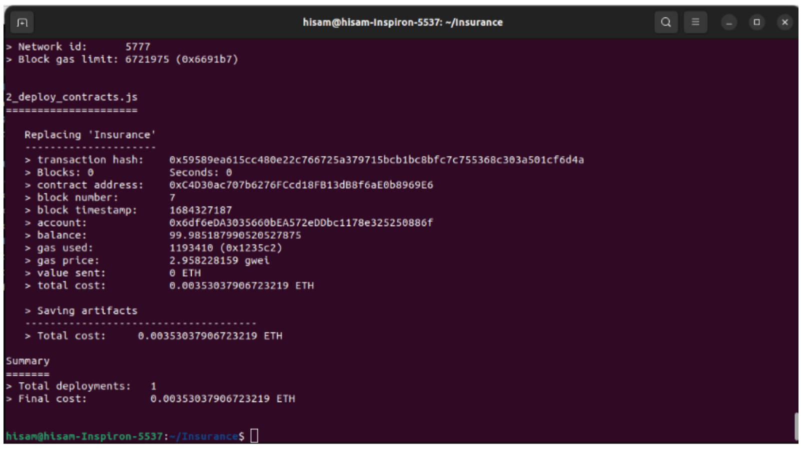


Figure 4.13: Metamask details about account and ether amount for transaction through the smart contracts.



```
module.exports = {
  // See <http://truffleframework.com/docs/advanced/configuration>
  // for more about customizing your Truffle configuration!
  networks: {
    development: {
      host: "127.0.0.1",
      port: 8545,
      network_id: "*" // Match any network id
    },
    develop: {
      port: 8545
    }
  },
  // Configure your compilers
 compilers: {
   solc: {
     version: "0.8.20",    // Fetch exact version from solc-bin
   }
 }

};
```

Figure 4.14: Migration file code for Smart contract Compilation and deployment.



```
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)


2_deploy_contracts.js
=====================

   Replacing 'Insurance'
   ---------------------
   > transaction hash:    0x59589ea615cc480e22c766725a379715bcb1bc8bfc7c755368c303a501cf6d4a
   > Blocks: 0            Seconds: 0
   > contract address:    0xC4D30ac707b6276FCcd18FB13dB8f6aE0b8969E6
   > block number:        7
   > block timestamp:     1684327187
   > account:             0x6df6eDA3035660bEA572eDDbc1178e325250886f
   > balance:             99.985187990520527875
   > gas used:            1193410 (0x1235c2)
   > gas price:           2.958228159 gwei
   > value sent:          0 ETH
   > total cost:          0.00353037906723219 ETH

   > Saving artifacts
   -------------------------------------
   > Total cost:     0.00353037906723219 ETH

Summary
=======
> Total deployments:   1
> Final cost:          0.00353037906723219 ETH


hisam@hisam-Inspiron-5537:~/Insurance$ 
```

Figure 4.15: Smart Contract deployment and contract details.

4.3 Function Description of Smart Contract (Insurance)

1. purchasePolicy() : This function allows the user to purchase a policy. At first the user requests for insurance by passing the credentials to the authority. Later the authority verifies the credentials by passing the credentials to the offchain database.

2. fileClaim() : This function will allow the user to claim a file. A user must have a valid policy to claim a file. Therefore, a user can claim a file by passing the amount which must be in a fixed limit. The upper limit cannot exceed the policy value and the lower limit cannot be null.

3. approveClaim() : This function will allow the owner to approve the claim. If a user claims any amount using the address of the policyholder it will verify first if It's the owner. Also the claim cannot be zero. After verifying It'll approve the claim.

4. getPolicy() : This function requires the address of the policyholder. After passing the address It'll get the policies in return.

5. getClaim() : This function also requires the address of the policyholder. After passing the address, it'll get the claims in return.

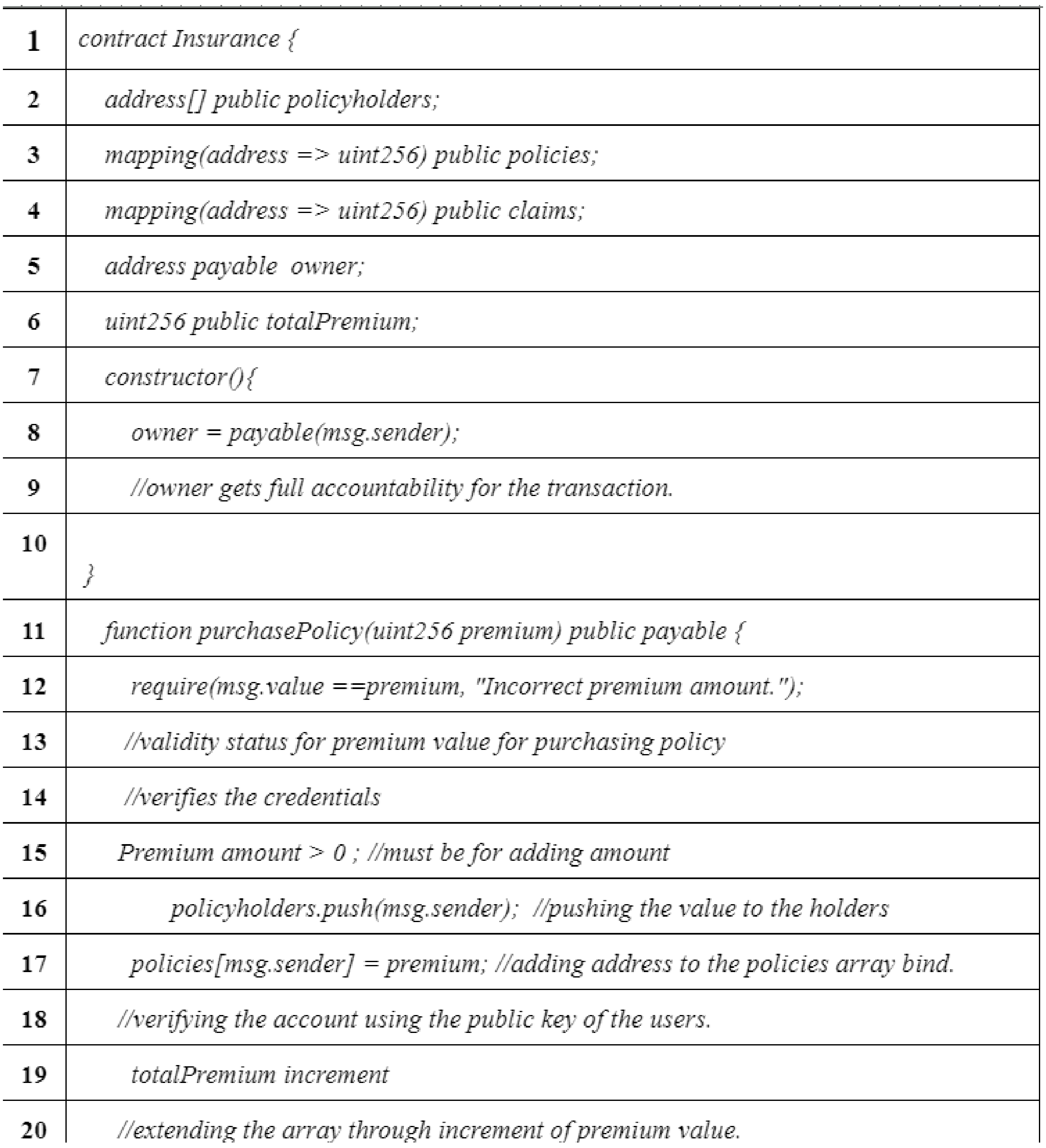
6. getTotalPremium() : This function allows the user to check the total premium.

