A THESIS

ON

BLOCKCHAIN TECHNOLOGY FOR CLOUD STORAGE

BY

Ms. Pratima Sharma

(2K18/PHD/CO/01)

UNDER THE SUPERVISION OF

Prof. Rajni Jindal

Professor

Department of Computer Science and Engineering

Delhi Technological University, Delhi

&

Dr. Malaya Dutta Borah

Assistant Professor

Department of Computer Science and Engineering

National Institute of Technology Silchar, Assam

Submitted in partial fulfilment of the requirements of the

DOCTOR OF PHILOSOPHY

IN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY, DELHI

INDIA

2022

DECLARATION

I, Pratima Sharma, a full-time scholar (Roll No: 2K18/PHD/CO/01), hereby declare that the thesis entitled "**Blockchain Technology for Cloud Storage**" which is being submitted for the award of the degree of Doctor of Philosophy in Computer Engineering, is a record of bonafide research work carried out by me in the Department of Computer Science & Engineering, Delhi Technological University. I further declare that the work presented in the thesis has not been submitted to any university or Institution for the award of any diploma or degree.

Ms. Pratima Sharma
Roll No: 2K18/PHD/CO/01
Department of Computer Science and Engineering
Delhi Technological University, Delhi

CERTIFICATE

This is to certify that the work embodied in the thesis entitled **"Blockchain Technology for Cloud Storage"** submitted by **Ms. Pratima Sharma**, Roll No: 2K18/PHD/CO/01 as a full-time scholar in the Department of Computer Science & Engineering, Delhi Technological University, is an authentic work carried out by her under my guidance. This work is based on the original research and has not been submitted in full or in part for any other diploma or degree of any university to the best of my knowledge and belief.

Supervisor

Prof. Rajni Jindal
Professor
CSE Department
Delhi Technological University, Delhi 110042

Joint-Supervisor

Dr. Malaya Dutta Borah
Assistant Professor
CSE Department
NIT Silchar, Assam-788010

ACKNOWLEDGEMENT

I owe a debt of my gratitude to my supervisor, Prof. Rajni Jindal, Professor, Department of Computer Science and Engineering, Delhi Technological University, and joint-supervisor, Dr. Malaya Dutta Borah, Assistant Professor, NIT Silchar, Assam, for their valuable guidance, motivation, constant support and encouragement, which has helped me in formulating an urge for this research work. Their exemplary hard work and insightful guidance have always helped me overcome difficulties.

I would like to take this opportunity to thank all the faculty members of the Computer Science & Engineering Department, Delhi Technological University, for their encouragement and support.

I am also thankful to all the staff members and research fellows for their untiring support.

I feel fortunate to have friends like Mrs. Amrita Sisodia and Mrs. Anshu Khurana, whose unbounded support kept my morale up and gave me positive reviews.

I would like to extend deep gratitude to my two sweet brothers, Mr. Anuj Sharma, who always wanted me to be a doctor, and Mr. Anshu Sharma for always being a pillar of strength and support to manage my personal and professional life. I feel fortunate to have a great family who has supported me throughout the journey of my research work.

Last but not least, without whom I could not imagine being enrolled in a Ph.D., my father, Mr. Shiv Dutt Sharma, and my mother, Mrs. Beena Sharma, for their unwavering encouragement, undeterred faith in me, and being the biggest pillar of strength which supported me till the end. It would not have been possible to complete my research without their love, encouragement, and support.

Ms. Pratima Sharma
Roll No: 2K18/PHD/CO/01
Department of Computer Science and Engineering
Delhi Technological University
Delhi-110042

ABSTRACT

The demand for blockchain innovation and the significance of its application has inspired ever-progressing exploration in various scientific and practical areas. Even though it is still in the initial testing stage, the blockchain is being viewed as a progressive solution to address present-day technology concerns, such as decentralization, identity, trust, ownership of data, and information-driven choices. Blockchain technology plays a vital role in various applications such as supply chain, Internet of Things, multimedia, healthcare sector, and cloud computing. Nowadays, blockchain is one of the advanced technologies to ensure sensitive or confidential data security. Thus, blockchain technology tremendously benefits the cloud storage system due to its several features, namely decentralization, confidentiality, security, privacy, etc. However, the conventional cloud storage model suffers from various security threats and privacy issues such as centralized data storage, which compromises server security, and the need for trusted third parties, affecting user privacy. Also, the data contained in many data centers are not completely distributed in today's distributed cloud computing. As a result, data security, integrity, and availability are adversely affected. Therefore, this research work integrates blockchain technology with a cloud storage system to provide reliable and secure cloud storage services for enterprises or individual users. This research work presented the five significant contributions in the blockchain-based cloud storage domain.

First, we have conducted a literature review of blockchain technology for cloud storage systems to highlight the challenges of the existing work and identify the various applications areas of the blockchain-based cloud system.

Second, we proposed a novel blockchain-based decentralized architecture for the cloud storage system to deal with identified challenges and provide an effective solution to cloud users. The proposed architecture implements the Ciphertext Policy Attribute-based Encryption (CP-ABE) algorithm to ensure confidentiality, integrity, and availability features. It provides a complete distributed environment for storing key-related information, user access policy, and integrity checking details in the blockchain structure. The proposed architecture is developed using the Java programming language. The various experiments are

conducted using tools such as CloudSim and Jmeter to evaluate the performance and analyze the efficiency of the proposed architecture.

Third, the proposed blockchain network also deployed the Honeybee optimization algorithm on a cloud storage system to optimize the resources and minimize the transaction response and execution time.

Fourth, the proposed work is further enhanced to include attribute authority entities and ensure fine-grained access control with the user revocation process without any trusted authority. Various experiments analyze the performance and compare the proposed architecture with various existing works.

Lastly, the proposed blockchain-based cloud architecture is deployed in different domains such as healthcare and intrusion detection systems to analyze its performance. The proposed architecture is implemented using the Ethereum platform and InterPlanetary File System (IPFS) storage system to manage the data in the healthcare domain. Further, the blockchain architecture is developed for intrusion detection systems to identify the intruder node in the voting system using an adaptive behavior model.

The experimental results, analysis, and performance evaluation demonstrate that the proposed work provides a feasible and reliable cloud environment. Furthermore, the comparative analysis shows that the proposed scheme is better than the existing techniques. Thus, this research work successfully provides an effective, optimal, and secure cloud storage system using blockchain technology.

Contents

Declaration……... i
Certificate……. i
Acknowledgement…… ii
Abstract…… iii
Contents…… v
List of Abbreviations…… viii
List of Figures…….. ix
List of Tables…… xii
List of Algorithms…… xiii
CHAPTER 1: INTRODUCTION
1.1 Background…… 1
1.1.1 Cloud Storage System and Associated Challenges…….. 1
1.1.2 Blockchain Technology…… 3
1.1.3 Blockchain-based Cloud Storage System……. 5
1.2 Motivation…….. 6
1.3 Problem Statement……. 7
1.4 Contribution of the Thesis……. 7
1.5 Thesis Organization…… 8
CHAPTER 2: LITERATURE REVIEW
2.1 Introduction…… 12
2.2 Planning the Review……... 13
2.3 Conducting the Review…….. 13
2.4 Reporting the Review……. 14
2.4.1 Traditional Cloud Storage Techniques……. 14
2.4.2 Blockchain Techniques for Cloud Storage…….. 17
2.4.3 Identification of Research Gaps……... 20
2.5 Concluding the Review…….. 21
CHAPTER 3: PRELIMINARIES
3.1 Blockchain Technology…….. 22
3.2 Types of Blockchain……... 24
3.3 Consensus Mechanism……... 25
3.3.1 Proof of Work…….. 25
3.3.2 Proof of Stack…….. 26
3.3.3 Proof of Authority…… 26
3.4 Cryptographic Algorithms…….. 27
3.4.1 Hashing……. 27

3.4.2 Merkle Tree…….. 27
3.4.3 Ciphertext Policy Attribute-Based Encryption Algorithm……... 28
3.5 Smart Contract…… 28
CHAPTER 4: BLOCKCHAIN-BASED DECENTRALIZED CLOUD STORAGE SYSTEM
4.1 Introduction…… 30
4.2 Proposed Work……... 32
4.2.1 System Model……... 32
4.2.2 Smart Contract Functions……. 38
4.3 Example Scenario……... 48
4.4 Security Analysis…… 49
4.5 Result Analysis……... 50
4.5.1 Experimental Setup…….. 50
4.5.2 Storage Performance…… 51
4.5.3 System Performance……. 53
4.5.4 Comparative Analysis…….. 55
4.6 Conclusion……. 58
CHAPTER 5: ACCESS CONTROL AND USER REVOCATION PROCESS IN CLOUD STORAGE USING BLOCKCHAIN
5.1 Introduction…… 60
5.2 Proposed Work……... 62
5.2.1 System Model……... 62
5.2.2 Smart Contract Functions……. 66
5.3 Security Analysis…… 75
5.4 Result Analysis……... 76
5.4.1 Experimental Setup…….. 76
5.4.2 System Performance..…..…… 77
5.4.3 Comparative Analysis…….. 79
5.5 Conclusion…….. 81
CHAPTER 6: APPLICATIONS OF PROPOSED WORK
6.1 Introduction…… 82
6.2 Blockchain-based Healthcare System…… 83
6.2.1 Proposed Architecture…….. 84
6.2.2 Smart Contract Functions……. 86
6.2.3 Conceptual Scenario……. 89
6.2.4 Result Analysis……..……... 93
6.2.4.1 Experimental Setup…… 93
6.2.4.2 Performance Evaluation……. 95
6.2.4.3 Comparison Analysis……..……... 97

6.2.5 Summary of Healthify…….. 98
6.3 Intrusion Detection System Using Blockchain Technology…….. 98
6.3.1 Proposed Architecture……. 99
6.3.2 Results…….. 104
6.3.3 Summary of LVRS…….. 106
CHAPTER 7: CONCLUSION AND FUTURE WORK
7.1 Summary of Work Done in the Thesis……... 107
7.2 Contribution of the Research……. 108
7.3 Future Work……... 109
Publications from the Thesis…… 110
References…… 112

