Blockchain-based Edge computing for Medical Data Storage & Processing using Federated Learning

by

Fazle Rabbi Faiyaz
17101369
Afrin Sultana Lisa
16201055
Laisa Rahat
17201036
Nafisa Tabassum
17141023
Walid Bin Istiaq
17101392

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 and Engineering

Department of Computer Science and Engineering
Brac University
June 2021

Declaration

It is hereby declared that

1. The thesis submitted is 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.

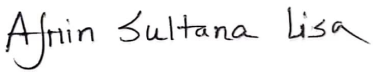
Student's Full Name & Signature:

Fazle Rabbi Faiyaz
17101369

Nafisa Tabassum
17141023

Laisa Rahat
17201036

Walid Bin Istiaq
17101392



Afrin Sultana Lisa
16201055

Approval

The thesis titled "Blockchain-based Edge computing for Medical Data Storage & Processing using Federated Learning" submitted by

1. Fazle Rabbi Faiyaz (17101369)
2. Nafisa Tabassum (17141023)
3. Laisa Rahat (17201036)
4. Walid Bin Istiaq (17101392)
5. Afrin Sultana Lisa (16201055)

of Spring, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on June 06, 2021.

Examining Committee:

Supervisor:
(Member)

Moin Mostakim
Lecturer
Department of Computer Science and Engineering
Brac University

Co-supervisor:
(Member)

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

Program Coordinator:
(Member)

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

Head of Department:
(Chairperson)

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

Abstract

With a great number of IoT devices being used in healthcare and a massive rise in medical data produced by these devices, data storage and processing systems using the traditional cloud computing framework are not enough to meet real-time data re- requirements in Internet-based services as data is transferred to faraway cloud servers for processing, resulting in high latency and costs. Edge computing can provide a solution to this problem by effectively offloading a portion of the workload from the cloud to nearby edge servers to perform data processing tasks close to the end-users, thus reducing latency and cost as well as improving the quality of service. However, edge computing faces threats regarding data privacy and security due to edge nodes being more vulnerable to cyber-attacks. To address this problem, blockchain can be integrated to protect data from tampering, maintain data integrity, and allow reliable access, distributed computation, and decentralized data storage. Thus, in this research, we present a secure medical data storage and processing system using blockchain and edge computing to preserve our clients' data privacy. To tackle privacy and security concerns, federated learning using a neural network has been used to train models locally using the data on the edge nodes rather than sending relevant private information to a centralized server for training, and model parameters, as well as IPFS file hashes and other private information, are securely stored on the blockchain by incorporating cryptographic techniques.

Keywords: IoT; Edge Computing; Blockchain; Federated Learning; IPFS

Acknowledgement

Firstly, we are grateful to the Almighty Allah for whom we have been able to finish our thesis without any major interruptions. Secondly, we wish to express our sincere thanks to our supervisor Mr. Moin Mostakim and co-supervisor Dr. Md. Golam Rabiul Alam of the Department of Computer Science and Engineering at BRAC University for their guidance and support throughout our entire thesis. They helped us whenever we needed it, motivating and enabling us to bring forth new ideas and make correct implementation decisions. Thirdly, we would like to take this chance to thank all of the faculty members for the help and support they have provided in our time in Brac University. Lastly, we would like to express our gratitude towards our parents for their continued prayers, encouragement, and support without which it would not have been possible for us to reach this point.

Table of Contents

Declaration i

Approval ii

Abstract iv

Acknowledgment v

Table of Contents vi

List of Figures viii

List of Tables ix

List of Abbreviations x

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Motivation . . . 1
- 1.3 Objectives . . . 2
- 1.4 Thesis Organization . . . 2

2 Literature Review 4

3 Proposed blockchain-based edge computing framework 9
- 3.1 Cloud layer . . . 9
- 3.2 Global blockchain network . . . 10
- 3.3 Edge server layer . . . 10
- 3.4 Local blockchain network . . . 10
- 3.5 IoT device layer . . . 11

4 Using Blockchain for Security & Data Preservation 12
- 4.1 Within the Ethereum smart contract . . . 12
- 4.2 Account access . . . 13
- 4.3 Storing and retrieving files . . . 14
- 4.4 File sharing . . . 15
- 4.5 Storing local model weights for FL . . . 17

5 Analysis & Results **18**
5.1 Evaluating the application of blockchain in the proposed framework . 18
5.2 Federated learning with blockchain 20
5.2.1 Data pre-processing and train-test split 20
5.2.2 Creating clients 21
5.2.3 Processing and batching clients' and test data 22
5.2.4 Creating the Multi-Layer Perceptron 23
5.2.5 Federated model training with blockchain 25
5.2.6 Evaluating performance 28

6 Conclusion **32**

Bibliography **33**

List of Figures

3.1 Blockchain-based edge computing framework 9

4.1 Inside the Storage smart contract 12
4.2 Account access 13
4.3 Storing files 14
4.4 File retrieval 15
4.5 File sharing 16
4.6 Storing weights 17

5.1 Gas consumption of smart contract processes on transactions 18
5.2 Performance of smart contract processes 19
5.3 Label counts of clients created using training data 22
5.4 Model summary of MLP for binary classification 23
5.5 FL architecture with MLP neural network 24
5.6 Loss of each client's local model using training data 26
5.7 Accuracy of each client's local model using training data 27
5.8 Loss of Global Model over 100 rounds 28
5.9 Accuracy of Global Model over 100 rounds 28
5.10 Confusion matrix of Global Model 29
5.11 Confusion matrices of each client's local model 31

List of Tables

5.1 Comparison between models 28

List of Abbreviations

The next list describes several abbreviations that will be later used within the body of the document

AES Advanced Encryption Standard

ANN Artificial Neural Network

CDN Content Delivery Network

Dapp Decentralized Application

DoS Denial-of-Service

FL Federated Learning

IoT Internet of Things

IPFS InterPlanetary File System

MEC Mobile Edge Computing

ML Machine Learning

MLP Multilayer Perceptron

P2P Peer-to-Peer

PoW Proof of Work

ReLU Rectified Linear Unit

RSA Rivest–Shamir–Adleman

SGD Stochastic Gradient Descent

Chapter 1

Introduction

1.1 Background

The current world revolves around data and the amount of data being generated is increasing at a rapid pace as time goes on. This is why so many cloud storage systems such as Google Drive, Dropbox, Microsoft OneDrive have been created where people store their important files/data. Medical data is very valuable because medical records often contain the private information of individuals such as social security numbers, their home address, the medical treatments they have undergone, information regarding family members and employment. According to reports, this type of information is worth more than a credit card on the black market [29]. It may seem like a good idea to store your medical data in a cloud storage system at first, however, transmitting data back and forth between distant cloud servers has resulted in high latency, bandwidth consumption, and energy consumption as well as there may be data loss due to a single point of failure and thus gradually cloud computing has become unable to meet the demands of IoT applications [13]. Moreover, current security systems are inadequate as medical data is still being stolen from healthcare organizations and people are having their private information leaked, removed, or altered even from cloud storage systems due to data breaches. Hence, securing and preserving private medical data has become a huge concern around the world nowadays.

1.2 Motivation

At present, maintaining the privacy and security of stored medical data is one of the greatest problems we are facing with the situation getting worse over the years. Due to the issues presented by the traditional cloud computing paradigm, we switch to edge computing. Edge computing allows us to move processing and storage tasks close to the network's edge and thus reduces latency, costs, and energy consumption. It also allows periodical data uploads to cloud servers for further processing and storage if needed. However, edge computing alone is not enough to keep data safe as it presents vulnerabilities due to multiple attack surfaces, the distributed nature of the edge nodes across the local network and cloud, closeness to sensitive data generators, collection of heterogeneous data in the devices, and the scale of the network [23]. If hackers can successfully take advantage of these vulnerabilities, there will be severe repercussions due to large amounts of data being leaked or

tampered with and it will be hard difficult to track them down as well. Although there are some countermeasures to this security risk such as CAPTCHAs, it still has limitations in protecting data [24]. Nowadays, blockchain is strongly recommended for privacy protection purposes as it creates a distributed digital ledger that keeps decentralized, tamper-resistant records of data. However, it is a challenging task to find a proper way to integrate these two architectures. Therefore, we have based our work on incorporating both edge computing and blockchain together in an effective manner in addition to applying federated learning and cryptography to provide a secure, user-friendly, decentralized storage and processing system that shifts data processing and storage near the network's edge, allowing data transmission at low latency and minimal cost as well as preserving the privacy and integrity of medical data.

1.3 Objectives

In our research, we aim to find a way to create a system by integrating blockchain with edge computing which will provide users with a fast, secure, and user-friendly way to store and process private medical data at minimal cost to be used in the healthcare industry. To accomplish our main goal, we will need to achieve the following objectives:

- Allow users to access their account in the site/application using their email address or phone number, password, and a user-set question (only in the case of the account being accessed from an unregistered device of the user or on password reset).
- Minimize latency, energy consumption, and costs of data transmission.
- Allowing secure storage of medical data provided by the user by encrypting it before storing it in IPFS and decrypting data after it is retrieved for use.
- Find a suitable way to incorporate edge computing and blockchain to provide fast access and data preservation.
- Allow users to share particular files or folders with other registered users securely when necessary.
- Use federated learning to address security and privacy concerns during data processing and analysis.

1.4 Thesis Organization

This section provides an overview of what has been discussed in each chapter of our thesis paper. After discussing what we plan on doing, the reason behind it, and what we hope to accomplish in this chapter, the next chapter is the literature review where we summarize and discuss the information collected from various scholarly articles, books, past research papers, and other sources of information which are relevant to our area of research. In chapter-3, we present our proposed blockchain-based edge computing framework and discuss it in detail, and in chapter-4, we describe

the application of blockchain in securing and preserving data during storage and processing. We analyze and evaluate the results of both the use of blockchain in our proposed framework and the performance of federated learning with blockchain in chapter-5 and finally, we conclude our work in chapter-6.

Chapter 2

Literature Review

Huge amounts of data are being produced every day through different IoT devices such as phones, tablets, PCs, sensors, etc. However, this has raised an important concern regarding a particular matter and that is the privacy of data. Data privacy refers to how data should be handled based on its relative performance [1]. According to Geambasu et al.'s paper [2], due to our rising dependence on Web services, our data is being cached, copied, and archived without us knowing about it or being able to control it. Data privacy is a very important concern because if our data is not properly handled, our data may get leaked due to a breach, leading to identity theft, unwanted disclosure of important information, and legal implications. The threat to data privacy is even more of an issue in the healthcare industry with patient data breaches on the rise, leading to millions of individual records being exposed. Therefore, our research aims to protect medical data using blockchain, edge computing, and federated learning.

So far, cloud computing has provided us reliable and convenient Internet-based services by processing and storing data in remote servers. However, according to Irei [3], due to the constant growth of connected devices and their continuous use of Internet-based services, cloud computing may soon prove to be insufficient to handle this situation. This is because, with the constant growth and continuous use of IoT devices, traditional cloud computing networks will soon be overwhelmed by the data traffic due to having centralized distant servers as well as causing a delay due to data transmission across long distances. To address this issue, we will use edge computing.

Edge computing has brought computation and data storage as close as possible to the network's edge, thus minimizing the distance between client and server [4]. According to Satyanarayanan [5], IoT devices forward their traffic to nearby edge nodes/servers to perform data processing tasks and provide IoT-edge-centric services locally or in nearby edge nodes using content delivery network (CDN) concepts with the cloud infrastructure. It provides multiple paths for devices to connect to the Internet and allows most of the data to remain scattered across distributed networks of the server while some necessary/valuable data is sent to the cloud server after being processed and refined. Hence, due to IoT data storage, processing, and analytics at the network's edge, the load on the cloud is reduced as well as reduced chances of data privacy leakage as all private information does not have to be moved to the cloud servers due to data being stored locally or in edge nodes. Since edge computing has decreased the physical distance for data transmission, latency is reduced and

response speed is higher compared to cloud computing. In addition, since only the required data is sent and stored in the cloud server, there is less redundant data storage and energy consumption in cloud servers along with reduced bandwidth costs. Moreover, due to the presence of edge data centers, even if the cloud server faces an outage or a failure, the IoT devices will still be able to work effectively because they can still perform various functions by themselves by rerouting data through multiple pathways and hence, users will still be able to access the information they need [6].

However, edge computing has a major issue regarding security. Edge computing security issues concern the end-to-end devices and the network in between. According to Acken and Sehgal [7], since the edge computing framework is more widely distributed, controlling access via identification authentication is especially difficult in an IoT environment in addition to preventing hackers from stealing or manipulating private data since due to attack surfaces available, it is possible for hackers to hack the device from any channel and hence it is difficult to track or trace them as well. Furthermore, the data produced by IoT devices are divided into several parts and stored within different edge servers that are located in separate locations, which makes it difficult to ensure data integrity due to data loss and may lead to erroneous data storage in edge servers. However, we may be able to prevent cyber attacks and information theft by establishing smart contracts with devices with a history of successful connections/logins/transactions via blockchain along with the application of cryptographic techniques.