7. grantAccess() : This function requires the address of the payable users. This function will only work if the proper owner uses it. Others cannot give access.

8. revokeAccess() : This function also requires the address of the payable users. This function will also work if the proper owner uses it. A proper owner can revoke the access by using this function. Also the user cannot be a current owner else it cannot revoke access.

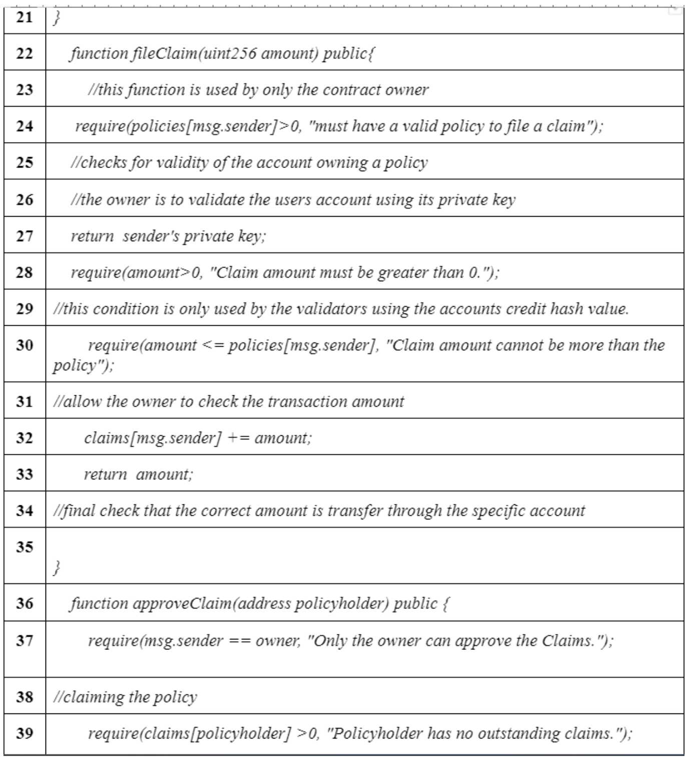
9. destroy() : This function can be used by the owners. An owner may destroy the contract anytime.

4.4 Pseudocode of Smart Contract no-1 (Insurance):



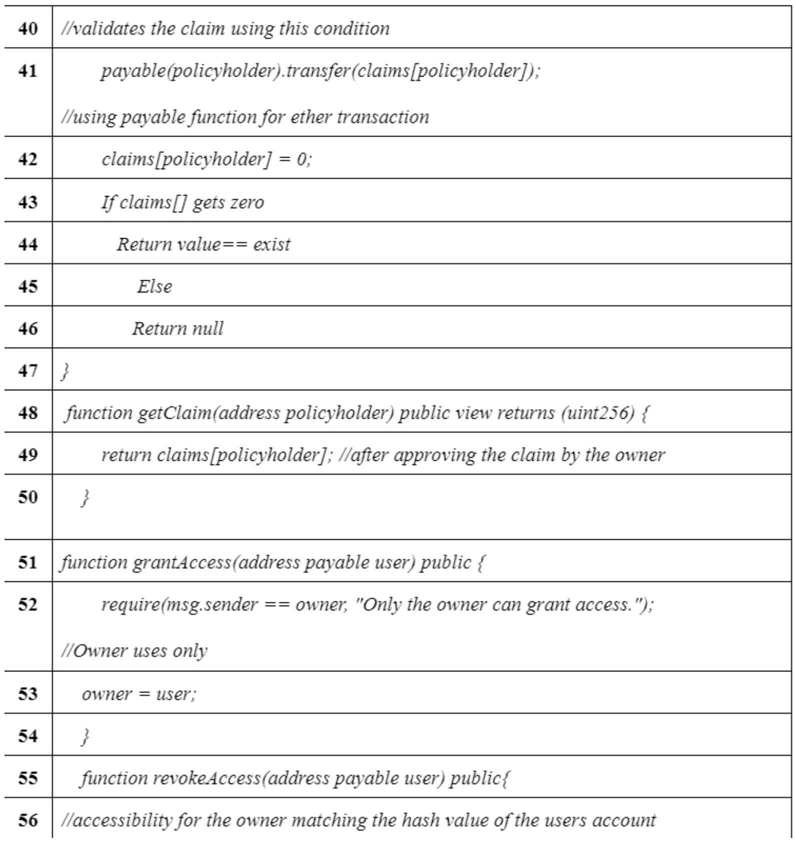
```
contract Insurance {
   address[] public policyholders;
   mapping(address => uint256) public policies;
   mapping(address => uint256) public claims;
   address payable  owner;
   uint256 public totalPremium;
   constructor(){
      owner = payable(msg.sender);
      //owner gets full accountability for the transaction.

}
   function purchasePolicy(uint256 premium) public payable {
      require(msg.value ==premium, "Incorrect premium amount.");
     //validity status for premium value for purchasing policy
     //verifies the credentials
     Premium amount > 0 ; //must be for adding amount
          policyholders.push(msg.sender);  //pushing the value to the holders
      policies[msg.sender] = premium; //adding address to the policies array bind.
     //verifying the account using the public key of the users.
      totalPremium increment
     //extending the array through increment of premium value.
```