List of Abbreviations

CP-ABE	Ciphertext-Policy Attribute-based Encryption Algorithm
RQ	Research Question
KU-CSP	Key Update Cloud Service Provider
ZEUS	ZEro-knowledge dedUplication reSponse framework
PKG	Public Key Generator
CSP	Cloud Service Provider
TPA	Third Party Auditor
PHC	Paillier Homomorphic Cryptography
SSE	Searchable Symmetric Encryption
BCpay	BlockChain Payment
IDaaS	ID as a Service
BIDaaS	Blockchain-based ID as a Service
PoW	Proof of Work
PoS	Proof of Stack
PoA	Proof of Authority
SHA-256	Secure Hash Algorithm
AES	Advanced Encryption Standard
NDN	Named Data Network
EHR	Electronic Health Records
IPFS	InterPlanetary File System
Dapp	Distributed Application
LVRS	Layered Voting Rule System
PLR	Positive Layer Repository
NLR	Negative Layer Repository
NA	Network Admin

List of Figures

Figure 1.1: Working Principle of the Blockchain…….. 4

Figure 2.1: Overall Flow of the Literature Review……. 12

Figure 3.1: Blockchain Architecture…….. 23

Figure 3.2: Structure of the Blockchain……..……. 23

Figure 3.3: PoW Consensus Mechanism……. 26

Figure 3.4: Structure of the Merkle Tree……. 27

Figure 4.1: System Model of the Proposed Work…….. 35

Figure 4.2: Registration Flow of Data Owner and User…….. 39

Figure 4.3: Data Outsource Workflow……..…….. 42

Figure 4.4: Access Control Workflow…… 44

Figure 4.5: Integrity Checking Process…….. 46

Figure 4.6: Average Response Time…….. 51

Figure 4.7: Average Execution Time……..…… 52

Figure 4.8: Average Resource Utilization…….. 53

Figure 4.9: Throughput of the Proposed Work……..... 54

Figure 4.10: Computation Time of the Proposed Work…….. 54

Figure 4.11: Delay of the Proposed Work…….. 55

Figure 4.12: Transmission Time Comparison between Proposed Work Without

Optimization and Block Secure……..... 56

Figure 4.13: Transmission Time Comparison between Proposed Work with Optimization and Block Secure……..... 57

Figure 4.14: Run-Time Comparison of Proposed Architecture with the Secure Access Control…….…… 57

Figure 4.15: Encoding Time Comparison between Proposed Work and Deduplication with Blockchain……... 58

Figure 4.16: Decoding Time Comparison between Proposed Work and Deduplication with Blockchain ……..... 58

Figure 5.1: System Model of the Proposed Architecture…….……... 63

Figure 5.2: Tree of the Access Structure…….…… 70

Figure 5.3: Key Generation Time Comparison between Proposed Scheme and Cryptcloud…… 77

Figure 5.4: Encryption Time Comparison between Proposed Scheme and Existing Techniques……... 78

Figure 5.5: Decryption Time Comparison between Proposed Scheme and Existing Techniques……... 78

Figure 5.6: Re-Encryption Time Comparison between Proposed Scheme and NDN Technique…… 79

Figure 6.1: Architecture of Blockchain-based Distributed Application for Healthcare…… 85

Figure 6.2: Smart Contract Functions……. 86

Figure 6.3 (a): The Flow Diagram of Patient……..... 90

(b): The Flow Diagram of Doctor……... 91

(c): The Flow Diagram of Diagnostic Centre……. 92

(d): The Flow Diagram of Healthcare Analyser……. 93

Figure 6.4: Computation Time Required for Completion of Transactions…….. 96

Figure 6.5: The Hop Behavior……..……. 99

Figure 6.6: Proposed Layer Voting Rule Architecture…… 101

Figure 6.7 (a): Positive Vote ……... 101

(b): Penalty…….. 101

(c): Decision Making of LVRS……... 102

Figure 6.8: Throughput of the Proposed Architecture……. 105

Figure 6.9: Power Consumption of the Proposed Architecture……..... 105

List of Tables

Table 4.1: Notations and Descriptions…… 33

Table 5.1: Notation Table…… 65

Table 5.2: Comparison of Related Work with Proposed Scheme……... 80

Table 5.3: Efficiency Comparison of Related Work with Proposed Scheme……. 81

Table 6.1: Development Environment for the Proposed Application……..... 94

Table 6.2: Development Environment for the Blockchain Smart Contract……..... 94

Table 6.3: Time Required for Uploading Different Sized Files..... 95

Table 6.4: Deployment Cost of Contracts……..... 96

Table 6.5: Gas Used in Calling/Sending Functions of Smart Contract……... 96

Table 6.6: Comparative Analysis of the Proposed Application with the Existing Studies……..... 97

Table 6.7: Deployment Parameters……..... 103

List of Algorithms

Algorithm 4.1: Key Generation……..……... 40

Algorithm 4.2: Data Outsourcing……..... 42

Algorithm 4.3: Access Control…….…… 44

Algorithm 4.4: Integrity Checking…….……... 46

Algorithm 4.5: Optimization……..……... 47

Algorithm 5.1: Key Generation Algorithm for Data Owner……..…….... 67

Algorithm 5.2: Key Generation Algorithm for Attribute Authority..... 67

Algorithm 5.3: Key Generation Algorithm for User…… 69

Algorithm 5.4: Encryption…….…… 71

Algorithm 5.5: Re-Encryption…….……..... 72

Algorithm 6.1: User Registration…...……... 87

Algorithm 6.2: Upload File…… 87

Algorithm 6.3: Share File……..... 88

Algorithm 6.4: Integrity Checking…… 89

Algorithm 6.5: Deployment of Node……. 103

Algorithm 6.6: Identification of Power Consumption……... 104

CHAPTER 1

INTRODUCTION

This chapter begins with background details of the cloud storage system, highlights the associated challenges, and introduces blockchain technology. It also discusses the concept and requirement of the blockchain-based cloud storage system, research motivation, problem statement, and contributions. The chapter concludes with the documentation of the organization of the thesis.

1.1 Background

This section covers details of the cloud storage system, identifies the associated challenges, explains the blockchain technology concept, and presents the need to integrate the blockchain technology with the cloud storage system.

1.1.1 Cloud Storage System and Associated Challenges

Nowadays, data are the main asset. Many electronic devices such as mobile phones, computers, cameras, and laptops generate a massive amount of data each day which needs more storage space and resources. According to the Forbes report [1], around 2.6 quintillion bytes of data was generated every day and out of which 90 percent of data was generated in just the last two years. Therefore, the cloud storage system is required to manage such a massive increase in the data and fulfil the storage requirements. The cloud storage system has distributed data centers or servers that utilize virtualization technology to work together and provide storage resources. Recently, the cloud storage system has attained massive attention from business organizations and individual users because it is convenient to use. Many users and organizations outsource their data on the cloud storage system to alleviate the burden of storage and maintenance in the local storage [2]. The cloud storage system provides on-demand and flexible storage services to an individual or business using a third party over the

internet. The service providers are responsible for maintaining cloud servers and providing services to the users to store and process their data over the internet.

The cloud storage system allows users to access the data anywhere and anytime, thereby supporting an on-demand and pay-per-use model. The user can rent and pay the storage and computation services based on the requirement with the help of the cloud storage model. The cloud provides many benefits and functionality to cloud users, such as low-cost storage, flexibility, automatic update, disaster tolerance, etc., [3-5]. However, it is important to protect user privacy [6] and ensure data protection [7] as data may leak while users store their data in the cloud. Also, users lose control over the data after outsourcing to the cloud, and cloud data may not be safe and vulnerable to various attacks [8-11]. The current cloud storage architectures have other flaws, such as centralized data storage, which compromises server security, and the need for trusted third parties affects user privacy.

Furthermore, the existing distributed cloud storage system stores the data in multiple data centers or servers in a distributed manner, but they are not completely distributed. Several data centers store the data at high density. Thus, a large amount of data will be exposed if one of the servers or data centers is compromised [12]. Public media worldwide have repeatedly documented unauthorized access concerns related to cloud storage, such as user's private files being leaked on iCloud [13]. Unfortunately, there are no viable solutions for the security of cloud systems yet.

Some of the challenges of the existing cloud storage system are identified as follows:

1. Security: The current cloud storage system provides services that are centrally managed and processed by the centralized authority. However, this configuration suffers from a single point of failure issues, which affects the availability security feature that provides on-demand access to cloud users.

2. Privacy: The cloud storage architecture provides convenient services to the users, but it also creates serious data privacy issues, given the large volume of user data that is collected, stored, and managed on the cloud networks. Thus, cloud users trust the cloud service providers to manage their data, even when they have limited knowledge of where their data is transmitted and who is accessing it [14]. Similarly, the distributed cloud storage architectures do not completely distribute the data among cloud servers, thus creating privacy issues if one of the servers is compromised, user data may be leaked.

3. Integrity: Integrity risks may arise due to storing and maintaining user data on the cloud. Outsourced data is at risk of being updated or deleted by third parties without user authorization. Adversaries can also tamper with cloud data resources for financial or political gain, jeopardizing data integrity. Due to these factors, many solutions based on public verification schemes using a third-party auditor have been proposed [15]. Still, they may raise several issues, such as irresponsible verification that results in skewed data integrity due to malicious auditors. Therefore, there is a requirement to develop new solutions to resolve the data integrity challenges necessary for the cloud storage system.

4. Data Breach: The data breach may be a human error, security vulnerability, or intended error in which protected data is leaked to the public by malicious users. Depending upon the sensitivity of the leaked data, it creates huge damage to the users. The data breach issue arises due to the lack of proper encryption techniques used by the cloud. It results in the theft of users' sensitive information. Therefore, the utilization of security measures like cryptographic algorithms could resolve the data breach issues.

5. Loss of Data Control: Cloud users hand over their sensitive or confidential data to the third party, considering that the data centers are secured and establish a trust relationship with the cloud service providers. But, the location of the user data is not shared with the cloud users. Thus, the user has no control over the data that is where it is stored or processed on the cloud platform. Better transparency between the cloud users and service providers should be required to store or process the data.