For data encryption and decryption purposes, we use two types of cryptography: symmetric key cryptography and asymmetric key cryptography. Symmetric key cryptography is a cryptographic technique that uses a single shared secret key to encrypt and decrypt data for all parties involved. The AES (Advanced Encryption Standard) algorithm is an example of symmetric key cryptography used in both hardware and software to encrypt susceptible data in blocks of 128 bits and consists of three block ciphers which are AES-128, AES-192, and AES-256 and they have a key size of 128, 192, and 256 bits respectively. It is not recommended to be used alone because, since the key is shared between the involved parties, if the key comes into the possession of a hacker somehow, they would be able to decrypt any file encrypted with that secret key and different keys would have to be generated for communicating with different people or groups of people. On the other hand, asymmetric key cryptography is a cryptographic technique that provides better security because a public-private key pair is used where the public key is open to all for encrypting data whereas the private key is kept secret by the user and is used for decrypting data [22]. The RSA (Rivest–Shamir–Adleman) algorithm is an example of asymmetric key cryptography used in modern computing systems where the algorithm is based on Euler's theorem and a public-private key pair is produced with key sizes typically of 512, 1024, 2048, or 4096 bits. Kartik et al. [20] proposed in their work that using AES and RSA algorithms together will ensure the integrity and confidentiality of data in storage systems.

A blockchain is a shared, decentralized digital ledger based on a P2P system that can be openly distributed among individual clients to create an unchangeable record of transactions where each transaction is time-stamped and connected to the previous one. Data is added as a new block to a continuously expanding chain each time a set of transactions is added where each block is connected to its prior one except for the

genesis block which is the first block created in the blockchain [12]. However, to add a block to the blockchain, a consensus mechanism is used by a network of participant nodes to work together to authenticate and validate a piece of data or a proposed transaction before adding it to the blockchain to provide secure and trustworthy storage of data [25]. This process is known as mining and miners are compensated in cryptocurrency in exchange for processing everyone's transactions and appending validated transactions to the network which requires very complex calculations that need massive amounts of processing power to run. Blockchain data is usually stored within many devices on a dispersed network of nodes due to which both the system and the data are highly impervious to technical issues and malicious attacks. Each node on the blockchain network makes a copy of the database and stores it, ensuring that there is no single point of failure. In contrast, standard databases usually depend on a single or a few servers and therefore are more susceptible to technical problems and cyber-attacks. However, blockchain has some demerits as well. As it uses asymmetric cryptography to allow clients to obtain ownership over their cryptocurrency units or any other blockchain data, if the private key is lost, then clients can no longer access their funds, and thus that the money is lost with no way to regain it. In addition, blockchain storage is limited, and storing huge amounts of file data in the blockchain causes access latency, expensive transaction costs as well as introduces the risk of losing nodes in the network [11]. However, to resolve this issue, different decentralized storage systems can be used to store huge amounts of data as well as to share files with other people. Among the different platforms available, the InterPlanetary File System (IPFS) is a peer-to-peer, distributed, open-source, content-addressed file sharing and storage system. Its goal is to replace HTTP to build a better web and instead of a central database/server, it is built around a decentralized system of storage providers, creating an effective system of file storage and sharing. When files are uploaded on IPFS, a cryptographic hash of the file is created which uniquely identifies the file and hence prevents duplicates from being stored. Any user in the network can retrieve the file using the file hash due to the distributed hash table present [16].

Blockchain is evolving at a rapid rate with its use spreading in many systems. It has found its use in many industrial sectors such as banking, healthcare, e-commerce, property, retail, trade, media and entertainment, and automotive where its implementation varies depending on the area it will be used in and the purpose it will be used for [8]. In Deloitte's 2018 global blockchain survey [9], it was stated that blockchain technology has enabled faster business process execution in comparison to existing systems due to its improved trustworthiness and ability to keep a history of records. Although centralized databases which are built on more mature technology are faster when it comes to transactions performed per second, they fall short when it comes to evaluating the end-to-end journey. With its chain of blocks containing information, blockchain acts as a database that is consensually shared in a decentralized manner and synchronized across various places and institutions for providing access to multiple people. Once data is inserted into a blockchain, it becomes extremely hard to alter it. Each block contains valuable data as well as the hash of the current block and the previous block [14]. The blockchain keeps track of information regarding its transactions, such as the sender, the receiver, and the amount of money. It also uses a hash like a fingerprint to uniquely identify a block and all of its data. After the creation of a block, its hash is calculated, and altering

anything within the block will cause the hash to change as well due to which it will no longer remain the same block and match the hash stored in any of the following blocks, thus making all subsequent blocks invalid. To make the blockchain work again, we have to tamper with all of the following blocks and recalculate all their hashes but it is not possible to do so due to consensus mechanisms, thus making the stored data unchangeable. Hence, hashes are very convenient in detecting any alterations in the data of a block.
Blockchain storage works by first sharding data and distributing the shards across thousands of nodes after which each data shard is encrypted on the local system to guarantee that no one can access the data shard except for the user to whom it belongs. Next, the blockchain storage system generates a unique hash and encrypted output string depending on the data shards or encryption keys. The hash is appended to both the ledger and the shard metadata to link transactions to the stored shards following which the system makes a copy of each data shard a specific number of times to make sure that sufficient copies are present to avoid the loss of valuable data. After that, a P2P network distributes the replicated shards to spatially scattered storage nodes belonging to individuals called farmers with the data still being accessible to its owner. Lastly, the storage system records all the information of the data stored or removed [15].
The blockchain architecture can be public or private. A private blockchain uses access control to limit the users who can participate in the network as well as the transactions they can perform while a public blockchain is permissionless and decentralized, allowing anyone to access the blockchain network while maintaining anonymity to read and write on the blockchain. Ethereum and Bitcoin are the two most popular examples of a public blockchain. Ethereum is a blockchain-based decentralized, open-source computing platform that allows the creation of smart contracts as well as decentralized applications (Dapps) [19]. In an Ethereum blockchain, a smart contract refers to a program that is stored at a specific address within the Ethereum blockchain and is used to execute an agreement between two people. Rather than being controlled by a client, smart contracts are deployed to the network and run like they are programmed to [21]. The Ethereum blockchain requires fees known as gas to operate due to avoiding network abuse problems and evading the questions coming from Turing completeness. Therefore, for any given amount of programmable computation in the Ethereum blockchain, there is a universally agreed fee represented by gas and it does not exist outside of the execution of a transaction. Every transaction comes with a specific amount of gas known as the gas limit. This refers to the maximum amount of gas you are willing to use on a transaction and it allows any unused gas at the end of a transaction to be sent back to the sender's account at the same rate of purchase. The purchase happens at the gas price which is the amount paid for each unit of gas and if the account balance is not enough to pay for a purchase, the transaction is considered invalid. Ethereum uses the cryptocurrency Ether to purchase gas consumed in a transaction and is delivered to the address of an account usually under the control of the miner. However, miners are free to select which transactions to ignore and transactors are free to state any gas price that they want. Senders will use more Ethers on a transaction with a higher gas price and provide more value to miners, and thus more miners will likely want to add the sender's block to the blockchain.
Various devices in the modern age such as mobile phones, laptops, PC, sensors, ac-

tuators, etc. are connected to the Internet via IoT systems to sense, communicate, and exchange information to reach a particular goal and blockchain can be used to provide security to such systems as well. Ramesh [18] proposed in his work a way to collect sensor data from IoT devices and provide secure, decentralized storage within a closed system using blockchain which can be applied to companies or industries like shipping where data needs to be shared as it reduces latency and lessens the dependence on cloud-based systems. In addition, he also discussed the performance of distributed systems like Ethereum Swarm and IPFS on low-powered devices such as Raspberry Pi was also compared there. However, to use blockchain in IoT systems, sufficient computing resources, and energy of IoT devices are required because of the high amount of processing power needed to be consumed for the mining process [26]. In Li et al.'s work [10], a blockchain-based decentralized federated learning framework is proposed which uses a committee consensus mechanism. This framework addresses the security concerns regarding continuous attacks by malicious clients or cloud servers on the global model or private data from the user's IoT device by replacing the cloud server with a blockchain that stores the global model and performs the local model update exchange as well using an efficient and secure committee consensus mechanism to effectively lessen the amount of consensus computing and ensure a decrease in malicious attacks.

Incorporation of edge computing with blockchain can be done to allow IoT devices to enhance their computing capability by offloading storage and processing tasks to nearby edge servers located at the base stations of radio access networks. In turn, blockchain can also alleviate security issues by hiding private client information and providing reliable access and control of the network through dispersed edge servers and cloud servers. However, according to Luo et al. [27], this integration presents several challenges that regarding scalability, consensus optimization, interoperability and cost standardization, the security of the blockchain itself as well as security challenges like communication security and protecting the privacy of devices. Despite the challenges presented in integration, some systems have been developed from the successful incorporation of blockchain and edge computing. Rahman et al. [28] present an in-home therapy management system where patients receive therapy from professional healthcare providers/therapists through the use of gesture-tracking IoT devices, along with other in-home sensors to collect multimedia data and a blockchain-based mobile edge computing (MEC) framework supporting anonymous fast, safe, spatiotemporal multimedia data communication which is available at all times and shares data whenever it is needed. In the work done by Nyamtiga et al. [17], an IoT design was discussed which achieves the necessary security and scalability levels for the successful incorporation of blockchain and edge computing to allow safe storage of IoT data and transactions in IoT systems. In our research, we investigate the different approaches used in other works and derive the most suitable approach that we can model our system on.

Chapter 3

Proposed blockchain-based edge computing framework

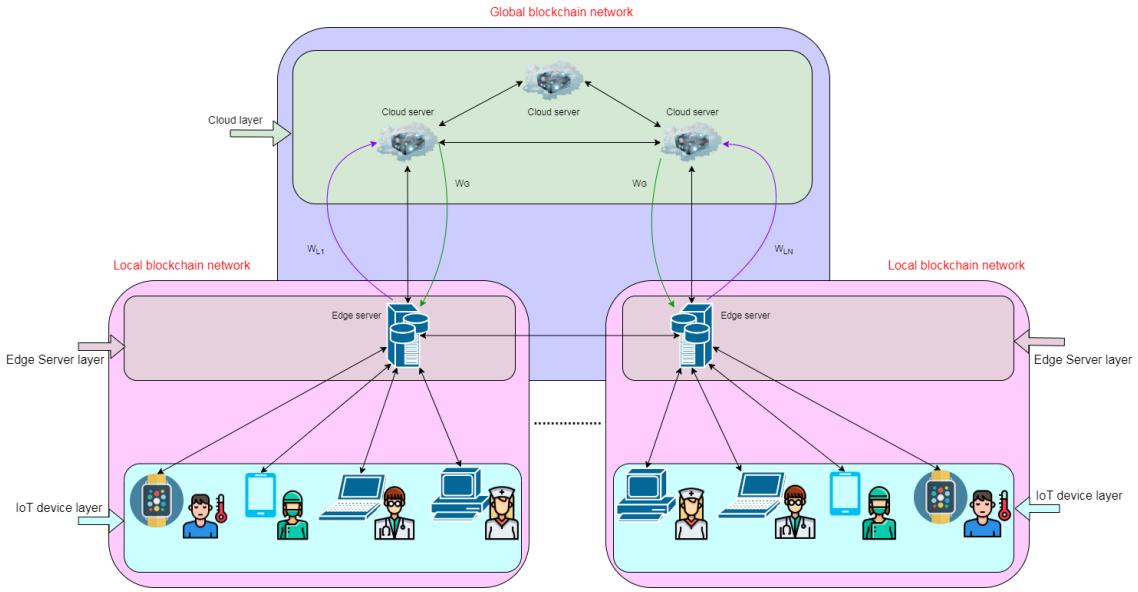


Figure 3.1: Blockchain-based edge computing framework

Figure 3.1 illustrates our proposed framework which supports both data storage and federated learning for data processing and analysis with minimal cost and latency while addressing privacy and security concerns. Our framework can be split into the following components:

3.1 Cloud layer

In edge computing, the topmost layer is the cloud layer which consists of multiple cloud servers communicate with each other and provide cloud storage and processing services in a centralized location. Although cloud servers may have large storage and processing power compared to edge devices, due to their centralized/ remote nature, there is latency and higher cost due to data transmitting from a distant remote location. Hence, this layer provides large storage space and performs big data analytics for the edge networks with limited resources. In addition, cloud

servers are generally equipped with better security policies than edge servers to stop the spread of malware. Hence, some data is sent to the cloud server periodically to be processed or stored while some data is processed and stored in edge devices. Cloud servers broadcast the weights of the global model used for federated learning as well as use the local model weights of each client to perform model aggregation via the federated averaging process to update the global model.