Figure 4.16: Smart Contract Insurance part-1



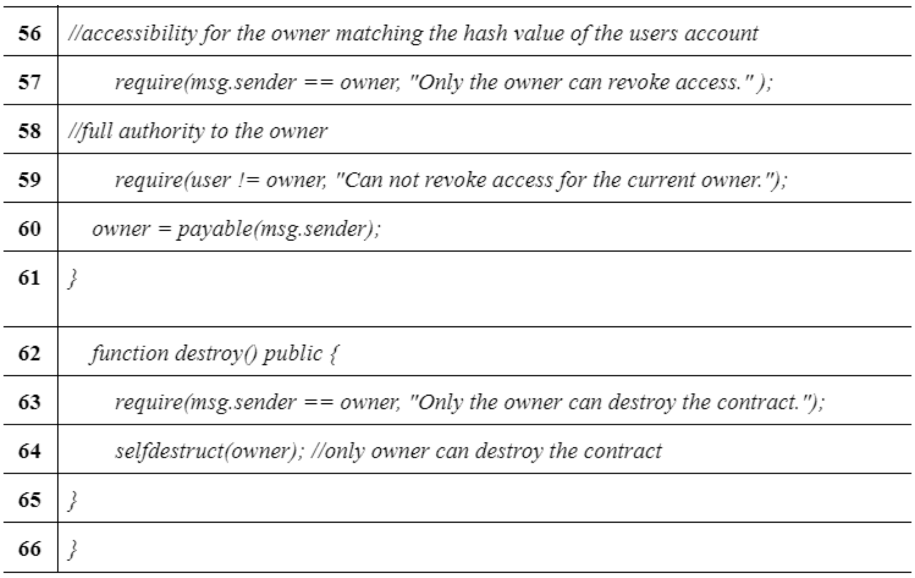
```
}
   function fileClaim(uint256 amount) public{
      //this function is used by only the contract owner
    require(policies[msg.sender]>0, "must have a valid policy to file a claim");
   //checks for validity of the account owning a policy
   //the owner is to validate the users account using its private key
   return  sender's private key;
   require(amount>0, "Claim amount must be greater than 0.");
//this condition is only used by the validators using the accounts credit hash value.
      require(amount <= policies[msg.sender], "Claim amount cannot be more than the
policy");
//allow the owner to check the transaction amount
      claims[msg.sender] += amount;
      return  amount;
//final check that the correct amount is transfer through the specific account

}
   function approveClaim(address policyholder) public {
      require(msg.sender == owner, "Only the owner can approve the Claims.");

//claiming the policy
      require(claims[policyholder] >0, "Policyholder has no outstanding claims.");
```

Figure 4.17: Smart Contract Insurance part-2



```
//validates the claim using this condition
      payable(policyholder).transfer(claims[policyholder]);
//using payable function for ether transaction
      claims[policyholder] = 0;
      If claims[] gets zero
        Return value== exist
           Else
          Return null
}
 function getClaim(address policyholder) public view returns (uint256) {
      return claims[policyholder]; //after approving the claim by the owner
   }

function grantAccess(address payable user) public {
      require(msg.sender == owner, "Only the owner can grant access.");
//Owner uses only
   owner = user;
   }
   function revokeAccess(address payable user) public{
//accessibility for the owner matching the hash value of the users account
```

Figure 4.18: Smart Contract Insurance part-3



```
//accessibility for the owner matching the hash value of the users account
        require(msg.sender == owner, "Only the owner can revoke access." );
//full authority to the owner
        require(user != owner, "Can not revoke access for the current owner.");
    owner = payable(msg.sender);
}

    function destroy() public {
        require(msg.sender == owner, "Only the owner can destroy the contract.");
        selfdestruct(owner); //only owner can destroy the contract
}
}
```

Figure 4.19: Smart Contract Insurance part-4

4.5 Function Description of Smart Contract (Health)

1. reg() : This function is used by the authorized validators after collecting information from the users. Our user's will be the insurance companies and our admin will be the validators. This function collects information like the name of the company, payment status, amount, company's license date and also the license expiry date.These informations are verified by the admin. After verifying they generate key pairs for the company. One will be the public key of the company and another will be the private one. After that our admin will generate accounts and return it to the companies.

2. allowHolder() : This function will pass a condition. If the amount is within the policy then the owner can use this to approve the amount. It'll directly work with the boolean condition.