1.1.2 Blockchain Technology

Blockchain is mainly considered the core technology of Bitcoin cryptocurrency, developed by an unknown person Nakamoto in 2008 [16]. In essence, blockchain technology is a peer-to-peer network that serves as a public trusted and shared ledger. This innovative technology has recently emerged as a popular technique for academicians and researchers as it has the potential to develop blockchain-based applications beyond Bitcoin cryptocurrency. The key focus of blockchain technology is decentralization which indicates that the blockchain is shared throughout the network nodes. Each network node has the authority to check the operation of other nodes in the network to generate, verify, and validate the new transactions of the blockchain network. The blockchain decentralization architecture provides reliable and secure operations with tamper resistance and no single point of failure features. Generally, the blockchain is classified into two categories, public and private blockchain networks. The

public blockchain is accessible to everyone, which means anyone can join and generate transactions and participate in the consensus mechanism (e.g., Bitcoin network). However, the private blockchain is a permissioned network in which all participants have to take permission from the authority to participate or create transactions in the blockchain network. The general overview of blockchain working is presented in Fig. 1.1.

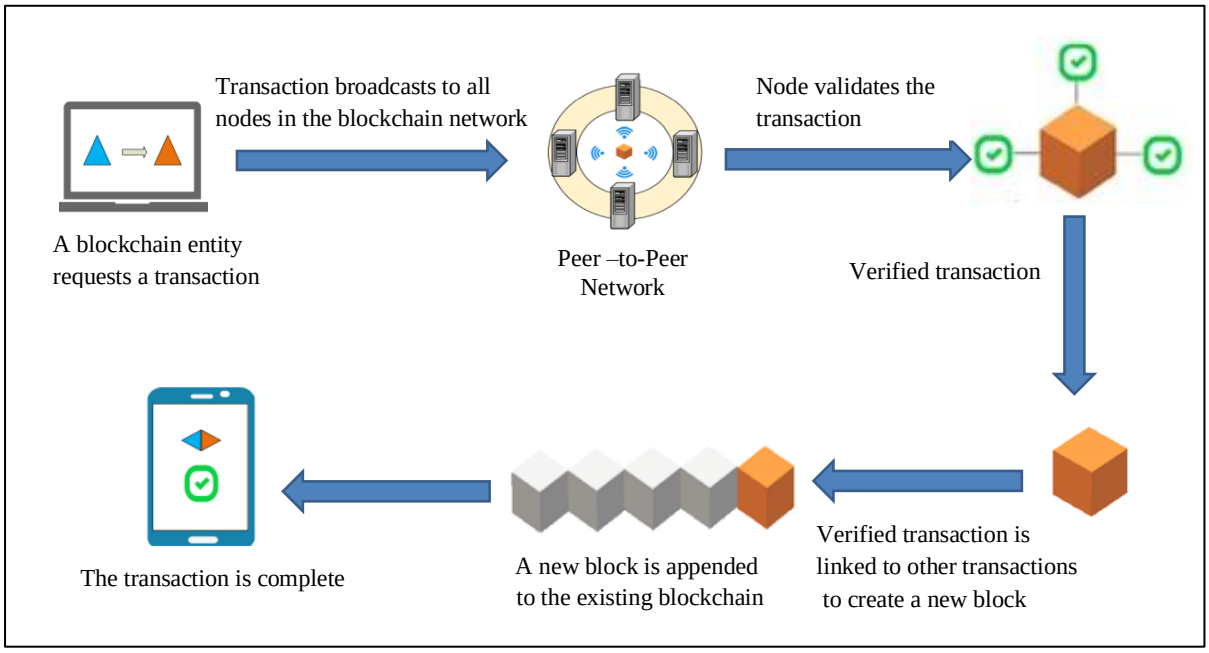


Figure 1.1: Working Principle of the Blockchain

The blockchain network core components include transaction, block, ledger, consensus, and smart contracts. In a blockchain network, a transaction stores the user's information in a block to form a ledger, and each block is linked to the immediately previous block using hash values. In this manner, all blocks are connected to form a chain. Each block can be traced back to the preceding block, and no alteration is allowed in the block data [17]. The blockchain ledger is a decentralized database that is shared and distributed to the blockchain network participants. Moreover, the consensus algorithm ensures the security of the blockchain network by allowing multiple untrustworthy nodes to agree on a single data block. Finally, smart contracts are programs deployed on the blockchain network to establish the contractual rules among the network participants [18]. As a result, blockchain technology provides high-security features for its application scenario, such as a cloud storage system. The most significant aspect of blockchain technology is decentralization. It indicates that the central authority does not manage blockchain transactions. This remarkable property has several intriguing advantages, including reducing the chance of a single point of failure

problem due to a breakdown of centralized authority, minimizing operations expenses, and increasing reliability. Also, blockchain allows transaction data to remain unchangeable or immutable over time. The new block's hash is always generated from the previous block hash value that makes the chain of blocks unchangeable. Once the block is validated and added to the blockchain network, it is impossible to edit, change or erase its data. Another key aspect is transparency, which means that all transaction data of the blockchain network is visible to all network participants. For public verifiability, the blockchain distributes the same copy of the transaction records in the form of a ledger across the blockchain network to allow full access, verification, and transaction tracking activities to all users with equal rights.

1.1.3 Blockchain-Based Cloud Storage System

With the increasing volume of data and the security issues involved in the cloud storage system, a new model for a storage system using blockchain technology is required to overcome these security issues. The combination of blockchain and cloud storage systems leads to a novel paradigm called a Blockchain-based Cloud Storage System. The convergence of these advancements provides great benefits to both domains, such as ensuring efficiency, decentralization, transparency, security, and providing better cloud services. Blockchain technology provides a fully decentralized storage architecture without involving any centralized authority. Blockchain enables the new cloud storage functions that are resistant to any data modifications and provides many potential benefits to cloud storage systems, as explained below:

1. Decentralization: Blockchain-based decentralized structure provides a promising solution to solve the single point of failure issue by eliminating the need for trusted third parties. Furthermore, the blockchain peer-to-peer architecture provides equal validation rights to the network participants to validate and verify the data correctness and ensure immutability features.

2. Security: Blockchain technology improves the security of the cloud storage system through the inherent properties of the blockchain network, such as confidentiality and availability. The blockchain network records all transactions that are cryptographically secured and stored in the form of hash values. Furthermore, blockchain transactions are signed by the network participants so that the user interaction with the cloud storage system remains confidential. The blockchain network also supports the off-chain storage solution to support the availability feature [19].

3. Privacy: Blockchain provides high-level protection to the cloud storage system with immutability, integrity, and transparency features. The blockchain network stores the information in a linked chain in which each block connects to the previous block. Thus, it is impossible to modify or alter the blockchain content as blockchain is preserved and controlled by the secure and immutable consensus mechanism.

4. System Complexity: The integration of blockchain with a cloud storage system significantly reduces the implementation complexity. The blockchain algorithms can be executed using the cloud infrastructure to reduce the cost required for blockchain resources. The combination of blockchain and cloud brings various solutions to deploy blockchain-based cloud storage systems at a large scale with low cost and simple implementations.

5. Feasibility: Many large organizations currently deploy blockchain as a service project to measure the feasibility of integrating blockchain with cloud systems. Many companies such as Amazon, IBM, Microsoft, and Oracle have launched the blockchain as a service platform on cloud computing [20]. Also, the integration of blockchain and cloud provides great benefits in various domains such as the Internet of Things, healthcare, banking, supply chain, and multimedia.

1.2 Motivation

Data storage over the cloud is a recent practice, and they offer free and large storage capability. Even though the cloud provides us with many advantages, security is the primary concern for the users, as highly confidential data will also be stored in the cloud. The integrity of data remains an essential requirement in a sharing-based cloud storage system. Moreover, the access control in the existing cloud model requires one or more completely trusted attribute or central authorities to maintain the access policy. If the central authority is broken down, the entire system is disturbed. Also, the user revocation where proxy servers and re-signature generation is required remains a major task. Therefore, achieving security and privacy features for encrypted data poses a significant challenge. As a result, decentralized systems are critical in integrity checking, access control, and user revocation to eliminate trusted center authority's possible threat. Therefore, to improve the performance of existing applications, there is a need to integrate cloud storage and blockchain technology to propose a blockchain-based distributed cloud storage architecture that can provide secure and reliable cloud storage services for end-users.

1.3 Problem Statement

Lots of efforts are going into enhancing the security of the cloud storage system. The main objective of this thesis is to develop blockchain-based decentralized cloud storage architecture to provide a privacy-preserving environment with all security features. Following are the problems addressed in this thesis:

1. Conventional cloud storage systems store data on central servers and involve trusted third parties that compromise cloud data security. Due to the rapid cloud storage system requirement, architecture needs to deal with various security issues.

2. The cloud storage system involves third-party auditors to verify the integrity of the stored cloud data. But this raises many issues, such as irresponsible verification that leads to manipulating the data integrity results or invalidating verification due to malicious auditors.

3. The cloud users have no control over where or how their data is stored on the cloud platform. The cloud storage system involves a third party to process the user's sensitive or confidential data, thus breaching data privacy.

4. The traditional encryption, access control, and integrity methods store information in one or more completely trusted authorities or centralized systems, which affects the system's security.

5. The centralized network infrastructure involves higher power consumption and communication latency which hinders the large-scale developments of the cloud storage systems in practical scenarios. Therefore, an optimization algorithm should be required to optimize resource utilization and minimize data processing time to increase network performance.

1.4 Contribution of the Thesis

The main objective of the thesis is to develop a blockchain-based secure architecture to store data on the cloud with authentication, integrity check, access control, and revocation process to deal with the disadvantages and challenges mentioned above. The proposed architecture deploys various cryptographic algorithms to provide a key generation mechanism, achieve confidentiality, perform a data access control mechanism, and ensure cloud data integrity. The architecture also utilizes a Honeybee optimization algorithm to optimize the resource

utilization on the cloud storage system and minimize transaction processing time in terms of execution and response time. By using blockchain technology, we achieve decentralization with security without the need for a trusted central authority.

The thesis contributions are as follows:

1. The literature review is conducted to explore and highlight the techniques of blockchain technology utilized to secure cloud storage systems. It also identifies the various application areas of integrating blockchain with the cloud storage system.

2. The distributed blockchain-based cloud storage system is proposed to provide security features without involving any trusted authority. The proposed architecture provides a complete distributed approach to generate key-related information without involving any centralized authority using the blockchain structure.