3.2 Global blockchain network

Among the cloud servers, there exists a global blockchain network. Each cloud server has a copy of all the blockchain data generated by every client and from the blockchain, the local model weights for each client can be retrieved so that they can be used for model aggregation in federated learning. Data cannot be tampered with once a data block is added to the blockchain because any block points to the hash value of its previous block and thus a change in data will cause a change in hash value which can be easily detected. Moreover, cloud servers can use the PoW (Proof of Work) consensus algorithm to update the global blockchain since we are using Ethereum blockchain which currently only supports PoW and PoW requires large amounts of computing power to solve a mathematical puzzle before adding a block to the chain.

3.3 Edge server layer

It consists of a set of edge servers which can be single-board computers such as Intel Xeon, Brix, or Raspberry Pi 3 are deployed at the base station to form an edge computing network. Edge servers locally manage registered IoT devices involved in healthcare and are responsible for delivering data to the user device when required as well as interacting with cloud servers in the upper layer to transfer data in the local edge network periodically to the cloud server for further processing or storage. IoT devices are registered by edge server nodes to avoid being added to the edge network without the permission of edge server nodes. This layer provides storage and computing facilities within a P2P network of edge servers to provide storage and prevent data loss due to a single point of failure which would be the case if we relied just on a centralized cloud server. It distributes computing and storage resources near the edge provides services locally in a reliable manner, and reduces latency and cost due to the distance required to travel being reduced. On top of providing the required resources to IoT devices close to the network's edge, the edge servers can also pass messages amongst themselves to allow each edge server to have a copy of data of other edge servers including the blockchain data stored in other edge servers and manage data processing tasks. In federated learning, edge servers create local models for each client and these models are trained on their respective client data to send local model weight updates for each client to the cloud server.

3.4 Local blockchain network

Between the edge center and the registered IoT devices, there exists a local blockchain network to allow secure transmission and storage of data where client data is en-

crypted and stored in IPFS while the IPFS hash is stored in the client's respective local blockchain. Healthcare-related IoT devices can also communicate with each other through edge servers for data sharing purposes. PoW is also used here to validate transactions and add new blocks to the local Ethereum blockchain. Edge servers periodically upload the data in local blockchains to the global blockchain by dividing the data into blocks and then to ensure that no changes are made to the data when uploaded to the cloud server, the hash values of these blocks are calculated by the cloud server and is compared to the hash values stored in the local blockchain. For federated learning, client data can be retrieved from the local blockchain so that the client's corresponding local model can train on it for data analysis tasks.

3.5 IoT device layer

This layer consists of IoT end devices that can be used for obtaining medical information such as smartphones, PCs, smartwatches, and laptops that use the application to store and share data with other users as well as analyze the data stored via blockchain, edge computing, and federated learning. As healthcare-related IoT devices have limited resources and capabilities, medical data is sent to edge servers through blockchain for storage and processing tasks.

Chapter 4

Using Blockchain for Security & Data Preservation

To explain our use of blockchain in our system, our total work can be divided into 5 segments in this chapter. First of all, we discuss the variables and methods within our smart contract. Then, we discuss the various applications of our smart contract in the blockchain.

4.1 Within the Ethereum smart contract

For coding our smart contract, we have used Solidity programming language which is a high-level, contract-oriented programming language used for writing Ethereum smart contracts for Dapps, and the smart contract is written in a browser-based IDE called Remix. Our smart contract for storage contains various methods and variables related to storage and retrieval as well as verification.

Figure 4.1: Inside the Storage smart contract

4.2 Account access

People provide info to create their accounts and become part of the storage network. These people can upload, download and share files with other users to allow them to access important medical data. When users want to be a part of this system, they must first register themselves by providing some mandatory information such as their password, phone number, and email. After submitting the required registration info, each user is assigned a unique user ID along and their device is registered in the network to complete the creation of a user profile. All this information is hashed using the SHA256 algorithm and stored on the blockchain and users then will be able to access their account by providing only their email/phone and password they provided during registration. When the user enters his/her email/phone and password, they are passed via POST requests and hashed. If the hash of the entered email/phone and password are equal to their existing hashes saved on the blockchain, the user will be redirected to the file storage page, or else the user will be redirected to the login screen with a flash message for the incorrect info entered. However, even if the login info is correct, if the device address is not registered as one of the user's trusted addresses, you have to answer a short question, or else you can't be able to access the account due to suspicious access.

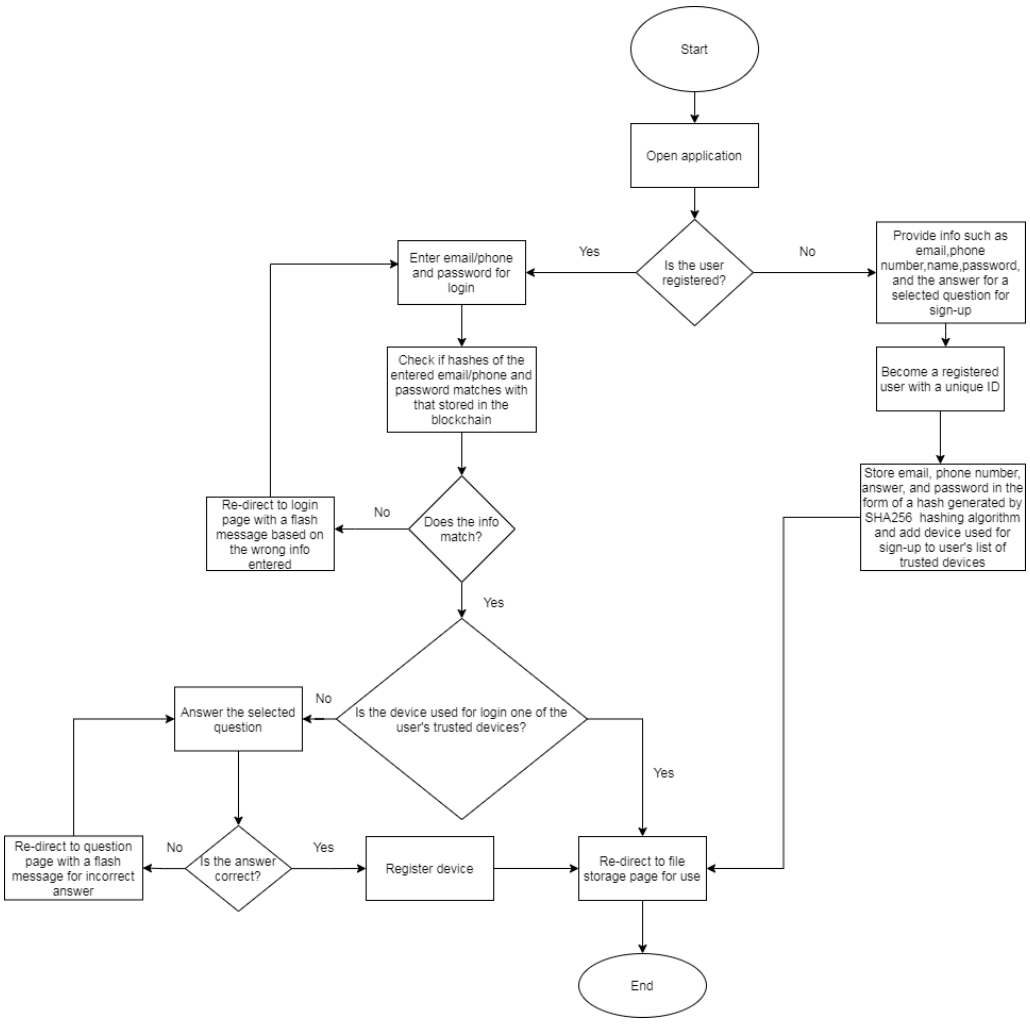


Figure 4.2: Account access

4.3 Storing and retrieving files

Since directly storing entire medical files on the blockchain is a bad idea due to the amount of data capable of being stored limited either by protocols or because of the large transaction fees required to be paid alongside the access latency produced, we have chosen an alternative scheme. In Ethereum, there are data structures known as struct (similar to a JavaScript object) and mapping (like Python dictionaries). The DocInfo struct consists of a boolean variable for the file's existence and a variable for the IPFS hash which is used to store or retrieve data from IPFS. The mapping data structure is used by using the hashed filename as the key where the hash is generated using the SHA256 function and the value is the DocInfo struct. Simply storing files in IPFS and saving the IPFS hash in the blockchain is inadequate because if anyone gets the IPFS hash, they will gain access to that file. Hence, for data encryption, we have chosen the AES-256 symmetric key algorithm which is mathematically efficient and elegant, fast, and highly secure due to the higher-length secret keys (128, 192, and 256 bits). AES keys can be stored in the blockchain to be retrieved or updated when required from the AESkeyStore mapping data structure which uses the AESkeyDetails struct, which consists of the AES key and a boolean value for the key's existence, as the value and a string as the key.
To store a file, firstly, we use the storeFile method to encrypt the file using the AES-256 secret key. The encrypted file is then uploaded to IPFS and in return, we receive an IPFS hash of the file. Lastly, the IPFS hash of the encrypted file is stored in the DocInfo struct in the blockchain along with setting the boolean variable as true since the IPFS file hash now exists in the blockchain, and the struct is inserted into the mapping data structure with the hashed filename as the key.

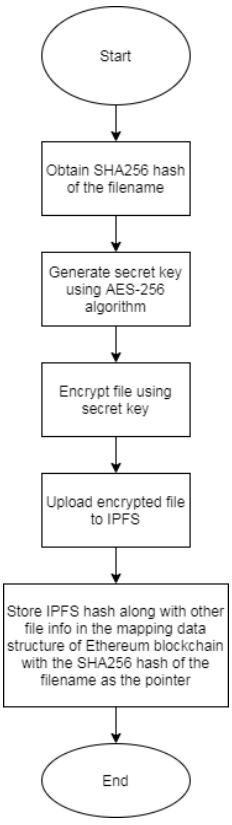


Figure 4.3: Storing files

To retrieve the file, firstly, we retrieve the IPFS hash of the file using the getFile method which checks if the filename hash exists in the mapping data structure, and if it exists, the filename hash is used to retrieve the IPFS hash of the file from the corresponding DocInfo struct. The IPFS hash is then used to retrieve the encrypted file from the IPFS storage. Lastly, we use its AES secret key to decrypt the file, turning it from cipher text to plain text.

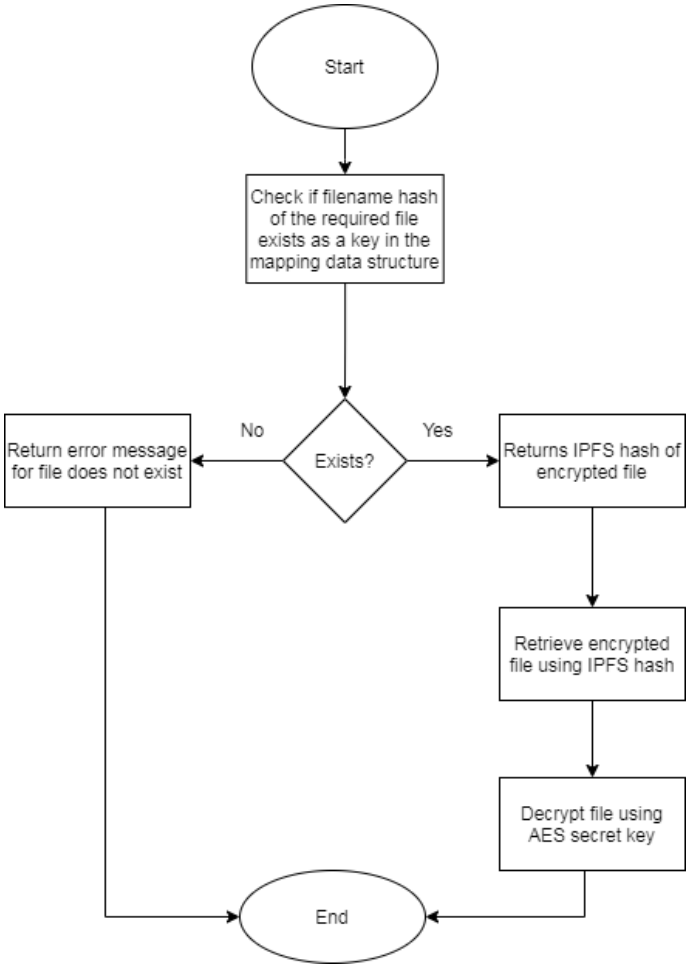


Figure 4.4: File retrieval

4.4 File sharing

Sometimes, we may need to share files with other users but simply using the AES-256 algorithm is not good enough because, like other symmetric key algorithms, a single secret key is used to encrypt and decrypt data and if the secret key is intercepted by a hacker while it is being transferred from the sender to the recipient, the hacker will be able to use it to decrypt the encrypted file. Therefore, to prevent it from falling into the wrong hands, we use the RSA asymmetric key algorithm for encrypting and decrypting the AES key rather than the user data because it is relatively slower and RSA can only encrypt data smaller than or equal to its key length. Using the getFile method, the sender gets the IPFS hash of the file he/she wants using which he/she gets the encrypted file to be sent to the recipient. A public-private key pair is generated on the receiver end using the RSA-1024 algorithm and the AES-256

secret key is encrypted using the RSA public key of the recipient. The encrypted key is sent alongside the encrypted file to the address of the recipient and thus, even if the hacker gets the encrypted file, they will not be able to decrypt the file as it requires its original AES secret key which has now been encrypted by the RSA algorithm. After the recipient receives both the encrypted file and secret key, the recipient uses his/her RSA private key to decrypt the AES-256 secret key which is used to obtain the original file.