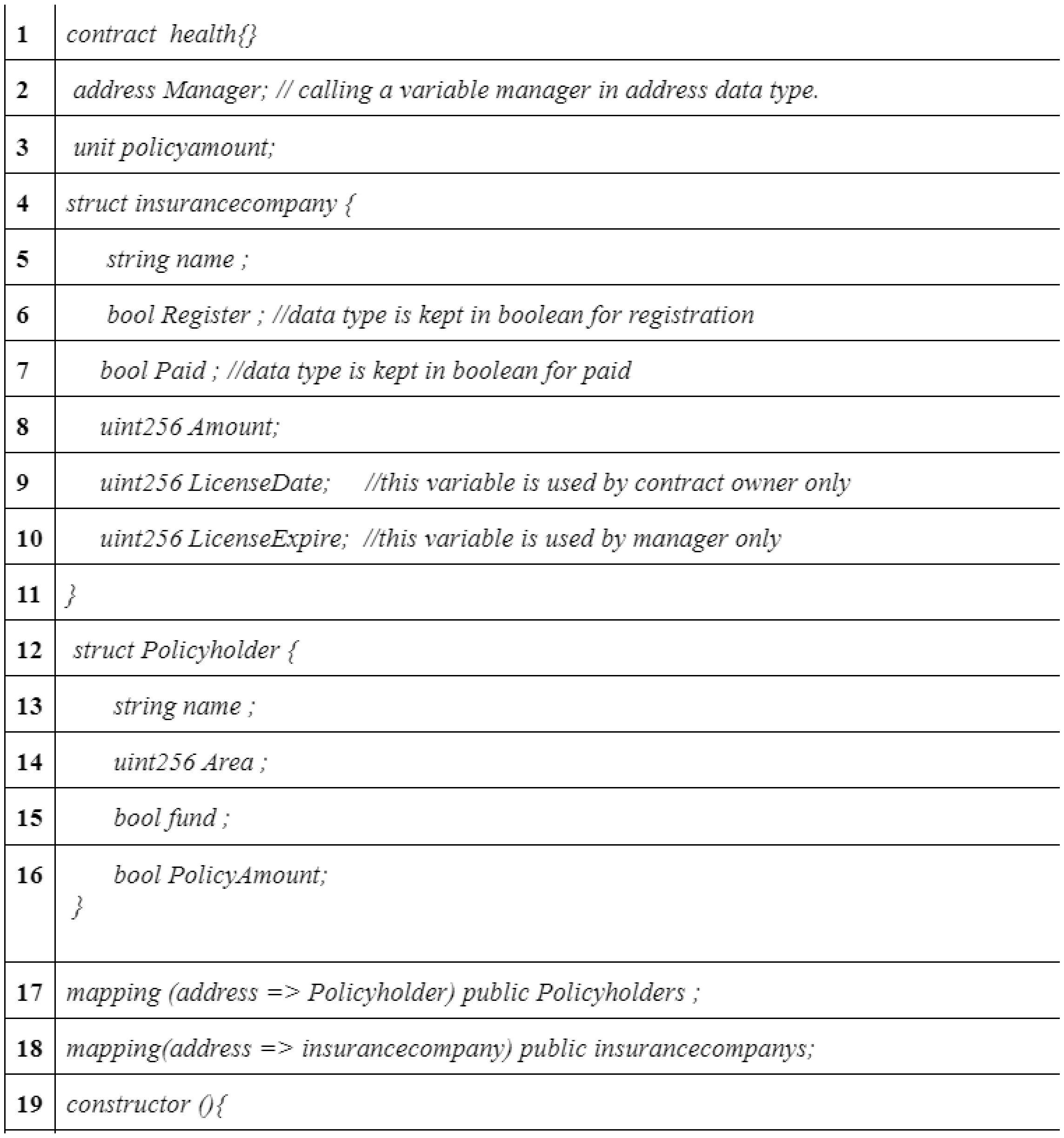
3. policyAmounts() : This function will set the policy amount as the amount. Only the owner can use it and view it.

4. policyHolder() : This function will create the policy holders. First, It'll check the policy amount with the values. If it matches then It'll set the policy holder's name and the address.

5. withdrawforHolder() : This function requires the policyholder's policy amount to be positive. Also if the fund is false then it means It's already funded. If the area is the given area then there will be a transfer and the fund will become true now.

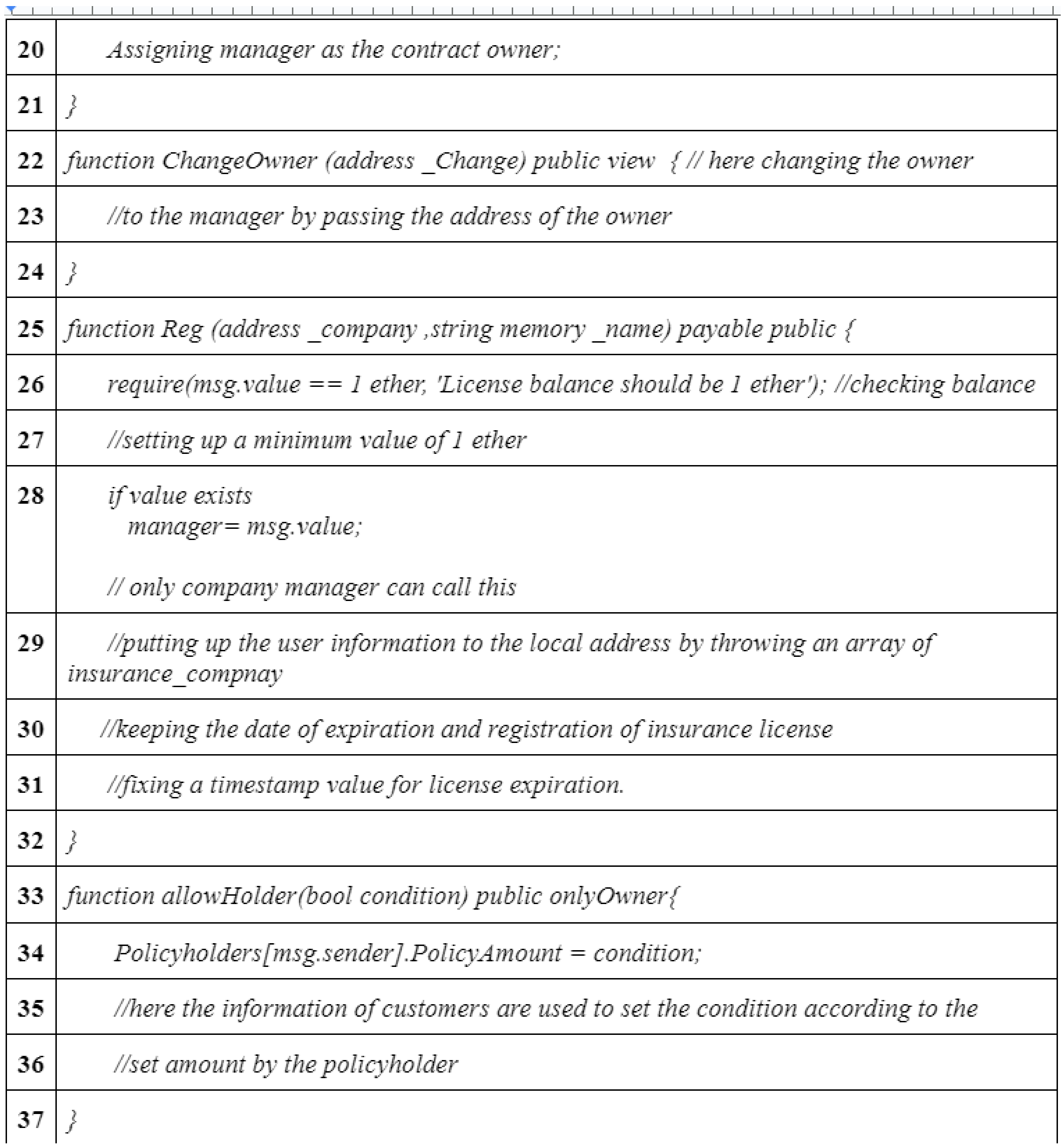
6. withdrawForCompany() : This function will check payment status for the license of the insurance company. Also, it'll check the expiry date of the license. Then It'll transfer the balance. Only the manager or the owner can modify it.