3. The proposed work implements the CP-ABE algorithm to ensure confidentiality, integrity, and availability features. It provides confidentiality by utilizing the encryption technique to encode the user data. The integrity feature is achieved using the blockchain-based Merkle root concept.

4. The blockchain-based cloud storage architecture implements the Honeybee optimization algorithm on a cloud storage system to optimize the resources and minimize the transaction response and execution time.

5. The proposed scheme is further enhanced to include the fine-grained access control with the immediate user revocation process at the system level rather than periodically.

6. The performance and security analysis based on experimental results are evaluated by deploying the proposed architecture in various domains such as healthcare, intrusion detection system, multimedia, etc., to show the system's capability.

1.5 Thesis Organization

The organization of the thesis is presented in this section which comprises seven chapters as listed below:

Chapter 1: Introduction

This chapter introduces the research work with the problem statement. We also present a brief outline of the thesis in this chapter.

Chapter 2: Related Work

This chapter presents a research methodology by defining and explaining the research problem in detail with the help of formulated research questions. We have also presented a brief description of existing blockchain and cloud storage work. This chapter is concluded by identifying research gaps based on existing studies.

The following paper has been published from this work:

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "Blockchain Technology for Cloud Storage: A Systematic Literature Review," *ACM Computing Surveys*, vol. 53, no. 4, 2020. **(SCI, IF: 10.2)**

Chapter 3: Preliminaries

This chapter explains the essential components of the proposed blockchain-based cloud storage architecture. We have presented the technical details related to the proposed blockchain structure, such as block header, block hash, nonce, transaction, mining, consensus, etc. This chapter also describes cryptographic algorithms and smart contracts used in the proposed work.

The following paper has been published/communicated from this work:

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "A Review of Smart Contract-based Platforms, Applications, and Challenges," *Cluster Computing*, Springer, vol. 2021. **(SCIE, IF: 1.8)**
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "A Comparative Analysis of Consensus Algorithms for Decentralized Storage Systems," *IT Professional*, IEEE. (Communicated)

Chapter 4: Proposed Blockchain-based Cloud Storage Architecture

This chapter covers a detailed description of the proposed distributed blockchain-based cloud storage architecture. It presents the complete discussion of security features achieved by the proposed work.

The following paper has been published from this work:

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "Blockchain-based Integrity Protection System for Cloud Storage," In *Proceedings of the 4th Technology Innovation Management and Engineering Science International Conference* (TIMES-iCON), Bangkok, Thailand, 11th-13th December, 2019, pp. 1-5.
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "Blockchain-based Decentralized Architecture for Cloud Storage System," *Journal of Information Security and Applications,* vol. 62, pp. 2214-2126, 2021. **(SCIE, IF: 3.8)**

Chapter 5: Access Control and Revocation Process in Cloud Storage using Blockchain

This chapter presents the access control and revocation process achieved in the proposed architecture. It gives the details of formulating an access policy for the user using an attribute list and provides the concept of re-encryption for the user revocation process.

The following paper has been published from this work:

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "Blockchain-based Cloud Storage System with CP-ABE based Access Control and Revocation Process," *Journal of Supercomputing*, Springer, 2021. **(SCI, IF: 2.4)**

Chapter 6: Applications of Proposed Work

This chapter evaluates the performance of the proposed architecture by deploying it in healthcare and intrusion detection systems.

The following paper has been published from this work

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "A Review of Blockchain-based Applications and Challenges," *Wireless Personal Communications: An International Journal*, Springer, 2021. **(SCIE, IF: 1.6)**

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, “Blockchain-based Secure Healthcare Application," In: edited book entitled Blockchain in Digital Healthcare, Taylor & Francis Group, pp. 35-54, 2021.
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, “A Preventive Intrusion Detection Architecture using Adaptive Blockchain Method," In *Proceedings of the International Conference for Big Data, Machine Learning and Applications* (BigDML 2019), NIT Silchar, Assam, 16th-19th December, 2019, pp. 25-35.
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, “Healthify: A Blockchain-based Distributed Application for Healthcare," In: edited book entitled Application of Blockchain Technology in Healthcare, Springer Book Series "Studies in Big Data”, pp. 171-198, 2021.

Chapter 7: Conclusion

The final chapter presents the conclusion and future scope of the research work. This chapter deliberates upon the importance of the proposed blockchain architecture for providing privacy-preserving and security to cloud storage systems.

List of Publications: This section lists published/accepted/communicated papers relating to this research work in International/National Journals/Conferences of repute.

References: This section is the list of references cited in this research work.

CHAPTER 2

LITERATURE REVIEW

This chapter presents a literature review on traditional cloud storage techniques that significantly improve security features and covers research work based on integrating blockchain and cloud storage systems. It formulates research questions, identifies research gaps, and provides future directions based on the study.

2.1 Introduction

This literature review aims to comprehensively study existing and on-going research work, explore the research gaps, and propose a solution. As shown in Fig. 2.1, the literature review follows the four steps. First, it plans the review process and identifies the categories for the review. Next, the research questions are formulated to review existing blockchain and cloud storage techniques. Then, the category-wise review report is presented, and research gaps are identified based on the study. Finally, the review is concluded with future directions.

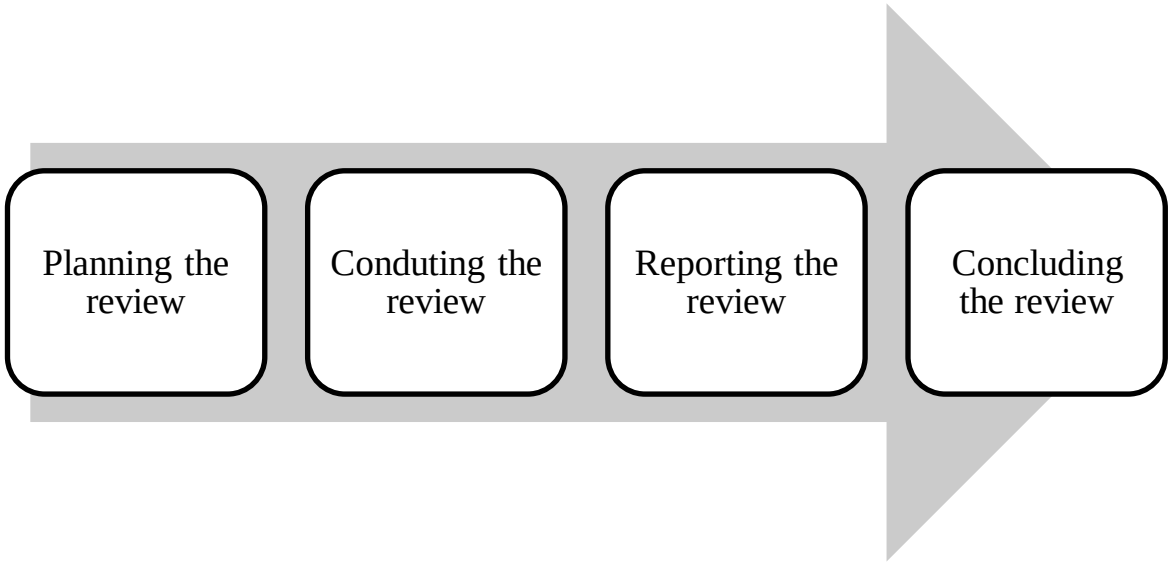


Figure 2.1: Overall Flow of the Literature Review

2.2 Planning the Review

This phase plans the strategies to be used in the process of a literature review. In the current research, the literature review principle methodology is utilized to discover the published papers in cloud storage and blockchain technology. Thus, we organized the review process into two categories:

- The first category covers work related to cloud computing towards increasing the security of cloud data.
- The second category discusses and summarizes the various approaches used for securing cloud data by utilizing blockchain technology.

2.3 Conducting the Review

Defining the research questions is an important step of this review process. The main objective of this research work is to present an outline of the current research on cloud storage and blockchain-based cloud storage techniques. Therefore, we explained six research questions.

- **RQ1: What are the different prevention methods cloud storage providers utilize to anticipate the threats during information sharing?**

To identify the present security methods used in the cloud computing environment.

- **RQ2: What are the different strategies used for preventing the illegal use of data stored on the cloud?**

To identify the different techniques used to prevent the stored information in the cloud.

- **RQ3: What are the main security difficulties to be faced in the future based on cloud storage technology?**

To discuss future cloud computing security techniques.

- **RQ4: What are the current research topics of blockchain technology?**

To study and understand blockchain technology, we collected critical research papers from logical databases and mapped the current research area to understand the blockchain-related topics and concepts.

- **RQ5: What are the current research topics of blockchain technology for cloud storage?**

To study and understand the existing techniques of blockchain technology and cloud storage system. This benefits the researchers to explore topics related to blockchain technology for cloud storage.

- **RQ6: What are the different methods utilized to secure the cloud data using blockchain technology?**

To study and understand the methods used to perform the various security operations on the cloud data using blockchain technology. This helps academicians identify the security-related research topics in blockchain-based cloud storage systems.

2.4 Reporting the Review

In this section, we have described the review of two approaches listed in Section 2.2, and finally, we have identified the research gaps based on this study.

2.4.1 Traditional Cloud Storage Techniques

This section presents research work related to traditional cloud computing techniques utilized to improve the security features.

Ferretti et al. [21] proposed an asymmetric cryptographic scheme with encrypted Bloom filters to allow users to recognize unauthorized modifications in the outsourced data. Also, an analytical model was employed to minimize the network overhead and storage depending on the workload and the database structure. In terms of improved network cost and storage, the effectiveness and performance of the scheme were evaluated. However, the completeness and freshness of data are not guaranteed due to the depreciation of storage.

Li et al. [22] proposed revocable identity-based encryption at the server-side. This scheme offloads most of the key generation-related operations during key issuing and key update processes to a Key Update Cloud Service Provider (KU-CSP), leaving only a constant number of simple operations for Public Key Generator (PKG) and users to perform locally. In this technique, a hybrid key is issued for each user, in which AND gate is involved to connect the subcomponents, namely the identity component and time component. The unrevoked users periodically request the key update for the time component to the newly introduced entity, namely KU-CSP. This scheme achieves efficiency for both computations at PKG and

private key size at the user. Still, it did not resolve the security issue because this scheme is dependent on the trusted third party, i.e., KU-CSP.