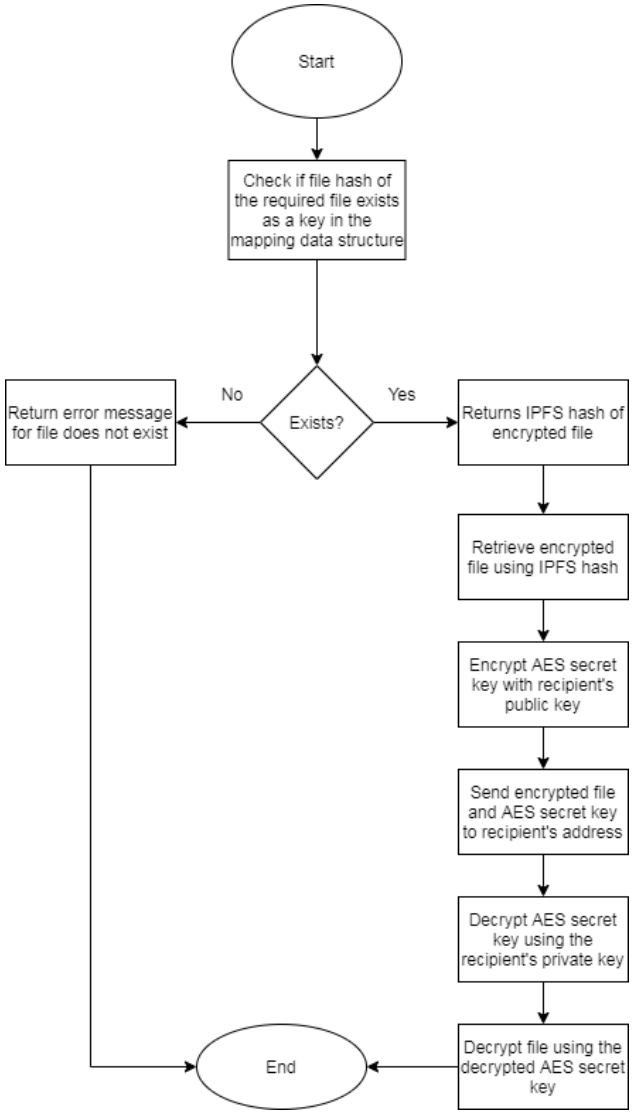


Figure 4.5: File sharing

To cut off the file access, after retrieving the IPFS hash using the filename hash, retrieving the encrypted file from IPFS, and decrypting the file using its AES secret key, the original AES secret key is replaced by another secret key, and the decrypted original file is again encrypted with the new key and stored in IPFS which in return gives a new IPFS hash after which by first using the removeFile method and then the storeFile method, the updated file information is placed into the mapping data structure of the Ethereum blockchain and the new key replaces the old key in the Ethereum blockchain by using the updateKey method.

4.5 Storing local model weights for FL

In our federated learning algorithm, the global model weights are broadcasted to the edge servers where these weights are used by the local model for each client for training on the client data and updating their weights. These updated local weights are then sent to the cloud servers and undergo the federated averaging process where a new global model is produced with adjusted weights and this process is repeated several times. If a hacker were to alter these local weights, it would severely impact the performance of the neural network and therefore, to protect these weights, a separate AES-256 secret key is created to encrypt the weights and store them in the blockchain for each client as bytes from an array using the setWeight method and the AES-256 secret key is encrypted with the RSA-1024 public key of the recipient cloud server. When the weights are required to be fetched, the AES key is decrypted using the RSA private key of the recipient server and then the AES key is used to decrypt the weights returned using the getWeight method, and lastly, the weights converted back to their original array format.

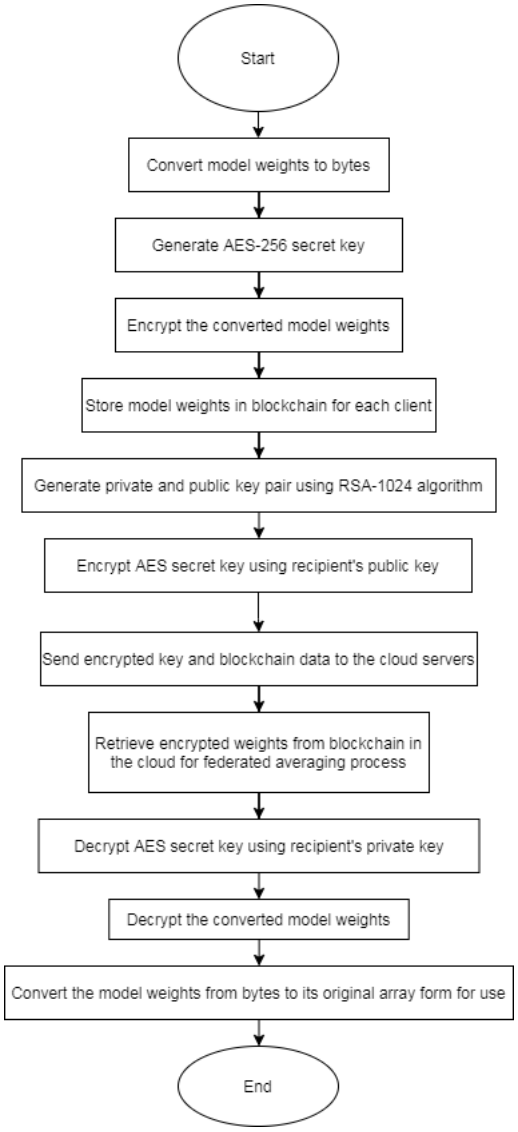


Figure 4.6: Storing weights

Chapter 5

Analysis & Results

5.1 Evaluating the application of blockchain in the proposed framework

Similar to how cars require fuel to operate, the Ethereum blockchain requires fees known as gas to successfully perform transactions or execute a contract. It measures the amount of computational work necessary to execute particular operations on the Ethereum network. A gas limit is specified for each transaction to set the maximum amount of gas available for consumption and any remaining gas from transactions is returned to the sender's account. In our program, we have imported the eth_tester and web3 packages to test our blockchain code and set a gas limit of 10000000000 to avoid any errors occurring due to insufficient gas remaining in the account after each transaction is performed. In our code, get-methods contain the function type keyword 'view' since they do not change the state of the contract and is only used for retrieving and thus we use the call() function instead of the transact() function in the case of get-methods as no transactions are created here. The following graph shows the methods that alter the state of the contract and thus uses the transact() function to generate transactions and consume gas for execution on the blockchain.

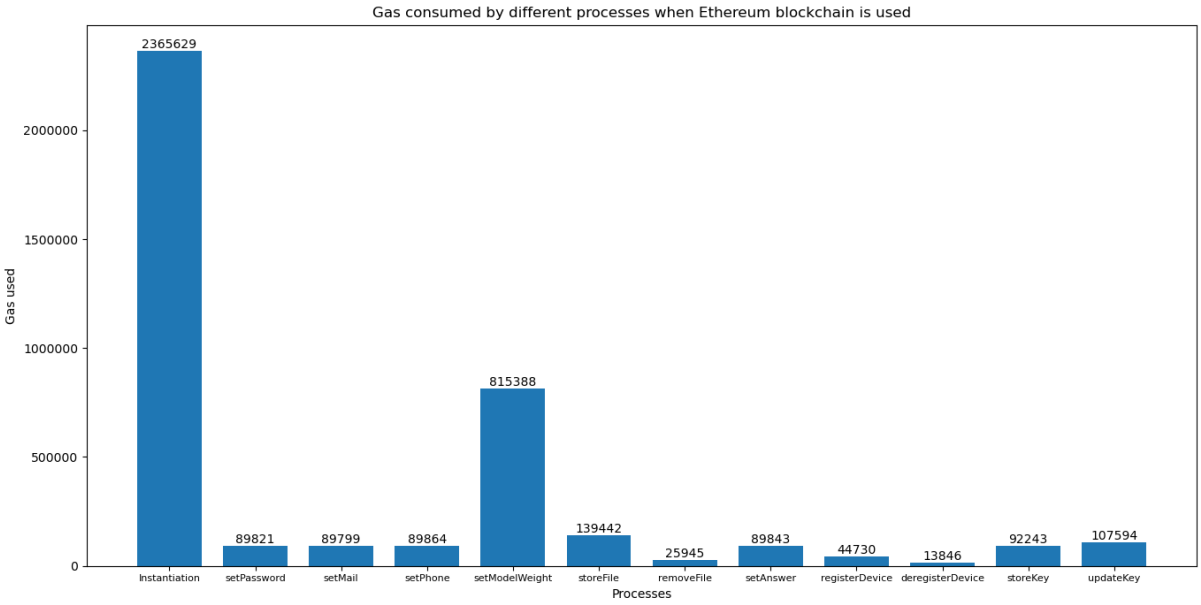


Figure 5.1: Gas consumption of smart contract processes on transactions

Since any transaction on the blockchain uses gas, from the graph we can see that the deployment or instantiation of smart contract uses the most gas out of any of the other processes available in the smart contract. This means that to deploy the smart and successfully perform its transaction, miners will need the most computational effort here. In contrast, the miners will need the least computing energy when it comes to de-registering devices that can read or write to the system. Since gas is used by the contract to compensate miners who secure a particular transaction on the blockchain, the process of de-registering devices faces the greatest risk for transactions not occurring because, if the amount of gas is low, miners receive comparatively lower compensation for their efforts due to which they may abandon the job. The method for setting model weight consumes the second most amount of gas which means that it will require the second most amount of computing energy to storage the large array of neural network weights in the blockchain. The methods for setting password, email, phone number, and answer requires similar units of gas when compared to the rest of the functions performed in the blockchain and hence require similar levels of computational work.
Similarly, we also measure the performance of different processes in our smart contract which are used for the Ethereum blockchain network using the average time taken to complete each process as shown in the following graph:

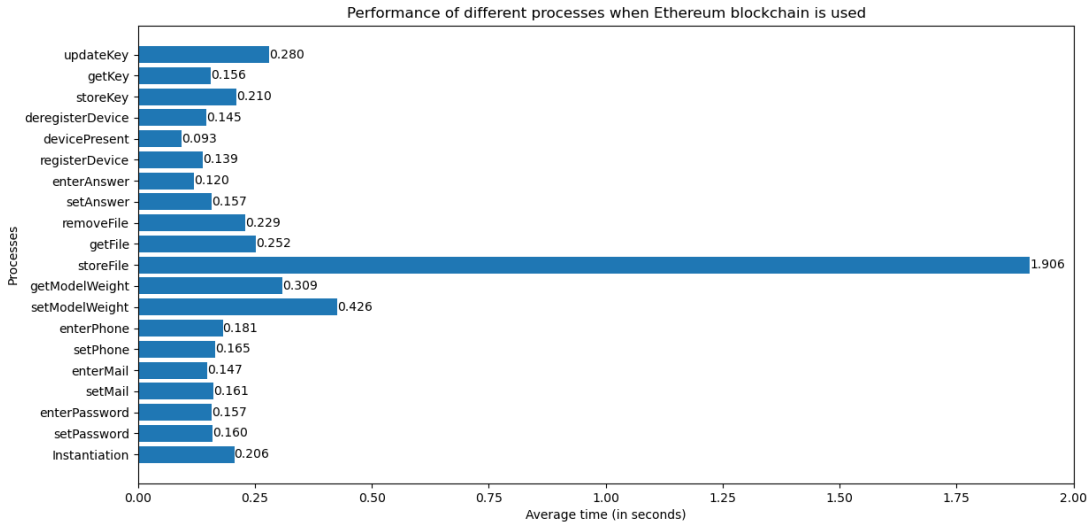


Figure 5.2: Performance of smart contract processes

Among the processes here, the devicePresent method has the best performance as it is the fastest with an average time of 0.093 seconds. Like before, the setModelWeight method is in second but here it is in terms of performance with an average time of 0.426 seconds. In addition, the methods for setting the password, email, phone number, and answer as well the methods for retrieving AES key and entering passwords have average time values very close or equal to one another indicating that they have similar performance. Instantiation of smart contract and the method for storing AES key has similar performance as well along with the registerDevice, deregisterDevice device, and enterMail methods. The storeFile method has the worst performance as it is the slowest with an average time of 1.906 seconds and in

testing this method, we have used our dataset file which has a file size of 122KB.