4.6 Pseudocode of Smart Contract no-2 (Health)



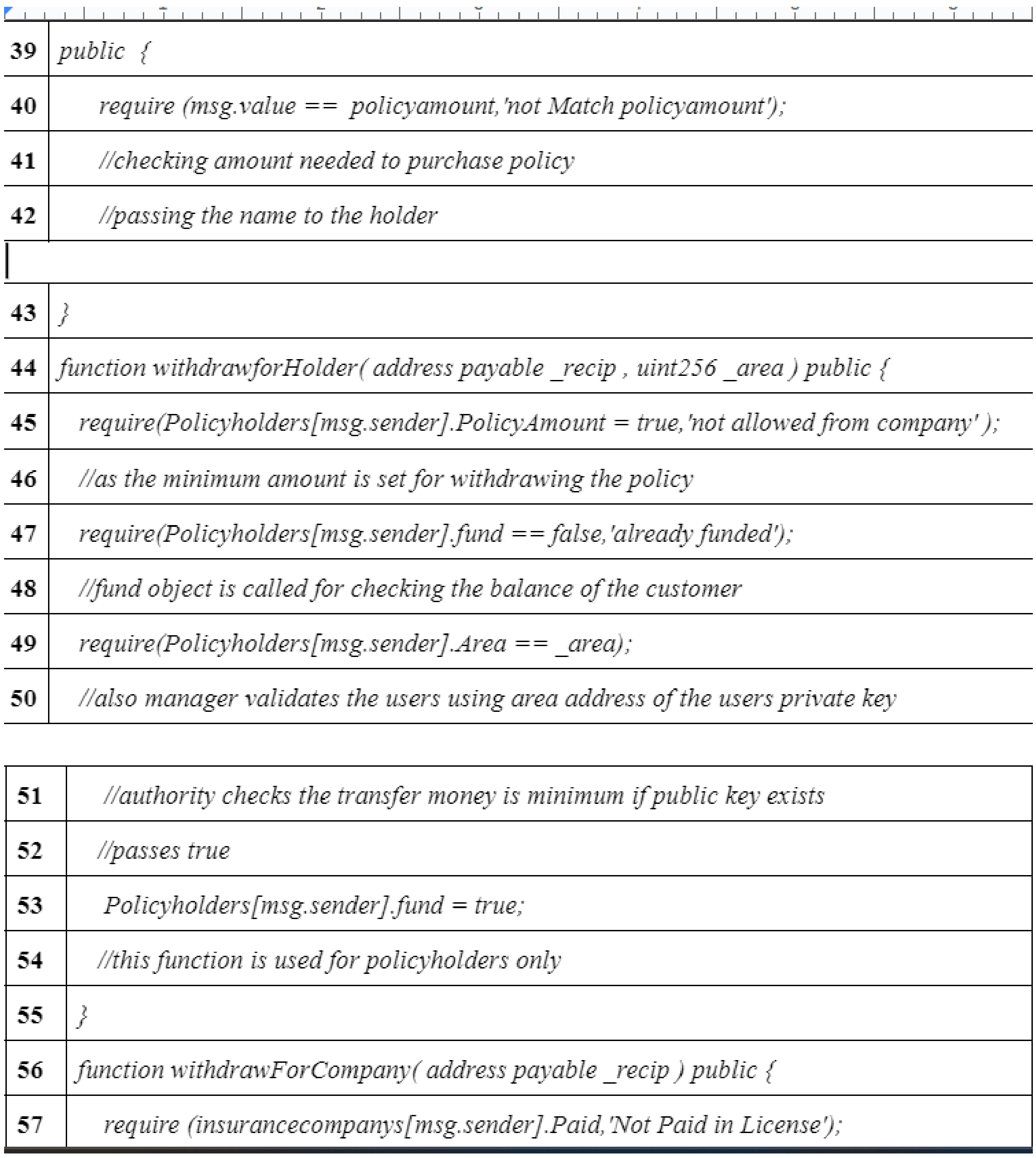
```
contract  health{}
 address Manager; // calling a variable manager in address data type.
 unit policyamount;
struct insurancecompany {
      string name ;
      bool Register ; //data type is kept in boolean for registration
     bool Paid ; //data type is kept in boolean for paid
     uint256 Amount;
     uint256 LicenseDate;     //this variable is used by contract owner only
     uint256 LicenseExpire;  //this variable is used by manager only
}
 struct Policyholder {
       string name ;
       uint256 Area ;
       bool fund ;
       bool PolicyAmount;
 }
mapping (address => Policyholder) public Policyholders ;
mapping(address => insurancecompany) public insurancecompanys;
constructor (){
```