Feng et al. [23] suggested a privacy-preserving auditing protocol to audit the security of the data stored in cloud computing. This scheme enables an external auditor to audit users' cloud data without learning the content of the data. The proposed approach efficiently supports dynamic update operations, but it is always assumed that the third-party auditor was fully trusted; this assumption is unrealistic.

Yu et al. [24] developed two privacy-preserving deduplication protocols to eliminate redundant data by considering only one file copy, thus reducing storage and bandwidth requirements. The first protocol, namely the ZEro-knowledge dedUplication reSponse framework (ZEUS), guarantees weaker privacy properties with efficient communication cost. This is the first solution that addresses two side privacy with limited extra communication based on a weak assumption on user behavior. Another protocol, ZEUS+, ensures stronger privacy features with increased communication costs. However the proposed protocols increase the system's overall complexity due to the involvement of lots of computations.

Yan et al. [25] developed a protection scheme to enhance the security of signature information for user data. The realization of dynamic integrity and lattice and Bloom filter methods were utilized where the document and the signature were sent to the Cloud Service Provider (CSP) and Third-Party Auditor (TPA). This scheme was efficient, cost-effective and ultimately improved the utilization of cloud space, but it did not resolve the issues of cloud data security.

Xu et al. [26] proposed an integrity verification algorithm, which presents the generic storage model for different control methods. Verification tags and proofs generating methods were dependent on the index pointers. Furthermore, a random data sampling method was employed to perform random diffusion extraction in the version group, which improved the verification efficiency. The experimental analysis indicated that the proposed scheme was effective. Though the universality and efficiency of data were improved, it required more processing time to generate verification tags and proofs.

Jin et al. [27] highlighted a technique that was the integration of both cryptographic storage design and proof of storage technique. This model was designed with flexible block structures and integrated key regression, broadcast encryption, Merkle hash tree, and proof of

storage. A quick freshness check was involved in protecting the retrieved data from potential replay attacks. An evaluation through a prototype illustrated that the cost and throughput were improved with minimal degradation of security.

Saxena et al. [28] introduced a novel and efficient data integrity verification technique. This technique was built with a variant of the Paillier Homomorphic Cryptography (PHC) system and Homomorphic tag together with combinatorial batch codes. Homomorphic encryption on data blocks had been performed in the PHC system. Each data block was assigned with a special verifiable value by Homomorphic tags. Combinatorial batch codes were derived from storing the integral data into various distributed servers of the cloud. Experimental results have proven that the proposed method provides a better privacy preserved auditing service. However, this was not applicable for large data sizes due to its higher auditing frequency.

Mao et al. [29] proposed a publicly verifiable scheme based on a position-aware Merkle tree to support the dynamic environment and protect the data integrity. The nodes of the new Merkle tree were defined by a 3-tuple adaptation. The users can directly compute the root value and verify the consistency of the challenge-response blocks. Several verification challenges were overcome by this method, but the involvement of the challenge-based method increases the verification time.

Zhang et al. [30] proposed a novel public verification scheme using indistinguishability obfuscation. This scheme significantly reduced the computation overhead, and the batch verification overhead was independent of the number of verification tasks. This scenario is useful in the places where frequent data integrity verifications need to be executed, and the number of verification tasks is numerous. Moreover, this scheme has better communication and computation efficiency. The multiparty computation is not secured due to the variant of computing resources.

Jiang et al. [31] presented an effective public integrity auditing scheme with secure group user revocation. They had designed a concrete scheme based on vector commitment and verifier-local group signature revocation. This system supported public checking and effective user revocation. Additionally, this public checking behavior has the properties like efficiency and confidentiality. The proposed scheme depends on the third-party auditor for the auditing process. Therefore, it suffers from a single point of failure issues.

Zhang et al. [32] deployed the cloud storage auditing model to enable an efficient user revocation process. The proposed user revocation scheme is independent of the number of file blocks processed by the revoked user. It updates the private keys of the non-revoked group users instead of the revoked users. The work utilizes the identity-based encryption algorithm to eliminate the certificate management overhead of traditional public key infrastructure schemes. This scheme requires more processing time to update keys of non-revoked users.

Li et al. [33] suggested an extended access control scheme that encrypts multiple files on the same access level using a proposed file hierarchy CP-ABE technique. The scheme is specifically designed for big companies and institutions with hierarchical data and saves the storage space and computation cost required storing them on cloud servers. It deploys the trusted authority center to maintain the key and access policy-related details, which affects the system's security. If authority is compromised, the malicious user gets all decryption keys and decrypts the stored information.

2.4.2 Blockchain Techniques for Cloud Storage

This section details out the blockchain-based methods used for protecting cloud data.

Wilkinson [34] states a need for a model for cloud storage that is not dependent on the trust factor between the client and the host. All confidential and private data of the client, including the date, filename, or other metadata, needs to be encrypted before facilitating any transfer from the client's computer to the cloud. Legal or political attack vectors cannot be used as a centralized point of attack. Any payments in terms of incentives for the consumer or the resource provider is largely automated and affected with pseudonymous cryptocurrency.

Sukhodoskiy et al. [35] presented a multi-system prototype to maintain access control for datasets stored in the untrusted cloud environment. The designed system utilized a CP-ABE scheme to provide an access control mechanism and maintain immutable records on a blockchain-based decentralized ledger. It only transfers the hash code of ciphertext through the blockchain ledger. The proposed prototype was implemented using the smart contract functionality of the Ethereum platform.

Qiu et al. [36] developed a novel cloud-based integrated development environment using blockchain smart contracts. The proposed scheme was developed for cross-blockchain systems such as Ethereum, Libra, Nervos, Utrain, Cocos, etc., using smart contract

functionality to avoid setting up and building processes. It provides an efficient cloud-based development environment for different blockchain systems. The proposed integrated environment provides a web interface to the blockchain developers, which interacts with the blockchain.

Li et al. [37] suggested a data-sharing model that allows users to share the encoded data using blockchain-based architecture. The proposed blockchain-based decentralized storage architecture encrypts the data using the owner's public key and stores it on the blockchain network with key-related details. It uses the proxy-based re-encryption method to secure the data sharing process in the untrusted environment. Due to the usage of proxy servers, the designed system is free from collusion attacks and maintains Meta key data in blockchain structure.

Li et al. [38] proposed a fair Searchable Symmetric Encryption (SSE) scheme using blockchain technology. The proposed scheme encrypted and stored documents on the cloud server. It ensures the fairness guarantee for the cloud server and the user. If one party does not perform the task honestly, it will cost its deposit. Similarly, the malicious server suffers from service charge loss. Moreover, the proposed scheme does not allow users to verify the searching results, thus increasing the search efficiency.

Zhang et al. [39] suggested a secure and fair payment method for outsourcing services in the cloud computing environment using blockchain technology called BCpay. The architecture and specifications of BCpay were designed without relying on any third parties. The designed architecture follows all or nothing checking proof protocol means either the service provider gets the service fees for the outsourcing service or pays the penalty for the illegal activity. Thus, it provides protection against eavesdropping and malleability attacks.

Lee [40] introduced a digital identity management system using blockchain technology called ID as a Service (IDaaS). The Blockchain-based ID as a Service (BIDaaS) is designed to provide identity and authentication services from the provider to its partners. The BIDaaS provider performs the write operation using the user's public key and digital signature on the private blockchain network. In contrast, the partner has access to the blockchain network with only read permission. The provider maintains the blockchain network and provides service as per the user request by authenticating the user ID.

Yang et al. [41] constructed a novel blockchain-based publicly verifiable data deletion scheme. If the cloud server does not delete the data honestly, this scheme enables the data owner to detect the cloud server's malevolent operation. The blockchain solves the public verification problem in the secure data deletion scheme.

PengCheng et al. [42] developed a blockchain-based integrity protection system for cloud storage. The proposed architecture deploys the challenge-response-based integrity verification process that includes the three phases. The first setup phase pre-processes the file information and generates the tag information. Next, the challenge phase generates the challenge data to check the stored data integrity. Finally, the check proof phase creates the corresponding response per the generated challenge and sends it to the verifier. The architecture also achieves the security features using multiparty calculations.

Bacis et al. [43] designed a decentralized cloud storage system to secure resources and address availability. The system enables resource owners to store resources in the decentralized cloud storage system and provides share and delete services. It uses the All or Nothing strategy to control resource slicing and allocation to the network nodes with availability and security features.

Zhang et al. [44] deployed the blockchain-based system against procrastinating auditors to verify cloud storage integrity. The proposed work implements the certificate-less method to publicly verify the integrity using blockchain technology. It allows the key auditor to store each verification result in a transaction on the blockchain network.

Wang et al. [45] constructed a distributed cloud storage architecture using blockchain technology with an access control mechanism. The architecture utilizes the Ethereum blockchain network with a CP-ABE scheme without trusted authority. Furthermore, it sets the valid access period with the user data so that only during valid access periods, the user is allowed to decrypt the ciphertext, thus maintaining the access control feature.

Awadallah et al. [46] developed blockchain-based integrated cloud architecture for maintaining security features. The proposed scheme implements a homomorphic encryption scheme on integrated blockchain-based cloud architecture to ensure integrity security features. It uses the Byzantine Fault Tolerance algorithm to construct a blockchain network of cloud service providers to handle user requirements. Each cloud service provider calculated

the master hash value of their database to store it on the Ethereum blockchain network. The user compares the master values to detect tampering and check the cloud data integrity.

Ali et al. [47] designed a secure log management system for cloud computing using blockchain technology. The constructed architecture ensures the security of audit logs to increase user trust in the cloud computing environment. It provides the immutability feature using the distributed ledger and utilizes Elasticsearch to save JSON files that are semantically rich and machine-understandable. The system is deployed in a heterogeneous environment and monitors the activities of administrators and users troubleshoot to figure out the security issues in the information system.