5.2 Federated learning with blockchain

Federated learning refers to machine learning on decentralized data with privacy by default. Rather than sending data to a centralized server for training, models are sent to edge nodes to be trained locally using the local data of clients, and data at edge nodes never leave the device; only the new model parameters leave the device [30]. However, model parameters are susceptible to alteration or theft by hackers, and hence we the Breast Cancer Wisconsin (diagnostic) dataset [31] to perform federated learning with blockchain to address security and privacy concerns in addition to enhancing performance.

5.2.1 Data pre-processing and train-test split

Feature Selection

Feature selection is used to remove redundant features from our dataset so that we can train our algorithm only on important features and thus remove the complexity from our model and improve its performance. Strongly correlated features are removed using feature selection because they bring similar types of information into the algorithm and the performance of the model will be impacted by multicollinearity. Thus, this will allow a change in one feature to bring about a huge change in another feature. The stronger the correlation, the more difficult it is to change one feature value without it affecting the value of another feature and thus it becomes difficult to differentiate between the individual effects of the independent feature on the dependent output variable as the values of strongly correlated features change in unison. This can lead to overfitting and problems with interpreting the results, therefore, reducing the model's performance. Therefore, we used a threshold value of 0.8, and any features with correlation values greater than or equal to this value are considered to be strongly correlated and the corresponding feature columns were removed from the dataset.

Stratified train-test split

After feature selection, we make separate numpy arrays for feature data and label data from the original dataset. Next, since the label values are strings, we convert them into integer values of 0 and 1 for binary classification using the LabelEncoder class from scikit-learn. Finally, due to some classification problems not having a balanced number of examples for each class label, we have used stratified train-test split to divide the original dataset into the training and test dataset in a 7:3 ratio such that it preserves the same proportions of examples in each class like in the original dataset.

Feature Scaling

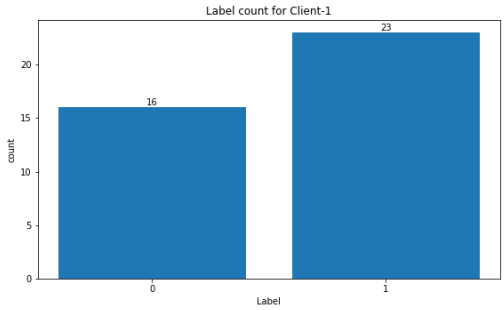
Feature scaling is a data pre-processing technique that is used to normalize the range of features of data. It is important because most of the time, our dataset will contain features that are highly differing in magnitudes, units, or a range of values

due to which the algorithm may end up assigning higher weights to features with greater values regardless of the unit of these values as a change in that feature can be considered more important than the other feature and hence the objective functions may not work properly. We have used standardization as our feature scaling method which re-scales each feature in our dataset, ensuring each feature has a distribution with a mean and the standard deviation of 0 and 1 respectively, and it is useful for optimization algorithms used in neural networks. It uses the equation:

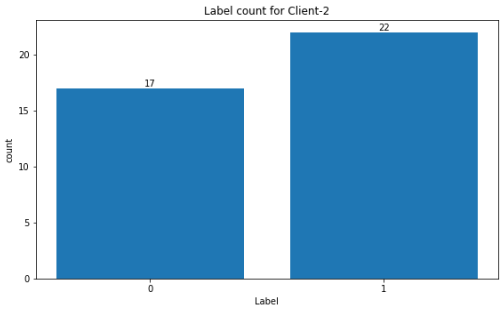
$$x_{std} = \frac{x_i - \overline{x}}{\sigma} \tag{5.1}$$

5.2.2 Creating clients

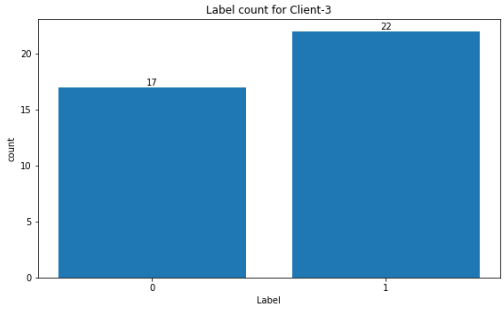
Just like in a real-life implementation of federated learning, each client has their own data and training has to be carried out in a decentralized manner as the system will not have direct access to all the training data. In this step, we create clients and the data they have by sharding the training data since each client will have their own separate data. Sharding (also known as horizontal partitioning) allows us to break our data into smaller chunks or shards to be stored in separate servers, just like breaking up a single large table to create multiple tables with different rows but the same schema and columns. Here, we have used the training data to create 10 data shards, one for each client and a dictionary is created for the 10 clients where each client's id is the key and their shard of the training data is the value. The graphs below show how the label counts for each client according to the data shard they possess:



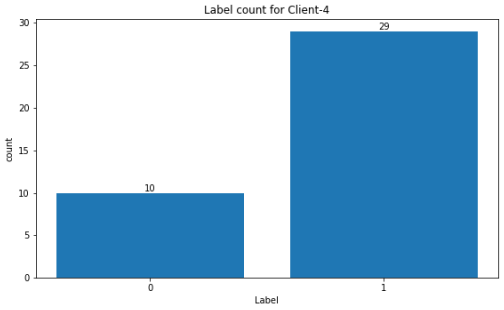
(a) Label count for client-1



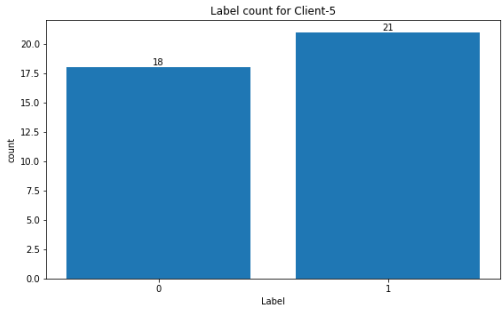
(b) Label count for client-2



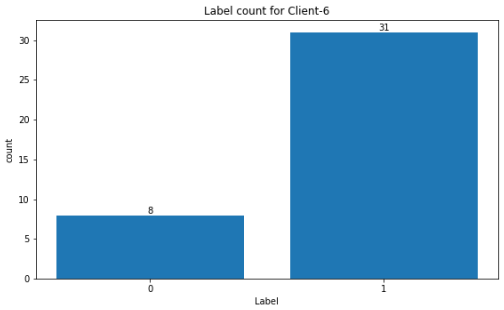
(c) Label count for client-3



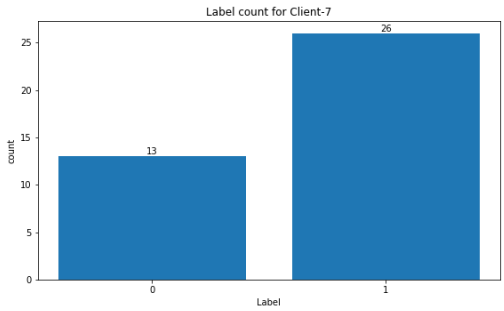
(d) Label count for client-4



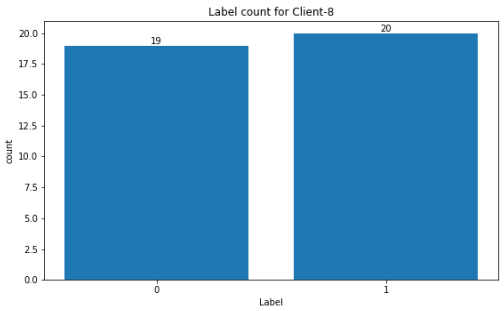
(e) Label count for client-5



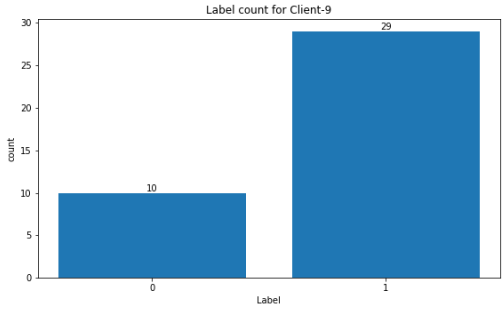
(f) Label count for client-6



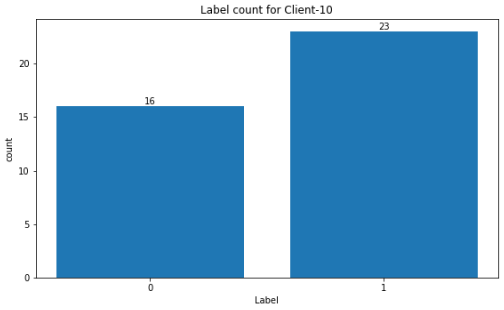
(g) Label count for client-7



(h) Label count for client-8



(i) Label count for client-9



(j) Label count for client-10

Figure 5.3: Label counts of clients created using training data

5.2.3 Processing and batching clients' and test data

After creating our client dictionary which consists of client ids as keys and their corresponding training data shards as values, for each client, we create a TensorFlow dataset using their corresponding data shard and then batch them. Batching is necessary because the data shard of each client can have a huge size and if you load the entire data into the memory, the training time of the model will be very long because a lot of memory is used in your CPU which is very inefficient and hence it is very costly for the computer. Moreover, if the neural network has to store error values for all consecutive elements of the dataset in the memory, training will become even slower. All of this can also lead to out-of-memory errors. To ensure faster training, prevent out-of-memory errors, less noisy gradients, and perform highly optimized array operations, batching is used and the model updates its hyperparameters after completing each batch. Like with our client data, we also do the same for our test data. We have used appropriate batch sizes so that overfitting of data does not happen.