Figure 4.20: Smart Contract Health part-1



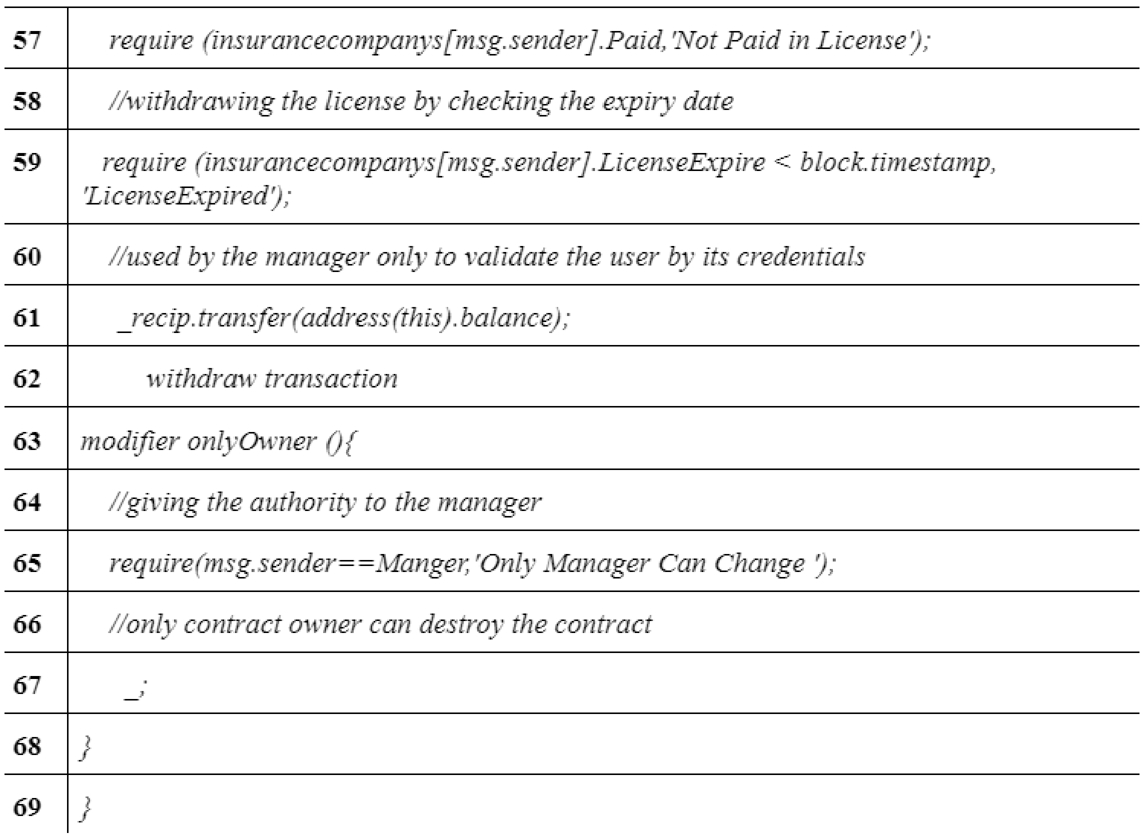
```
    Assigning manager as the contract owner;
}
function ChangeOwner (address _Change) public view  { // here changing the owner
    //to the manager by passing the address of the owner
}
function Reg (address _company ,string memory _name) payable public {
    require(msg.value == 1 ether, 'License balance should be 1 ether'); //checking balance
    //setting up a minimum value of 1 ether
    if value exists
      manager= msg.value;

    // only company manager can call this
    //putting up the user information to the local address by throwing an array of
insurance_compnay
   //keeping the date of expiration and registration of insurance license
    //fixing a timestamp value for license expiration.
}
function allowHolder(bool condition) public onlyOwner{
     Policyholders[msg.sender].PolicyAmount = condition;
     //here the information of customers are used to set the condition according to the
     //set amount by the policyholder
}
```

Figure 4.21: Smart Contract Health part-2



```
public  {
    require (msg.value ==  policyamount,'not Match policyamount');
    //checking amount needed to purchase policy
    //passing the name to the holder

}
function withdrawforHolder( address payable _recip , uint256 _area ) public {
  require(Policyholders[msg.sender].PolicyAmount = true,'not allowed from company' );
  //as the minimum amount is set for withdrawing the policy
  require(Policyholders[msg.sender].fund == false,'already funded');
  //fund object is called for checking the balance of the customer
  require(Policyholders[msg.sender].Area == _area);
  //also manager validates the users using area address of the users private key
    //authority checks the transfer money is minimum if public key exists
   //passes true
    Policyholders[msg.sender].fund = true;
   //this function is used for policyholders only
}
function withdrawForCompany( address payable _recip ) public {
    require (insurancecompanys[msg.sender].Paid,'Not Paid in License');
```

Figure 4.22: Smart Contract Health part-3



```
    require (insurancecompanys[msg.sender].Paid,'Not Paid in License');
    //withdrawing the license by checking the expiry date
   require (insurancecompanys[msg.sender].LicenseExpire < block.timestamp,
'LicenseExpired');
    //used by the manager only to validate the user by its credentials
     _recip.transfer(address(this).balance);
        withdraw transaction
modifier onlyOwner (){
    //giving the authority to the manager
    require(msg.sender==Manger,'Only Manager Can Change ');
    //only contract owner can destroy the contract
      _;
}
}
```

Figure 4.23: Smart Contract Health part-4

Chapter 5

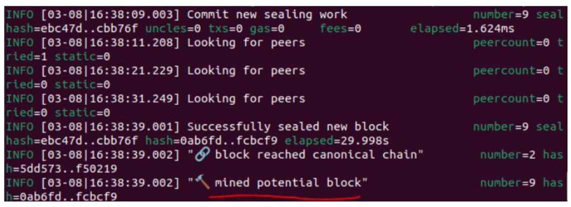
Result analysis

5.1 Transaction Speed using Geth-Client

A blockchain network's transaction rates per second rely on elements including block time, gas limit, transaction gas, and network traffic. The formula for calculating transaction speed (measured in transactions per second) is: transaction speed = Block gas limit (Transaction gas Block time). Block time is ambiguous in the geth implementation of Ethereum networks during the construction of the genesis block. It's the amount of time that passes before a new block is introduced to the network. The block time in the suggested framework is fixed as 15 s. Not many blocks are lost from the network because the block time is not extremely low. Another factor that affects an Ethereum network's throughput (transaction speed) is the gas limit.