2.4.3 Identification of Research Gaps

- Most studies have used the centralized approach for storing the data on the cloud, and the need for trusted third-party harms the privacy of user's data. Thus, to improve the security of the users' data and remove centralized authority, the blockchain technology, a distributed database system is required.
- The current distributed cloud storage systems store the data in several data centers that are not fully distributed and involve the semi-trusted authorities/auditors to manage the cloud services. Since the designed systems are semi-distributed, it affects the trust of the cloud user and compromises the security of the stored data. Therefore, distributed ledger technology improves accuracy, which is difficult to tamper, forge and trace. The blockchain records all the information of the transactions, and once the data enter the blockchain, almost nobody can change it.
- Few studies focused on the user revocation process. The integration of blockchain and cloud storage systems helps to enhance the user revocation process without involving any proxy servers.
- The smart contract is a blockchain-based technology that helps us define the rules and regulations and improves security. However, only a few studies utilized the concept of smart contracts to provide security features to the cloud user without using any trusted third parties, semi-trusted auditors, and verifiers.
- Most studies focused on key management schemes, data deletion schemes, searchable schemes, and cloud storage payment schemes using blockchain

technology. Very few studies focused on the security of user's data, distributed key management process, access policy, integrity problem, user revocation process, and various other security services.

- There are several performance evaluation parameters used in cloud storage techniques to analyze network performance. Most studies used the transmission delay, file security, and computation cost as the evaluation parameters. We work on computation cost, latency, throughput, processing time, optimizing storage space, improving accuracy and security of the storage space.

2.5 Concluding the Review

From the literature review, it has been established that there are problems in terms of security, privacy, integrity, access, and user revocation process in the existing cloud storage systems. Hence, there is a scope for developing efficient, secure, and complete distributed blockchain-based architecture for the cloud storage system to enhance cloud services' performance, efficiency, and reliability.

CHAPTER 3

PRELIMINARIES

This chapter explains the essential components of the proposed blockchain-based cloud storage architecture. It covers the technical details related to the proposed blockchain structure such as block, block hash, nonce, transaction, consensus, mining etc. The chapter also describes cryptographic algorithms and smart contract functionality utilized in the proposed architecture.

3.1 Blockchain Technology

The developments in cryptography and distributed computing have introduced an advanced technology called blockchain. Blockchain is a distributed ledger that replicates and exchanges data through peer-to-peer networks. Blockchain was initially introduced by an unknown person, Satoshi Nakamoto, who created Bitcoin to trade digital currencies directly without third parties [16]. Nakamoto developed the paradigm of a network of nodes working to maintain a decentralized and secure database. The blockchain is the core technology behind cryptocurrencies- a shared public database or a continuously updated registry of all transactions. Blockchain can be regarded as a technical breakthrough and financial advancement [48, 49]. It provides a solution to any problem using a trustworthy ledger in a decentralized setting where it is impossible to trust actors, humans, and computers completely.

Furthermore, the blockchain follows a series of procedures and cryptographic mechanisms applied to a shared network to secure data storage within a distributed database composed of authenticated blocks encapsulating the data. It stores the data as an ordered list of blocks. By referencing the previous block's hash, each block distinguishes by the hash sequence and ties

to the preceding block. The only anomaly is the first block (called "Genesis block"), which does not have the previous block's hash value, known as the ancestor block. The data is passed to the miners, who verify it by solving mathematical puzzles and attaining consensus. The three key principles ensuring the system's functionality are 1) blocks and hashing, 2) mining, and 3) consensus [50]. The architecture of the blockchain is illustrated in Fig. 3.1.

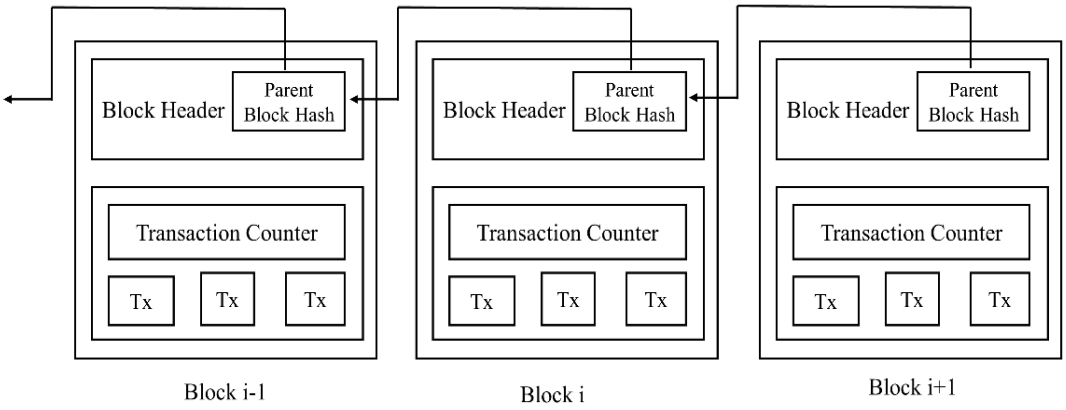


Figure 3.1: Blockchain Architecture

Fig. 3.2 depicts the blockchain structure in which each block contains the previous block's hash, and Nonce denotes the solution to the proof of work puzzle. The timestamp represents the block generation time, and the Merkle root authenticates all the transactions.

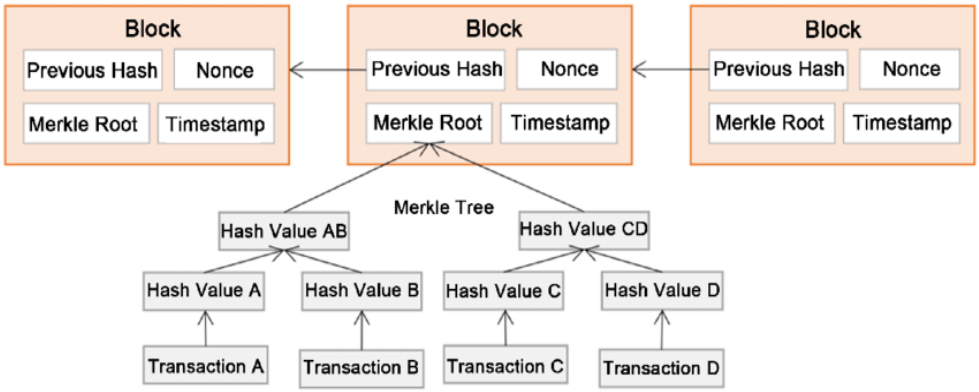


Figure 3.2: Structure of the Blockchain

$$Blockhash = (Prehash\,|timestamp|\,nonce)$$

Key Characteristics of Blockchain

- **Distributed:** The distributed environment uses the standard protocol, which ensures that every node receives each transaction and uses predefined rules for grouping the

transactions into blocks after processing. The blockchain is designed for distributing and synchronizing data across multiple networks.

- **Decentralization:** It is the core strength of blockchain because each node holds a record of all transaction data, so there is no need for a central authority. This prevents the single point of failure vulnerability. In the blockchain network, there is no single authority and no service fees, and the consensus algorithms are used to maintain the data consistency.
- **Consensus:** Consensus algorithms maintain data consistency within the blockchain and keep incorrect or false transactions away from the blockchain network [51]. All nodes must agree by executing the standard consensus algorithms to ensure the integrity of transaction data. Moreover, there must be an assertion between every member before one can execute the transaction; the transaction should only be valid [52]. This procedure is called a consensus.
- **Anonymity:** In the blockchain, every user can interact with a created address. The system will not disclose the user's actual details; however, the members can view the encoded transaction details.
- **Traceable:** The blockchain is time-stamped and digitally signed, which implies that the association can follow back to an explicit time for every transaction and further distinguish the relating party on the blockchain [53]. Consequently, each block is permanently connected to the past block [54].

3.2 Types of Blockchain

Blockchain technology is the future step in a peer-to-peer economy. A distributed network combines distributed data storage, cryptographic algorithms, and decentralized consensus mechanisms [55]. Blockchain technology is categorized into three types of structures, as explained below:

1. Public Blockchain: A public blockchain enables anyone to join the network. This means anyone can write, read and review without the permission of blockchain. Furthermore, anybody can survey the transaction at a given time since the blockchain is open and straightforward to the client. Moreover, this type of blockchain is decentralized in nature as nobody is in control of public blockchain due to the consensus components.

The public blockchain gives approval for the transactions in the system for anyone can copy or download the code. Therefore, anybody can send the transaction worldwide by blockchain

technology, and they can validate and verify the transaction. Some examples of blockchain are Litecoin, Bitcoin, and Ethereum.

2. Private Blockchain: In a private blockchain, transactional control such as creating, viewing (read), and validating transactions (write) is handled by their operators. It is a closed network that offers only specific pre-chosen entities to create a new transaction on the blockchain. As this is an evolving technology, it may encounter unexpected consequences in the context of security and privacy [56].

3. Consortium Blockchain: The third type of blockchain is considered as a hybrid or the consortium of federated blockchain. It is a kind of private blockchain. This blockchain is worked under the control of a group of organizations. On account of consortium blockchain, not every person has equal privileges of approval of transactions. However, just a couple of individuals are given certain benefits over approving the transactions.

3.3 Consensus Mechanism

There are many consensus algorithms utilized in the blockchain network. The consensus algorithm ensures the blockchain network participant agree on a common decision to create and add a new block in the blockchain network without any centralized authority. Major consensus mechanisms in the existing blockchain systems are as follows:

3.3.1 Proof of Work (PoW)

Proof of Work is used to create the new blocks and confirm the transaction in the blockchain system. In PoW, miners compete against each other to complete the transactions on the system and get remunerated as shown in Fig. 3.3. The users send computerized tokens to each other in a network. Further, the complete transactions of the block are gathered and controlled by the decentralized ledger. The PoW is mainly used for confirming the transactions and organize the blocks on blockchain technology. They use the special nodes known as miners and the process is also known as mining. The miners utilize tons of resources such as hardware, integrated circuits, and electricity to solve the block puzzle and once it is solved the block is added in the blockchain network.