5.2.4 Creating the Multi-Layer Perceptron

For binary classification, we have chosen the Multilayer Perceptron (MLP) model because our data is neither sequential nor does it have a spatial relationship. The multilayer perceptron is a fully connected, feedforward artificial neural network (ANN) composed of perceptrons which gives a solution to problems that need complex calculations. A perceptron is a neuron that can be defined as a linear classifier with one or more input connections with each connection having a weight/parameter learned from training, an activation function and produces a single output based on the result of the activation function on the sum of products of inputs, the weight of their connections, and biases from their predecessors in the previous layer since MLPs are fully connected. An MLP is typically composed of 3 types of layers: an input layer providing input signals from the data used for the model, an output layer which is the final hidden layer that outputs a prediction based on the provided input, and in between those two, an arbitrary number of hidden layers which play a critical role in the operation of an MLP. A network can consist of multiple hidden layers depending on the complexity of its functions. The hidden layers are called so because the results they produced are not displayed as their outputs are used as inputs for the output. Data flows from the input layer to the output layer and throughout the training process of an MLP, forward and backward passes occur. In the forward pass, using feed-forward propagation, the input signal flows through the network layer by layer from the input layer until it reaches the output layer, and the result of the output layer is measured against the desired response. In the backward pass, using backpropagation, partial derivatives of the error function with respect to the various biases and weights are propagated backward through the MLP layer by layer on comparing the output of the MLP with the desired response to make successive adjustments to the parameters of the model to minimize error [32]. The forward and backward pass continues until the network reaches the convergence state when the error can go no lower.
For federated learning, we have used an MLP with two fully connected hidden layer with the same number of neurons as the number of remaining features after feature selection and the output layer consists of a single neuron to make predictions.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 13)                182
_________________________________________________________________
dense_1 (Dense)              (None, 13)                182
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 14
=================================================================
Total params: 378
Trainable params: 378
Non-trainable params: 0
```

Figure 5.4: Model summary of MLP for binary classification

In our federated learning architecture, each global server has a global MLP while each client of the network has a local MLP which is trained using local data provided by clients. The architecture is depicted in the following figure:

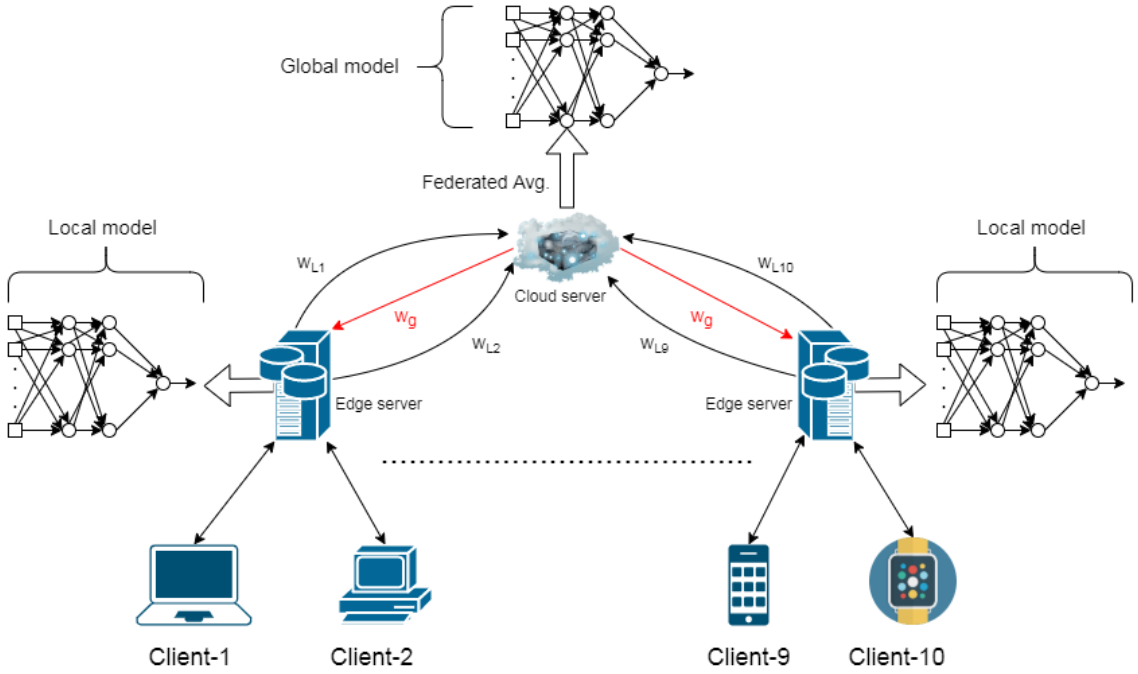


Figure 5.5: FL architecture with MLP neural network

In our MLP model, we have used the following:

- Sigmoid activation function: It is a type of activation function, which limits the output to a range between 0 and 1, and hence it is also known as a squashing function. It is usually used for supporting non-linear behavior and it is the correct choice for the output layer of a binary classification problem like ours since its output can be treated as probabilities of a data point belonging to a particular class as the input to the function is transformed into a value between the range of 0 and 1. One of its big advantages is that the derivative of a sigmoid function can be represented using the sigmoid function itself. The sigmoid function for all possible inputs results in an "S"-shaped curve and can be expressed by the following equation:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.2}$$

- ReLU activation function: It is a type of activation function that will output the input directly if the input value is positive, otherwise, it will produce an output of zero. It can be represented by the following equation and graph:

$$y = \max(0, x) \tag{5.3}$$

  Many neural networks use ReLU because it is easier to train models with it and hence they often perform better. This is due to the fact that the ReLU function does not activate all the neurons at the same time because if the

output of a linear transformation has a value less than 0, only those neurons will only be deactivated [33]. In our work, we have used the ReLU function for the hidden layers.

- Binary cross-entropy loss function: It measures the performance of a binary classification model. Loss functions typically measure the deviation of the output from the true result and the greater the deviation, the larger the loss. The binary cross-entropy loss function compares the predicted output to the actual class output which can be either 0 or 1 and then calculates the deviation from the actual value [34]. It calculates the loss of an example using the following equation where N is the output size:

$$Loss_{BCE} = -\frac{1}{N}\sum_{i=1}^{N} y_{true}\log(y_{pred}) + (1 - y_{true})\log(1 - y_{pred}) \tag{5.4}$$

- Adam optimizer: For our model, we used the Adam optimizer. It is a variant of stochastic gradient descent (SGD). SGD has a fixed learning rate which is why it takes longer for it to iterate over all the data and reach convergence. In addition, newer approaches can outperform SGD in terms of cost function optimization. Adam makes use of AdaGrad and RMSProp algorithms which are also variants of SGD by combining the best of both their properties and thus it is an optimization algorithm that is computationally efficient, has faster training time and better accuracy scores compared to the other optimization algorithm mentioned, suitable for problems that have very noisy/or sparse gradients and also are large in terms of data and/or parameter [35]. Also, since clients store huge amounts of data in a data/file storage system, Adam optimizer is preferable.

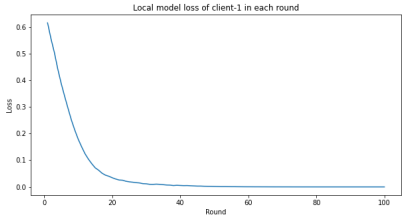
5.2.5 Federated model training with blockchain

Every client on the Ethereum network has an account. The cloud servers broadcast the weights of the global MLP model to all the edge servers and a local MLP model created for each client. Using each client's data stored in their local blockchain, we train their respective local models and place its updated weights on the blockchain after encrypting it using its AES-256 key. The AES-256 key itself is also encrypted using the public key generated from the RSA-1024 algorithm implemented on the recipient cloud server and stored in the blockchain. The local blockchain data is uploaded to the global blockchain and in the cloud servers, the AES-256 key for each client is retrieved from the blockchain and decrypted using the RSA private key, and the local model weights obtained from training are retrieved from the blockchain and then decrypted using the AES-256 key for model aggregation via federated averaging. We use the following equation for federated averaging:

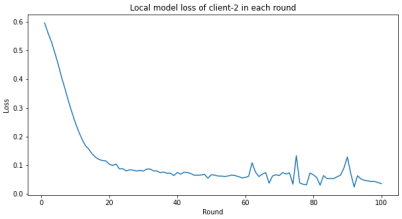