5.2 Transaction Speed using Ganache

An open-source blockchain called Ethereum makes it possible to use smart contract capabilities. A peer-to-peer network that runs client software to store, validate, and produce blocks is formed when hundreds of machines, known as "Ethernet nodes, unite to form the Ethereum network. One of those clients is Ganache. Which is an Ethereum client, enabling you to manage your own personal blockchain. By specifying, for instance, the number of threads you provide for mining, you can change your needs accordingly. Ganache is a command-line tool that can operate a complete blockchain network that has already been set up locally. You can run your own Javascript scripts using the command lines, a Json-rpc server, and an interactive console that is provided. Here we can see the results of blocks mined successfully.



```
INFO [03-08|16:38:09.003] Commit new sealing work                  number=9 seal
hash=ebc47d..cbb76f uncles=0 txs=0 gas=0     fees=0       elapsed=1.624ms
INFO [03-08|16:38:11.208] Looking for peers                        peercount=0 t
ried=1 static=0
INFO [03-08|16:38:21.229] Looking for peers                        peercount=0 t
ried=0 static=0
INFO [03-08|16:38:31.249] Looking for peers                        peercount=0 t
ried=0 static=0
INFO [03-08|16:38:39.001] Successfully sealed new block            number=9 seal
hash=ebc47d..cbb76f hash=0ab6fd..fcbcf9 elapsed=29.998s
INFO [03-08|16:38:39.002] "🔗 block reached canonical chain"        number=2 has
h=5dd573..f50219
INFO [03-08|16:38:39.002] "🔨 mined potential block"                number=9 has
h=0ab6fd..fcbcf9
```

Figure 5.1: Minned Block

5.3 What we will achieve in contrast to what we have

This piece highlights the things that we may achieve that are absent in the current system. The security of traditional health insurance is compromised by data manipulation, while advanced health insurance is fortified by blockchain technology. The process of verifying an account involves the utilization of both public and private keys along with the hash value. However, it is important to note that this process is susceptible to compromise. The act of adding information is a process that is comparatively simpler and more expedient. The process of establishing a connection with newly introduced medical services has been expedited. The process of integrating additional payment gateways has been streamlined for improved efficiency and simplicity. Customer service can be obtained instantly. Modern health insurance employs a stronger security mechanism through the utilization of public and private keys, as well as the hash value. The implementation of the system will result in gaining the trust of customers. The system will provide transaction proof, require identity verification, and ensure policy immutability with full transparency. In the event of a false claim or the creation of a false policy, an alert will be generated and directed against the responsible policymaker or individual. The implementation of measures to prevent fraud will lead to a reduction in the number of fraudulent claims and decrease the likelihood of fake policy creation. This will result in a more trustworthy insurance sector for clients. Apart from these, an added advantage is the scalability of the system. The system's efficiency can be achieved through the proposed model, and unlike other blockchain-based techniques, it will not only verify the policy stake but also keep the transaction records and authenticity of the client.

5.4 Comparison between Regular Health Insurance and Advanced Health Insurance

Comparison between Regular Health Insurance and Advanced Health Insurance		
Comparison Action	Regular Health Insurance	Advanced Health Insurance
Data manipulation	Data can be manipulated.	Data cannot be manipulated because of strong security system of blockchain.
Changing information	Easier to change any information which makes the information vulnerable	It takes verification steps to change any information.
Adding information	Any additional information can be added easily.	Any additional information need the proper verification to be added.
Binding with new medical services	The paperwork or the traditional system will take some time to bind with any new services.	It will be faster to bind with new medical services.
Adding new payment gateway	Adding a payment gateway will require lots of paperwork	It's easier and faster in our advanced system to add any new payment gateway
Taking customer service	It requires time to get customer service	They get instant customer services.
Account verification	Account verification is done with the basic information which makes the security weak.	Account verification system is done using public and private keys and the hash value which make the security really strong.
Can be damaged	The whole system might be damaged.	Our system security and backup makes it reliable

5.5 Cost Analysis

We have used blockchain technology in our insurance system. It is beneficial economically mostly because of the technical benefits. As this system ensures the elimination of the intermediaries which is of no use in blockchain protocol. That's why initially the expenses are lessened. It is estimated that a figure between USD 15 billion to USD 20 billion financial institution infrastructure charges can be reduced in a calendar year. Moreover, there is no need for complex audits as blockchain never stops its personal audits and publishes them without any kind of latency. This particularly reduces the charges due to financial institutions outside audits significantly which is much more extraordinarily high priced. As a result, the reduction of charges turns out quite beneficial for the banks. Since the blockchain system has the self-maintenance mechanism, it actually cuts back the maintenance fee also. It is a blessing technically and economically for the banks and can plunge the regular expenses quite substantially.

Chapter 6

Conclusion

The purpose of this research is to introduce a smart contract insurance architecture that operates on the blockchain. Health insurance transactions are processed using a private, distributed system based on Ethereum, which is extremely safe. Standard insurance contracts are created in this architecture using smart contracts. The decentralized Solidity smart contracts employed in this system are immutable, which streamlines the insurance and claim settlement procedures.Within this specific architecture, the Proof of Authority (PoA) method is used, which enhances cost-effectiveness and storage efficiency. The framework provides a safe and efficient means of conducting business in the insurance industry. Popular because it makes it easy to design unique smart contracts, the framework has attracted a lot of developers. This framework provides a secure and transparent way to carry out the whole insur- ance process, from application to refund. The scalability of this design is assessed by expanding the number of peers for a given block size of 20. It has been shown that the confirmation time increases with the number of peers. The confirmation procedure takes longer with more peers, but security is noticeably increased with more validators. The recommended system framework still has plenty of room for improvement. In a nutshell, it is the central element of all insurance-related activity. if it is thought that a special circumstance like the IoT-based car insurance process exists. Based on the suggested framework, insurance policies or disciplines similar to supply chain, etc., can be easily implemented based on the given framework. Thus, it has numerous opportunities to develop.
This study presents a framework that is not specific to any one domain. It focuses on a standard approach for standard insurance programs. The above algorithm is more suitable for enhancing the PoA, PoW, and PoS protocols in the necessary models for any particular type of insurance. In general, these technologies have the potential to provide financial security and improved, more personalized healthcare insurance to a greater number of individuals, especially those with lower incomes. The models of distribution channels can also simplify matters. Introducing insurance to less developed nations and streamlining the health insurance process.