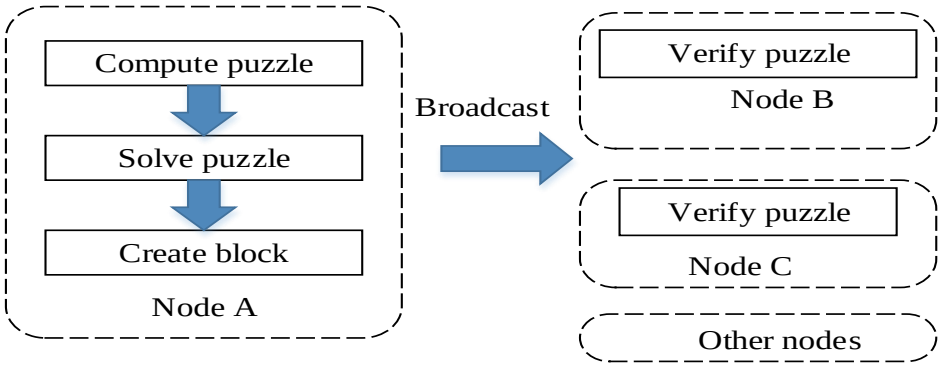


Figure 3.3: PoW Consensus Mechanism

3.3.2 Proof of Stack (PoS)

The primary energy-saving alternative of PoW is the Proof of Stack algorithm. PoS algorithm governs the rules in the blockchain network and the creation of its native coin; that is, PoS has the same objective as a PoW algorithm in the sense that it achieves the consensus mechanism. However, unlike PoW, there are no miners involved in the process. Instead, participants in the network who want to prove the validity of network transactions and create blocks in a PoS network have to hold a certain stake in the network, for instance, by placing a certain amount of the network's currency in a wallet connected to its blockchain. This is known as "placing a stake" or "staking". A block creator in a PoS system is limited to creating blocks proportionate to his or her stake in the network. Thus, PoS networks are based on deterministic algorithms, meaning that validators of blocks are elected depending on the nature of the stake.

3.3.3 Proof-of-Authority (PoA)

The blocks and transactions are approved by affirmed accounts in the PoA consensus mechanism known as validators. The validators run software to complete the validation process and put transactions in blocks. It is an automatic process that needs not be continuously monitored by the validators. The PoA-based framework uses the incentive method to give the rights to participants to act as validators. It assigns the reputation to the identity of the validator to perform the validation process. The PoA algorithm is considered more robust than the PoS algorithms as in PoS, stakes between two parties may be even, and it does not consider total holdings [59]. This means that incentives can be unbalanced. On the other hand, PoA only allows non-consecutive block approval from anyone validator, meaning that the risk of serious damage is centralized to the authority node.

3.4 Cryptographic Algorithms

This section presents cryptographic algorithms utilized in the proposed blockchain-based cloud storage architecture.

3.4.1 Hashing

The hashing is the main core part of the blockchain network. The hashing algorithm calculates the transaction hash value to generate the fixed-length output. The primary blockchain application, Bitcoin cryptocurrency, utilizes an SHA-256 hashing algorithm to store, manage, and create transactions and blockchain structure. The unique hashing features such as collision-resistant unique always generate the different hash values for different inputs [57]. It always generates the fixed size output even for the different input values, and it is impossible to retain the original value from the generated hashes.

3.4.2 Merkle Tree

The Merkle tree utilizes cryptographic hash functions to create the binary tree structure by combining the leaf nodes to form intermediate nodes and the root node, as shown in Fig. 3.4. The blockchain transaction data is divided into multiple chunks stored at the bottom of the tree structure as leaf nodes, and then intermediate nodes are created by combining the hash values of the child nodes. Finally, the Merkle root combines its two child nodes and forms the tree's top root node. Merkle root helps validate the data stored in leaf nodes as hash values and considers all transactions signatures that are part of the block in the blockchain network. The Merkle structure requires less storage space and easy computing.

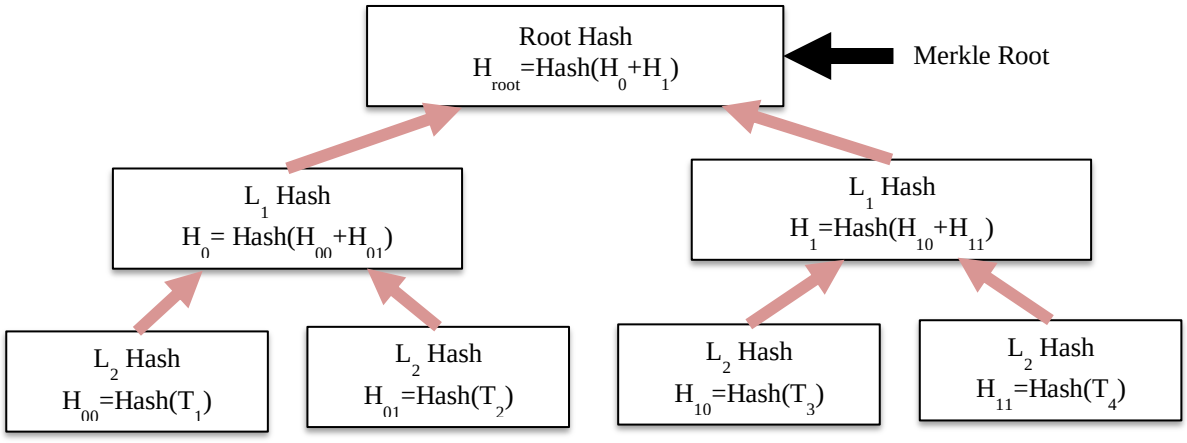


Figure 3.4: Structure of the Merkle Tree

3.4.3 CP-ABE Algorithm

Encryption technology can be considered a security guarantee for confidentiality and data access control. However, achieving access control and integrity for encrypted data poses a significant challenge. Bethencourt was the first to implement CP-ABE in 2007. A ciphertext is linked to an access structure in CP-ABE mode, and a user's secret key is linked to attributes [60]. The user can decrypt the given ciphertext if and only if his attribute set complies with the data owner's access policy. According to the attribute, the data user obtains the corresponding key from the attribute authority core. According to the access policy, the data owner will monitor who has access to the data. The CP-ABE scheme needs one or more completely trusted attribute authorities or central authorities. Due to the presence of centralized authority, it suffers from a single point of failure issue. Therefore, the decentralized systems are utilized to remove the need for trusted authorities, ensure security features, and prevent a single point of failure. Although some people have researched blockchain-based security schemes recently, most suggest a mechanism or concept for such systems. There is no straightforward approach for realizing the convergence of blockchain technology's decentralization concept with security methods. This is an environment where there is still a lot of work to be done. As a result, blockchain-based decentralized cloud storage system with security methods research is valuable and essential.

3.5 Smart Contract

The smart contract idea was developed a long time ago, but it was introduced recently. The idea of a smart contract was articulated about twenty years ago by a cryptographer scholar, Nick Szabo [61, 62]. The fundamental principle is to insert contractual concepts into computer components such as liens, trusts, etc. He proposed four fundamental principles of contract design: evaluation, validation, privacy, and enforceability. Based on Szabo's concept, contracting parties should evaluate their success, check whether the contract was performed or violated. Also, the smart contract protects all parties' privacy and distributes the details as much as is required, and eventually, it would execute automatically.

Nevertheless, the necessary architecture and specifications were not available at that time. The concept remained only an abstract term; today, smart contract deployment has become realistic by advancing blockchain technology. Smart contracts are a package of codes that encode and replicate real-world contractual agreements in the computer domain. A basic

principle for contracts is to create a legal agreement between two or more parties that each party is required to meet its contractual obligations. The important consideration is that the contract will be governed by a legit entity (organization). Smart contracts are eliminating the trusted third parties; that is, the mediators between contract members. They manage the mediators by using automatic execution of programs in a blockchain network, which is decentralized and evaluated by the network nodes. The smart contract also allows transactions between untrusted parties (i) mediator commission fees, (ii) trusted-party dependency, and (iii) the counterparties' need for mutual interaction.

Smart contracts comprise a contract space, a balance, and code. It can be generated and granted access to any node in a network simply by publishing a transaction to the blockchain. When included in the blockchain, the smart contract code is fixed and cannot be changed. A network of miners who are accountable for managing the blockchain runs smart contracts. Miners agree on the smart contract's implementation result and upgrade the blockchain accordingly. Once implemented, each contract is assigned a 160-bit address and is executed using this address if a transaction is generated. The smart contract uses various platforms for the development of applications. Ethereum is the largest and most important platform for developing decentralized applications, ranging from predicting markets and identity systems to other economic applications. Bitcoin is also a blockchain software that offered a modern, improved exchange of money, emerging markets, and new independent decentralized organizations [63]. While Bitcoin's primary purpose is to transfer capital, its blockchain's immutability and openness have facilitated the creation of protocols that implement smart contracts. Many smart contracts store the blockchain data via the Bitcoin scripting language [64]. Apart from Bitcoin and Ethereum, there are continually evolving numbers of alternative systems derived from or are separate from the initial Bitcoin network and provide enhancements and innovative solutions for different impediments encountered in the former.

CHAPTER 4

BLOCKCHAIN-BASED DECENTRALIZED CLOUD STORAGE SYSTEM

It has been observed from chapter 2 that the traditional cloud storage systems suffer from various security and privacy issues such as single point of failure, loss of data control, data breach problems, integrity, and access control issues. This chapter proposes decentralized cloud storage systems using blockchain technology to ensure security features without involving trusted third parties.

The proposed architecture implements the CP-ABE algorithm to ensure confidentiality, integrity, and availability features. In addition, it provides a complete distributed environment for storing key-related information, outsourcing-related details, and integrity checking information in the blockchain structure. The experimental results, analysis, and performance evaluation show that our proposed architecture provides a feasible and reliable cloud environment.

4.1 Introduction

Data are presented in a variety of forms whenever it is needed as a valuable resource. Massive amounts of data on storage devices create maintenance issues [65]. Therefore, many users prefer to outsource their data to the cloud to alleviate the heavy burden of storage. A cloud is an excellent infrastructure that provides us with many benefits and functionality. Nevertheless, data security issues arise in the cloud storage system [66, 67]. Data may leak while users store their data in the cloud; it's critical to protect privacy [68] and provide data protection [69]. With the rapid advancement of cloud computing and big data technologies, an increasing number of companies and individuals opt to store their data in the cloud. The

majority of data stored in the cloud is susceptible, such as personal medical records, multimedia documents, and company internal data [70-72].