$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k}\sum_{i \in \mathcal{P}_k} f_i(w) \tag{5.5}$$

The equation on the right-hand side estimates the model weights for each client based on the loss values found across every data point they trained with. On the other

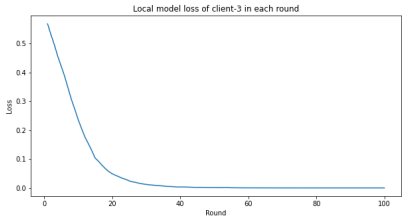
hand, the equation on the left-hand side involves scaling each of those parameters and summing them all component-wise. The updated weights obtained from training the local model are in fact the scaled model weights and in the central/cloud server, we obtain the average weight/gradient over all the local models by summing the scaled weights of all clients. This average weight is then used by each local model for the next training round.



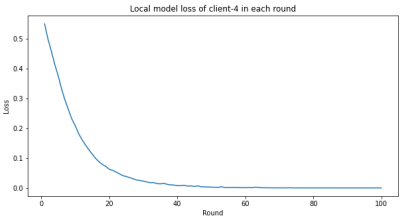
(a) Loss of Local Model for client-1



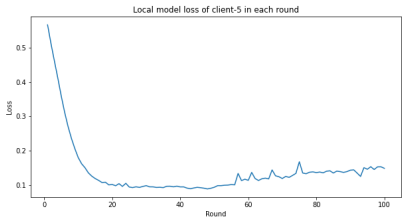
(b) Loss of Local Model for client-2



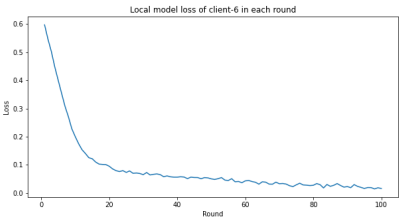
(c) Loss of Local Model for client-3



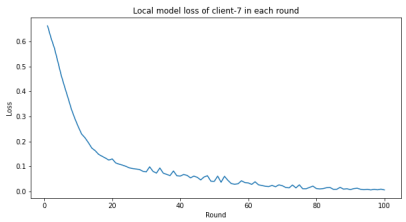
(d) Loss of Local Model for client-4



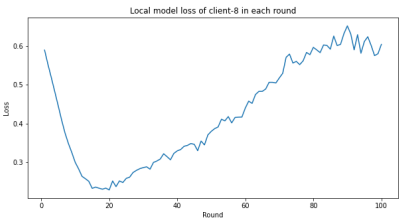
(e) Loss of Local Model for client-5



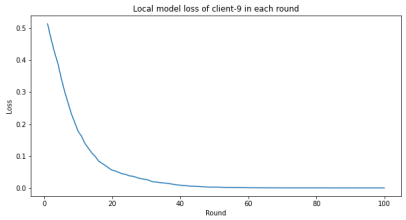
(f) Loss of Local Model for client-6



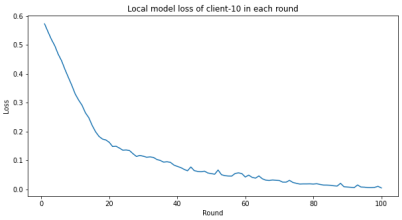
(g) Loss of Local Model for client-7



(h) Loss of Local Model for client-8

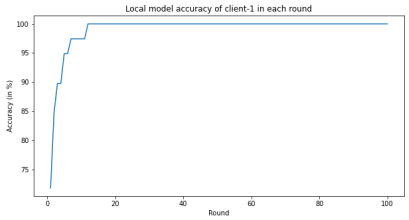


(i) Loss of Local Model for client-9

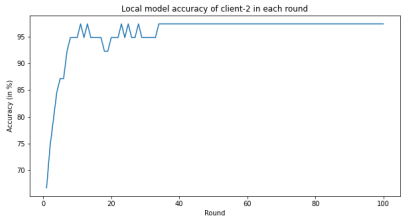


(j) Loss of Local Model for client-10

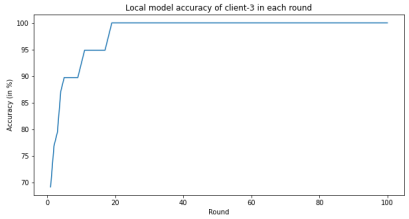
Figure 5.6: Loss of each client's local model using training data



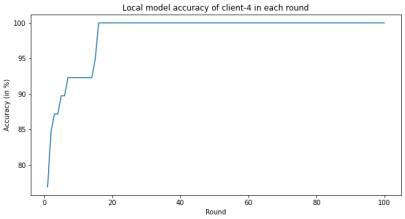
(a) Accuracy of Local Model for client-1



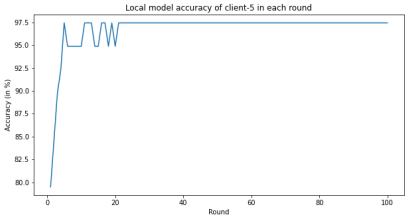
(b) Accuracy of Local Model for client-2



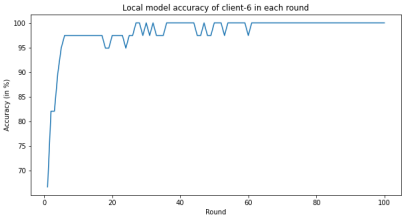
(c) Accuracy of Local Model for client-3



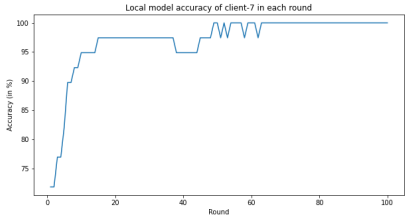
(d) Accuracy of Local Model for client-4



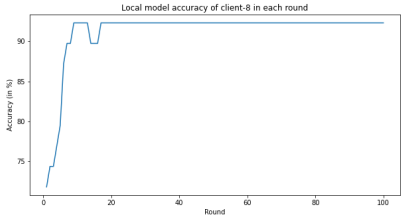
(e) Accuracy of Local Model for client-5



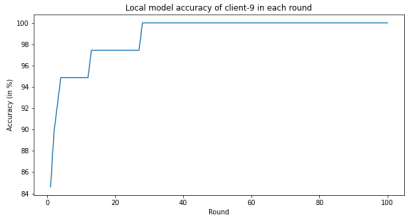
(f) Accuracy of Local Model for client-6



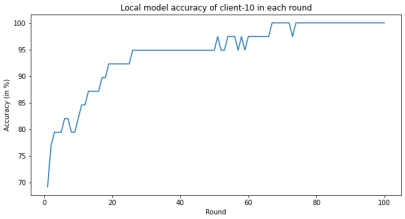
(g) Accuracy of Local Model for client-7



(h) Accuracy of Local Model for client-8



(i) Accuracy of Local Model for client-9



(j) Accuracy of Local Model for client-10

Figure 5.7: Accuracy of each client's local model using training data

5.2.6 Evaluating performance

The following graphs shows how the performance of our global model varied in terms of binary cross-entropy loss and accuracy over each round. We have also compared this model to a standard ML model which uses the same MLP model as the FL model and uses the same number of rounds/epochs.

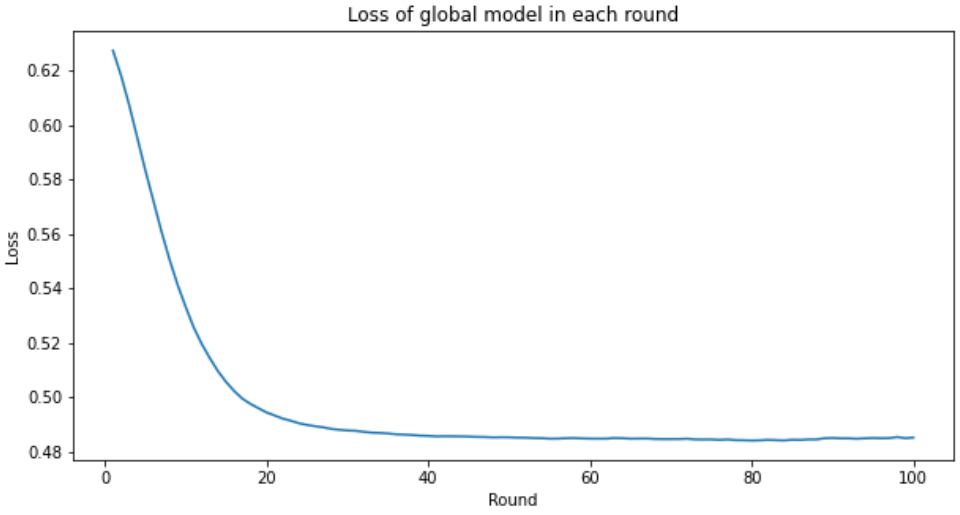


Figure 5.8: Loss of Global Model over 100 rounds

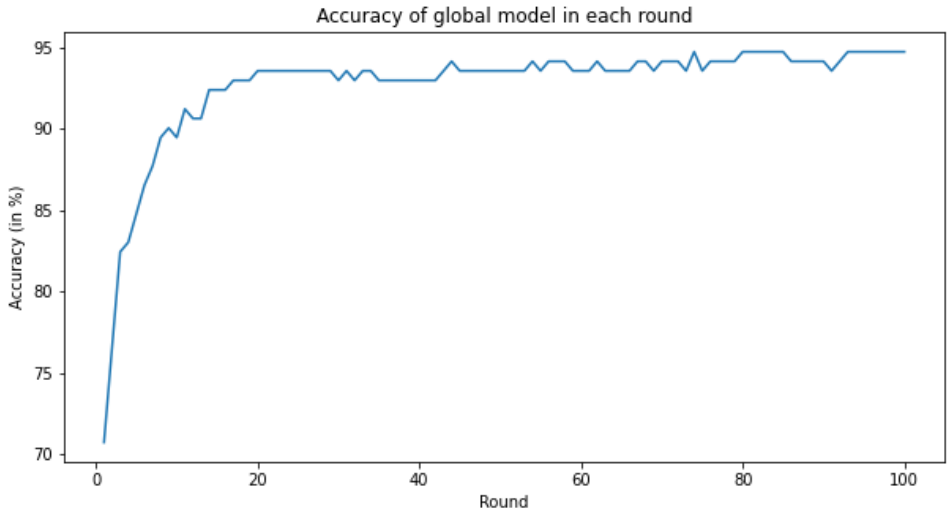


Figure 5.9: Accuracy of Global Model over 100 rounds

Model type	Loss	Accuracy (in %)
Federated model	0.485	94.737
Standard ML model	0.498	91.813

Table 5.1: Comparison between models

In our graph for the loss of the global model, we can see that the fall in binary cross-entropy loss reduces after each round, eventually converging at a loss of 0.485 at about the 90th round. Also, the rise in accuracy of the global model reduces after each round, eventually converging at an accuracy of 94.737% past the 90th round. Our global/federated model yields a better performance too in comparison to the standard ML model which is indicated by the loss and accuracy values found.
In addition, to the previous graphs, we have constructed confusion matrices of the global model and each local model at the end of training to see their performance on test data.

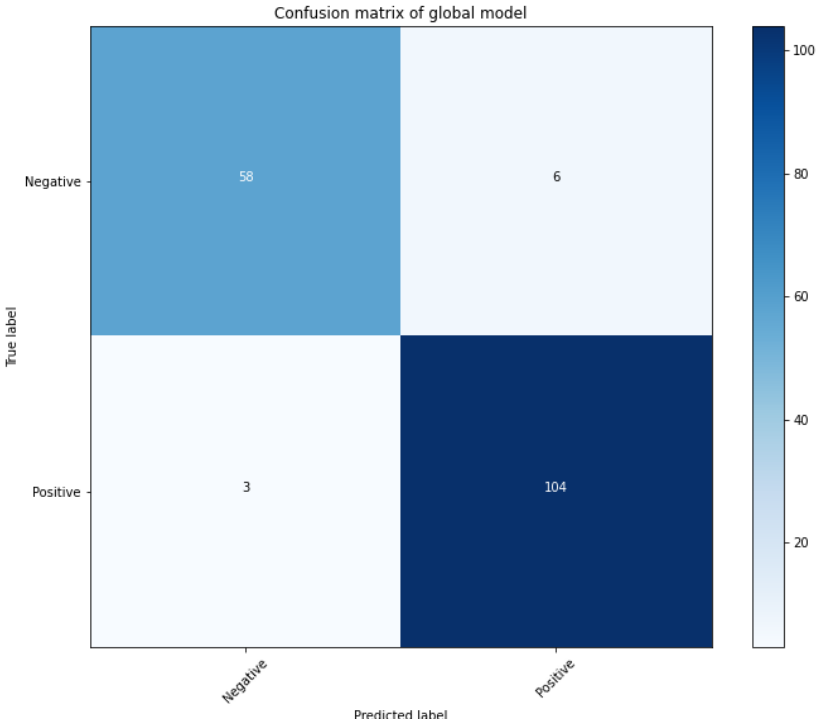
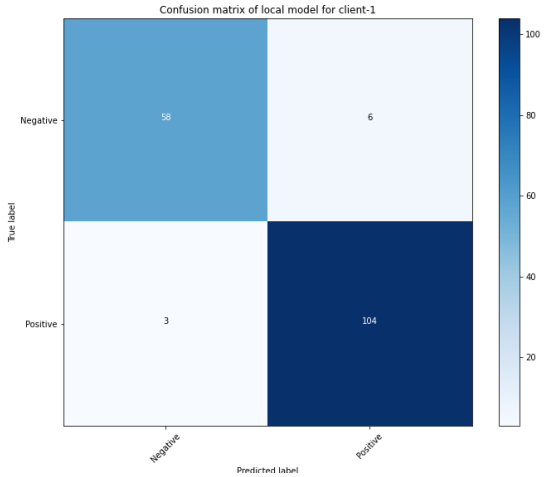
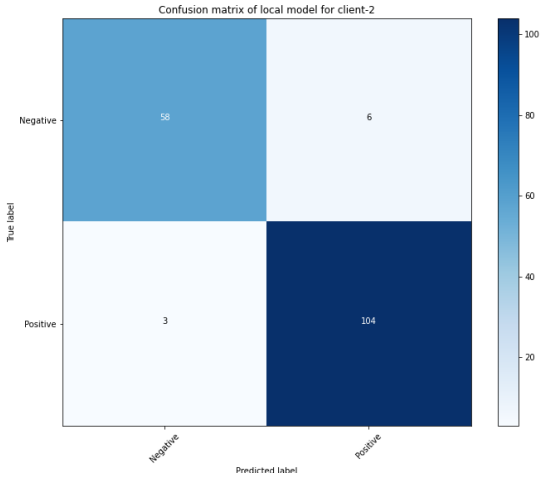


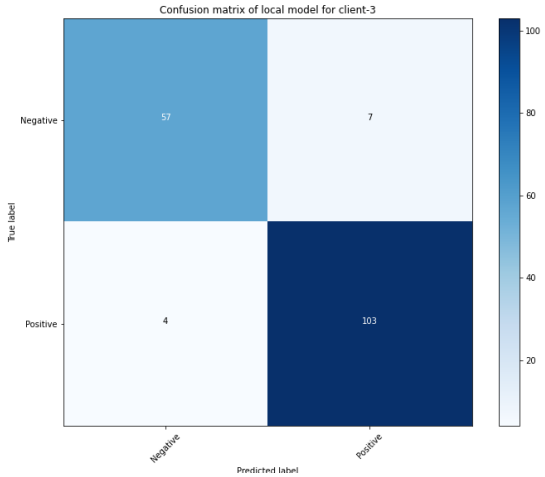
Figure 5.10: Confusion matrix of Global Model



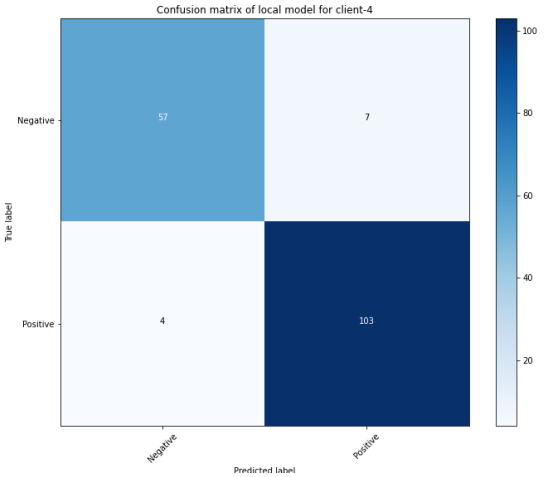
(a) Confusion matrix of Local Model for client-1



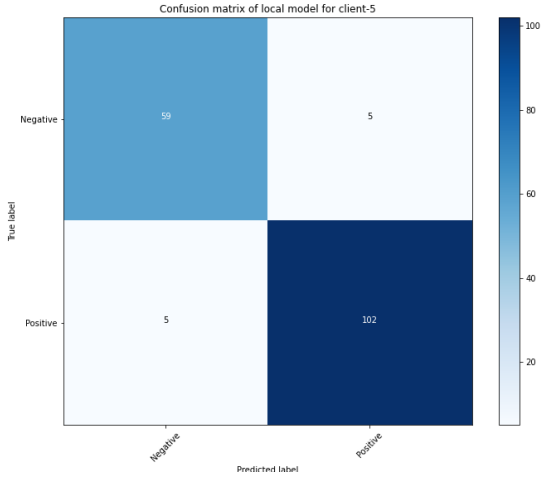
(b) Confusion matrix of Local Model for client-2



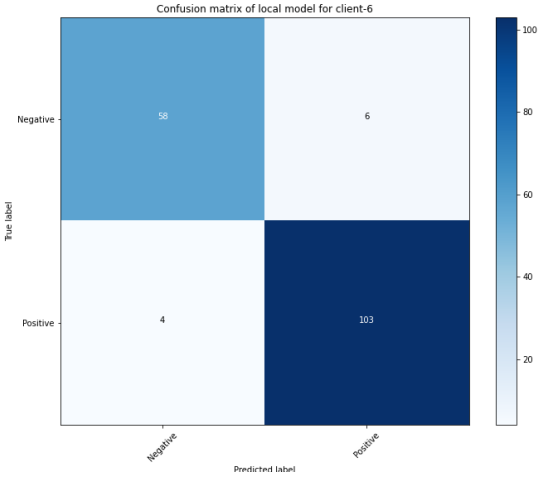
(c) Confusion matrix of Local Model for client-3



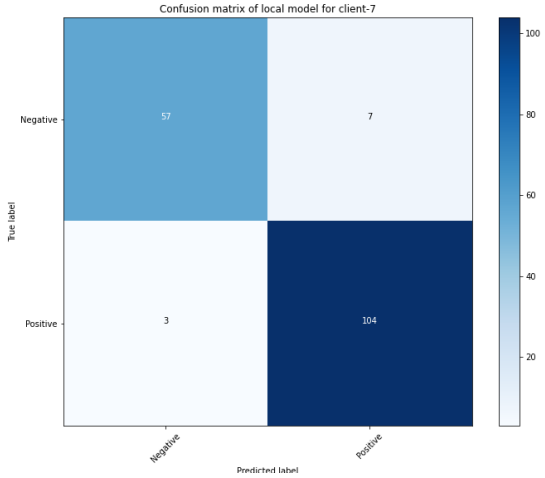
(d) Confusion matrix of Local Model for client-4



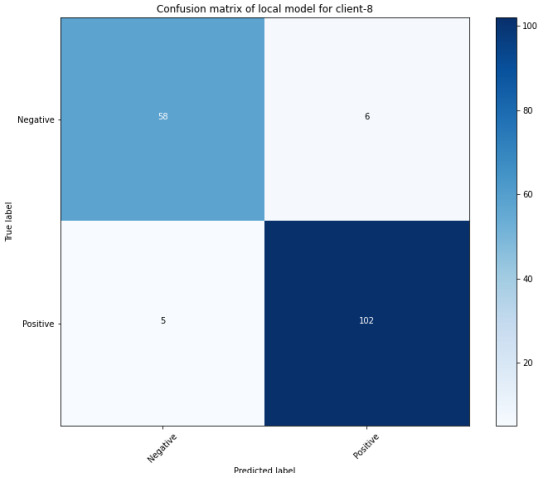
(e) Confusion matrix of Local Model for client-5



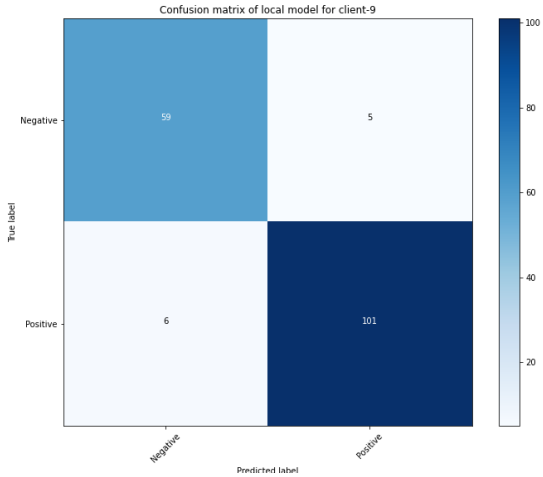
(f) Confusion matrix of Local Model for client-6



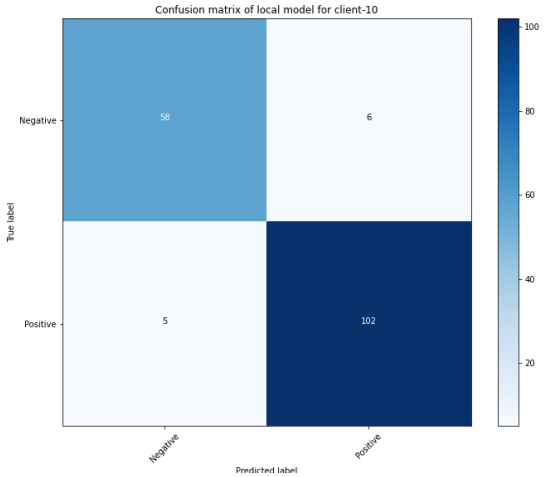
(g) Confusion matrix of Local Model for client-7



(h) Confusion matrix of Local Model for client-8



(i) Confusion matrix of Local Model for client-9



(j) Confusion matrix of Local Model for client-10

Figure 5.11: Confusion matrices of each client's local model

From our confusion matrices of the global model and each local model, we can see that local models of client-1 and client-2 have been able to achieve the same performance as the global model while the other local models have not. The confusion matrix for the local model of client-6 and client-7 has the closest performance among the local models which do not have the same performance as the global model and the performance of the local models for clients 8 and 10 are the same. Moreover, the local model for client-9 has the worst performance among all the local models.

Chapter 6

Conclusion

Maintaining the privacy and security of people's medical data is a matter of great concern in today's world. Information theft and cyber-attacks for medical data are rising exponentially year after year because medical data is very valuable, especially on the black market, because it can contain a patient's personally identifiable information such as social security numbers and health insurance credentials. Moreover, there are a large number of interconnected devices involved in healthcare, and healthcare organizations often need to share information across these devices. According to a Trustwave report, a medical record may fetch up to $250 per record on the black market whereas, in comparison, a credit card is valued at $5.40 [36]. This is because if fraud or theft is detected, the credit card or the account can get canceled but medical records may contain information that is very difficult or not possible to change, and thus the demand for such information is increasing. Therefore, people are looking for ways to upgrade their security and maintain the privacy and integrity of medical data because this might end up putting someone's life at risk. If you solely rely on cloud storage, you might not be able to access your data when outages or cloud server failure occurs. Through our proposed system, we hope to provide fast, secure, and reliable data storage and processing using edge computing and blockchain with federated learning so that data will not be stolen, lost, or otherwise tampered with due to data breaches or cloud server failure. It is going to be a challenge integrating edge computing and blockchain along with using federated learning in such a way so that they can both contribute to preserving people's medical data. Nonetheless, through our research, we hope to help healthcare organizations to keep their valuable medical data safe from being compromised or stolen by those with malicious intent and allow them to store, access, and process medical data using the Internet in a fast, efficient and reliable manner.