Bibliography

References

1. W. Liu, Q. Yu, Z. Li, Z. Li, Y. Su and J. Zhou, ”A Blockchain-Based System for Anti-Fraud of Healthcare Insurance,” 2019 IEEE 5th International Conference on Computer and Communications (ICCC), 2019, pp. 1264-1268, doi: 10.1109/ICCC47050.2019.9064274

2. Y. Hu et al., ”A Delay-Tolerant Payment Scheme Based on the Ethereum Blockchain,” in IEEE Access, vol. 7, pp. 33159-33172, 2019, doi: 10.1109/ACCESS.2019.2903271.

3. V. Aleksieva, H. Valchanov and A. Huliyan, ”Application of Smart Contracts based on Ethereum Blockchain for the Purpose of Insurance Services,” 2019 International Conference on Biomedical Innovations and Applications (BIA), 2019, pp. 1-4, doi: 10.1109/BIA48344.2019.8967468.

4. G. Saldamli, V. Reddy, K. S. Bojja, M. K. Gururaja, Y. Doddaveerappa and L. Tawalbeh, ”Health Care Insurance Fraud Detection Using Blockchain,” 2020 Seventh International Conference on Software Defined Systems (SDS), 2020, pp. 145-152, doi: 10.1109/SDS49854.2020.9143900.

5. G. Saldamli, V. Reddy, K. S. Bojja, M. K. Gururaja, Y. Doddaveerappa and L. Tawalbeh, ”Health Care Insurance Fraud Detection Using Blockchain,” 2020 Seventh International Conference on Software Defined Systems (SDS), 2020, pp. 145-152, doi: 10.1109/SDS49854.2020.9143900.

6. Viaene, S., Dedene, G. (2004). Insurance Fraud: Issues and Challenges. The Geneva Papers on Risk and Insurance. Issues and Practice, 29(2), 313–333. http://www.jstor.org/stable/41953118

7. K. Kapadiya et al., ”Blockchain and AI-Empowered Healthcare Insurance Fraud Detection: an Analysis, Architecture, and Future Prospects,” in IEEE Access, vol. 10, pp. 79606-79627, 2022, doi: 10.1109/ACCESS.2022.3194569.

8. A. Verma, A. Taneja and A. Arora, ”Fraud detection and frequent pattern matching in insurance claims using data mining techniques,” 2017 Tenth International Conference on Contemporary Computing (IC3), 2017, pp. 1-7, doi: 10.1109/IC3.2017.8284299.

9. E. A. Duman and Ş. Sağıroğlu, ”Health care fraud detection methods and new approaches,” 2017 International Conference on Computer Science and Engineering (UBMK), 2017, pp. 839-844, doi: 10.1109/UBMK.2017.8093544.

10. A. Pinna, S. Ibba, G. Baralla, R. Tonelli and M. Marchesi, ”A Massive Analysis of Ethereum Smart Contracts Empirical Study and Code Metrics,” in IEEE Access, vol. 7, pp. 78194-78213, 2019, doi: 10.1109/ACCESS.2019.2921936.

11. M. Zghaibeh, U. Farooq, N. U. Hasan and I. Baig, ”SHealth: A Blockchain-Based Health System With Smart Contracts Capabilities,” in IEEE Access, vol. 8, pp. 70030-70043, 2020, doi: 10.1109/ACCESS.2020.2986789.

12. F. Tang, S. Ma, Y. Xiang and C. Lin, "An Efficient Authentication Scheme for Blockchain-Based Electronic Health Records," in IEEE Access, vol. 7, pp. 41678-41689, 2019, doi: 10.1109/ACCESS.2019.2904300.

13. Hayes, Adam. "Blockchain Facts: What Is It, How It Works, and How It Can Be Used." Investopedia, 24 June 2022, www.investopedia.com/terms/b/blockchain.asp.

14. Proof-of-Stake (PoS). (2022, June 10). Investopedia. Retrieved September 16,2022, from https://www.investopedia.com/terms/p/proof-stake-pos.asp

15. Cointelegraph. (2022, July 27). Proof-of-authority vs. proof-of-stake: Key differences explained. Retrieved September 16, 2022, from https://cointelegraph.com/blockchain-for-beginners/proof-of-authority-vs-proof-of-stake-key-differences-explained

16. Nakamoto, S. (n.d.). Bitcoin: A Peer-to-Peer Electronic Cash System [Paper].

17. D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei and C. Qijun, "A review on consensus algorithm of blockchain," 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 2017, pp. 2567-2572, doi: 10.1109/SMC.2017.8123011

18. Yusoff, J., Mohamad, Z. and Anuar, M. (2022) "A review: Consensus algorithms on Blockchain," Journal of Computer and Communications, 10(09), pp. 37–50. Available at: https://doi.org/10.4236/jcc.2022.109003.

19. Chawla, V. (2020) What Are The Top Blockchain Consensus Algorithms? https://analyticsindiamag.com/blockchain-consensus-algorithms/

20. D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei and C. Qijun, "A review on consensus algorithm of blockchain," 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 2017, pp. 2567-2572, doi: 10.1109/SMC.2017.8123011.

21. T. Hepp, M. Sharinghousen, P. Ehret, A. Schoenhals, and B. Gipp, "On-chain vs. off-chain storage for supply-and blockchain integration," it-Information Technology, vol. 60, no. 5-6, pp. 283–291, 2018.

22. Smart Contracts in Insurance: Revolutionizing Claims Processing and More. ScienceSoft. Retrieved from https://www.scnsoft.com/insurance/smart-contracts.

23. https://www.oreilly.com/library/view/blockchain-quick-start/9781789807974/9134ee72-d1a3-42ad-82c3-38d3f2fb8aa8.xhtml