Bibliography

[1] LifeLock Official Site, "What Is Data Privacy and Why Is it Important?," Jan. 18, 2021. [Online]. Available: https://www.lifelock.com/learn-identity-theft-resources-what-is-data-privacy-and-why-is-it-important.html

[2] R. Geambasu, T. K. Kohno, A. A. Levy, and H. M. Levy, "Vanish: increasing data privacy with self-destructing data," in *Proceedings of the 18th Conference on USENIX Security Symposium*, pp. 299–316, 2009.

[3] A. Irei, "Understand why edge computing technology matters," SearchNetworking, Apr. 23, 2019. [Online]. Available: https://searchnetworking.techtarget.com/feature/Understand-why-edge-computing-technology-matters.

[4] K. Shaw, "What is edge computing and why it matters," Network World, Nov. 13, 2019. [Online]. Available: https://www.networkworld.com/article/3224893/what-is-edge-computing-and-how-it-s-changing-the-network.html

[5] M. Satyanarayanan, "The Emergence of Edge Computing," Computer, vol. 50, no. 1, pp. 30–39, 2017. DOI: 10.1109/MC.2017.9

[6] K. Gyarmathy, "The Benefits, Potential, and Future of Edge Computing," Data Centers and Colocation Services, blog, Apr. 29, 2021. [Online]. Available: https://www.vxchnge.com/blog/the-5-best-benefits-of-edge-computing

[7] J. M. Acken and N. K. Sehgal, "Security Considerations for Edge Computing," presented at the 9th International Conference on Computer Science, Engineering and Applications (CCSEA 2019), pp. 187–194, Jul. 13–14, 2019. DOI: 10.5121/csit.2019.90915

[8] A. Patel, "The Top Advantages Of Blockchain For Businesses," SmartData Collective, Jul. 08, 2021. [Online]. Available: https://www.smartdatacollective.com/top-advantages-blockchain-for-businesses/

[9] O. Rowe, "The pros and cons of blockchain adoption," FM Magazine, Dec. 01, 2019. [Online]. Available: https://www.fm-magazine.com/issues/2019/dec/blockchain-pros-and-cons.htm

[10] Y. Li, C. Chen, N. Liu, H. Huang, Z. Zheng, and Q. Yan, "A Blockchain-Based Decentralized Federated Learning Framework with Committee Consensus," *IEEE Network*, vol. 35, no. 1, pp. 234–241, 2021. DOI: 10.1109/mnet.011.2000263

[11] Binance Academy, "Blockchain Advantages and Disadvantages," Binance Academy, Oct. 21, 2020. [Online]. Available: https://academy.binance.com/en/articles/positives-and-negatives-of-blockchain

[12] L. Mearian, "Blockchain Techology What is blockchain? The complete guide," Computerworld, Jan. 30, 2019. [Online]. Available: https://www.shirebiz.net.au/nrsite/wp-content/uploads/2020/09/Block-Chain-Technology-Explained.pdf

[13] Y. Ren, Y. Leng, Y. Chang, and J. Wang, "Secure data storage based on blockchain and coding in edge computing," *Mathematical Biosciences and Engineering*, vol. 16, no. 4, pp. 1874–1892, Mar. 2019. DOI: 10.3934/mbe.2019091

[14] B. Brode, "What Is Blockchain and How Does It Work?," Hashed Out by The SSL Store™, blog, Mar. 11, 2021. [Online]. Available: https://www.thesslstore.com/blog/what-is-blockchain-how-does-blockchain-work/

[15] R. Sheldon, "6 steps to how blockchain storage works," SearchStorage, Jan. 10, 2019. [Online]. Available: https://searchstorage.techtarget.com/tip/6-steps-to-how-blockchain-storage-works

[16] M. B. Zahid, M. B. Rasheed, and N. B. Javaid, "Balancing Electricity Demand and Supply in Smart Grids using Blockchain," M.S. thesis, Dept. of CS, CUI, Islamabad, Pakistan, 2019

[17] B. W. Nyamtiga, J. C. S. Sicato, S. Rathore, Y. Sung, and J. H. Park, "Blockchain-Based Secure Storage Management with Edge Computing for IoT," *Electronics*, vol. 8, no. 8, p. 828, Jul. 2019. DOI: 10.3390/electronics8080828

[18] V. K. C. Ramesh, "Storing IOT Data Securely in a Private Ethereum Blockchain," M.S. thesis, Dept. of CS, UNLV, Nevada, USA, 2019

[19] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, vol. 151, pp. 1–32, 2014.

[20] Z. Kartit, A. Azougaghe, H. K. Idrissi, M. E. Marraki, M. Hedabou, M. Belkasmi, and A. Kartit, "Applying Encryption Algorithm for Data Security in Cloud Storage," *Lecture Notes in Electrical Engineering Advances in Ubiquitous Networking*, vol. 366, pp. 141–154, 2016. DOI: 10.1007/978-981-287-990-5_12

[21] K. Ziechmann, S. Richards, C. Jones, P. Wackerow, D. Awad, W. Entriken, and R. Cordell, "Introduction to smart contracts," ethereum.org, Mar. 30, 2021. [Online]. Available: https://ethereum.org/en/developers/docs/smart-contracts/

[22] J. Parms, "Symmetric vs. Asymmetric Encryption - What are differences?," SSL2BUY Wiki - Get Solution for SSL Certificate Queries, Nov. 30, 2020. [Online]. Available: https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences

[23] U. Jayasinghe, G. M. Lee, Á. Macdermott, and W. S. Rhee, "TrustChain: A Privacy Preserving Blockchain with Edge Computing," *Wireless Communications and Mobile Computing*, vol. 2019, pp. 1–17, Jul. 2019. DOI: 10.1155/2019/2014697

[24] Z. Xu, W. Liu, J. Huang, C. Yang, J. Lu, and H. Tan, "Artificial Intelligence for Securing IoT Services in Edge Computing: A Survey," *Security and Communication Networks*, vol. 2020, pp. 1–13, Sep. 2020. DOI: 10.1155/2020/8872586

[25] K. Wright, M. Martinez, U. Chadha and B. Krishnamachari, "SmartEdge: A Smart Contract for Edge Computing," *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1685-1690. DOI: 10.1109/Cybermatics_2018.2018.00281

[26] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When Mobile Blockchain Meets Edge Computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, 2018. DOI: 10.1109/mcom.2018.1701095

[27] C. Luo, L. Xu, D. Li, and W. Wu, "Edge Computing Integrated with Blockchain Technologies," *Complexity and Approximation Lecture Notes in Computer Science*, pp. 268–288, 2020. DOI: 10.1007/978-3-030-41672-0_17

[28] M. A. Rahman, M. S. Hossain, G. Loukas, E. Hassanain, S. S. Rahman, M. F. Alhamid, and M. Guizani, "Blockchain-Based Mobile Edge Computing Framework for Secure Therapy Applications," *IEEE Access*, vol. 6, pp. 72469–72478, 2018. DOI: 10.1109/access.2018.2881246

[29] F. Y. Rashid, "Why hackers want your health care data most of all," InfoWorld, Sep. 14, 2015. [Online]. Available: https://www.infoworld.com/article/2983634/why-hackers-want-your-health-care-data-breaches-most-of-all.html

[30] Z. Li, V. Sharma, and S. P. Mohanty, "Preserving Data Privacy via Federated Learning: Challenges and Solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 3, pp. 8–16, May 2020. DOI: 10.1109/mce.2019.2959108

[31] W. H. Wolberg, W. N. Street, and O. L. Mangasaria, Breast Cancer Wisconsin (Diagnostic) Data Set, 2016. [Online]. Available: https://www.kaggle.com/data/46091

[32] S. S. Haykin, "Multilayer Perceptrons," in *Neural Networks and Learning Machines*, 3rd ed., Upper Saddle River, New Jersey: Prentice Hall, 2009, pp. 122–229.

[33] D. Gupta, "Activation Functions: Fundamentals Of Deep Learning," Analytics Vidhya, blog, Jan. 30, 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

[34] S. Saxena, "Binary Cross Entropy/Log Loss for Binary Classification," Analytics Vidhya, blog, Mar. 03, 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/

[35] J. Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," Machine Learning Mastery, Jul. 03, 2017. [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[36] E. Neveux, "Healthcare Data Breaches & the Value of Healthcare Data," SecureLink, blog, Feb. 05, 2020. [Online]. Available: https://www.securelink.com/blog/healthcare-data-new-prize-hackers/