Blockchain is a decentralized database defined as a linked chain of blocks and is difficult to tamper with, forge, or trace. The blockchain stores all transaction information, and almost no one can alter the data once it has been entered. As a result, blockchain technology is simpler and more stable than other security technologies. Blockchain technology in cloud storage architecture provides a more secure environment to the users. Instead of concentrating resources in a single data center or server, a blockchain network distributes them among nodes. Blockchain-based cloud storage eliminates the need for users to rely on a central authority to maintain their data, thus helping to eliminate the risks of large-scale data breaches. The blockchain splits the users' files into multiple shards called blocks. Each block is encrypted, giving each block of data a unique hash and then distributing that data across multiple computers or nodes on the network. This arrangement provides greater privacy compared to a centralized cloud storage system.

Furthermore, many researchers have already developed various blockchain-based cloud systems [73-76]. For example, in [73], the authors developed a distributed cloud storage system that divides and stores data shards in a peer-to-peer blockchain network. However, the designed scheme lacks the integrity checking process. Wang et al. [74] use Ethereum's smart contract functionality to provide a decentralized access control system. It modifies the traditional encryption algorithm to include attribute details of users and generates access rules. The architecture is based on a semi-honest cloud storage system, thus affecting cloud data security. In [75], the authors develop a distributed auditing approach for cloud storage and eliminate third-party auditors. It is analyzed that the designed scheme only considers the auditing and integrity security features however lacks the access control mechanism. Li et al. [76] present an optimization method that uses deduplication schemes to allocate files on multiple servers and maintain details in the blockchain storage system. They use smart contract functions to ensure a secure optimization method without any trusted third party. The suggested scheme only considers the mechanism to reduce the redundant data without considering the integrity and access control methods. Similarly, most of the research papers focus on cloud storage-specific features like access control [77-79], integrity checking [80, 81], encryption schemes [82], and searching processes [83]. Unlike the above work, we proposed a complete blockchain-based cloud storage framework with multiple features like

access control, integrity checking, optimization features, and network performance evaluation.

To deal with the disadvantages and challenges mentioned above, we designed Java-based blockchain architecture for the cloud storage system to provide a secure environment. The proposed architecture implements CP-ABE to provide a key generation mechanism using bilinear mapping-based cryptography and perform data access control mechanisms. Data owners manage the key related and user access policy details in a distributed manner by utilizing the blockchain structure and providing a more robust environment. The blockchain network keeps Meta details of user data and tracks all access and validation records. The architecture also deploys a Honeybee optimization algorithm to optimize the resource utilization on the cloud storage system and minimize transaction processing time. By using blockchain technology, we achieve decentralization with security without a trusted central authority. It provides four primary services to ensure security features in a cloud storage system:

1. A registration process is designed to register data owners and users using a key generation technique to provide an authentication feature.

2. The data owners save the user data Meta details in the blockchain structure and set access rules to maintain authorization.

3. Data owners maintain data integrity using the Merkle root concept.

4. The cloud storage system stores the original data and uses the optimization algorithm to reduce the transaction processing time.

4.2 Proposed Work

In this section, the entire proposed work has been discussed in detail.

4.2.1 System Model

Fig. 4.1 shows the decentralized permissioned blockchain architecture for a cloud storage system with security features. The proposed system model consists of four core entities:

- **Blockchain:** Blockchain maintains the transparent, tamper-proof structure to store the Meta details of the data and users. Every function call saves the related information in

the blockchain structure. Therefore, the data transfer between the users of the proposed architecture is non-tamper and transparent.

- **Data Owner:** Data owner's task is to generate system parameters and keys for the users according to their attributes/identity. The data owner is also responsible for outsourcing encrypted files, managing access policies, checking integrity, and maintaining Meta information in the blockchain structure.
- **User:** Users can access the data when their attributes satisfy the ciphertext's access policies. The user can use the decryption algorithm to decrypt the file using the secret key and obtain the plaintext.
- **Cloud Storage:** Cloud storage is responsible for storing encrypted files uploaded by the data owners. Data files are first divided into multiple shards of the same size and distributed on multiple servers. The proposed architecture uses the Honeybee optimization algorithm [84] to effectively manage the resources during the data transaction process and minimize the processing time.

Table 4.1: Notation and Descriptions

Notations	**Description**
GP	Global parameters $(p, \mathbb{G}, g, e, \mathbb{G}_T, H)$
PK	Public key of the data owner
MK	Master key of the data owner
$User_{id}$	Public key of the user
SK_A	Secret key used for encryption and decryption process
$Merkle_{root}$	SHA256 generated file Merkle root
CT	Encrypted file
F	File of the user
$\{A_1, A_2, A_3, \ldots, A_n\}$	User's attribute list
DO	Data owner
F_{id}	SHA256 generated file hash
$TransID$	Blockchain network generated transaction-ID
S	Number of servers
$\{B_1, B_2 \ldots\ldots B_n\}$	Number of blocks in each server

The proposed architecture workflow, as shown in Fig. 4.1, is as follows:

1. Data owners send key generation requests to generate global parameters and public and master keys to register the proposed architecture.
2. The blockchain network executes the key generation function and generates a public key and master key for the data owner.
3. The user sends a registration request to the data owner with an attribute list like user ID, department, email id, contact number, address, DOB, etc.
4. The data owner saves user details in the proposed architecture and generates access policies corresponding to its attribute list using the CP-ABE algorithm. The public key corresponds to the user ID shared in the attribute list. The data owner generates a secret key using the bilinear pairing-based CP-ABE algorithm.
5. The data owner generates the user keys and maps each other using generated keys, and this information is saved in the proposed architecture.
6. The data owner is responsible for sharing a secret key with the user using the Diffie-Hellman algorithm. This shared secret key is utilized to encrypt and decrypt the data using the AES algorithm.
7. The data owner uploads the data to the cloud storage using the proposed architecture functionality. The data owner generates the file's Meta details like file hash, Merkle root, mapping details of file and owner/user, and storing them in the blockchain structure.
8. The data owner uses the generated secret key to encrypt and store the data in cloud storage.
9. The cloud storage servers store the data and optimize the resource to minimize the execution time and overhead while storing it. The cloud storage follows the optimization approach to balance the servers' data load and reduce the transactional process time.
10. The user sends the access request to the corresponding data owner along with the file ID.
11. The data owner fetches the user and file mapped information and corresponding access policy from the blockchain network.
12. After checking the access details, access may be granted or denied to the user.

13. After the access verification phase, the encrypted data fetch from the cloud storage and sent to the requested user.
14. The user uses the secret key to decrypt the received encrypted data and access the plaintext.
15. The user also checks the integrity of the received data by sending the request to the data owner.
16. The data owner fetches the user and files details from the blockchain, performs the integrity verification process using the Merkle root method, and returns the result to the user.

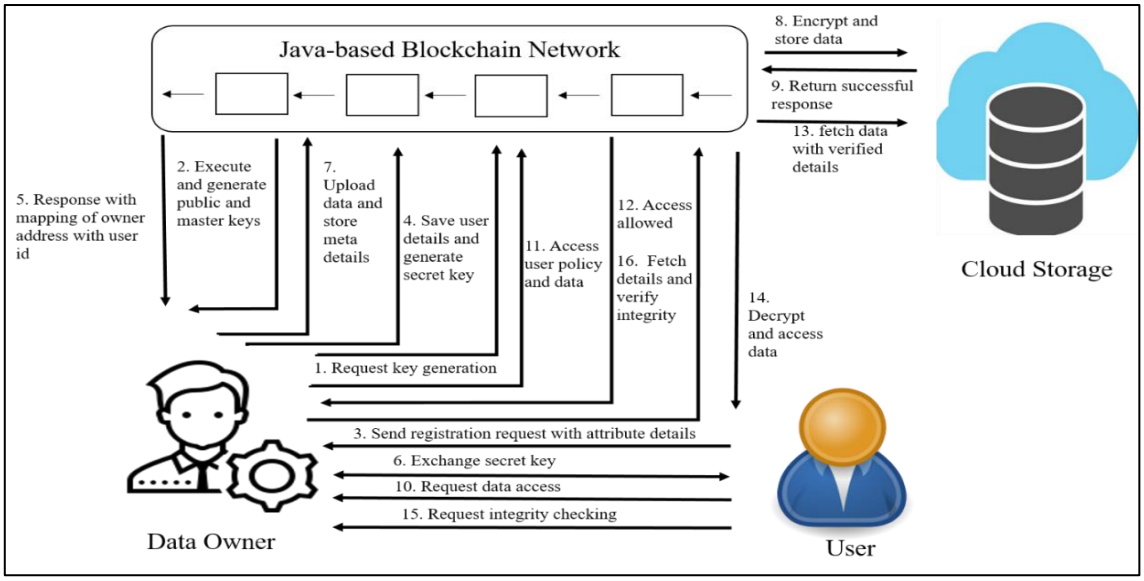


Figure 4.1: System Model of the Proposed Work

The architecture is composed of the following algorithms to achieve the basic functionalities:

$\boldsymbol{Global_{setup}(GP)}$**:** The proposed architecture uses a CP-ABE algorithm. The blockchain network selects a bilinear group $\mathbb{G}$ of prime order p and generator g and two random numbers $a, b \in \mathbb{Z}_p$. The data owner executes the global setup algorithm and takes the global parameter GP as input and output public key PK, and master key MK. Let p be a prime number and $\mathbb{G}$, $\mathbb{G}_T$ be multiplicative cyclic groups of order p. A map e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ satisfying the following properties is called a bilinear map or bilinear pairing.

- $e(u^a, v^b) = e(u, v)^{ab}, \forall\, u, v \in \mathbb{G}, a, b \in \mathbb{Z}_p$
- If g is a generator of $\mathbb{G}$, then $e(g, g)$ is a generator of $\mathbb{G}_T$
- $e(u, v)$ is efficiently computable for all $u, v \in \mathbb{G}$
- Let $H: \{0,1\}^* \rightarrow \mathbb{G}$ be a hash function that maps attribute to the random element of $\mathbb{G}$.

$\boldsymbol{Key}_{\boldsymbol{generation}}(\boldsymbol{p}, \mathbb{G}, \boldsymbol{g}, \boldsymbol{e}, \mathbb{G}_{\boldsymbol{T}}, \boldsymbol{H})$: The data owner generates the public and master key to register the proposed system using a key generation algorithm. Also, users send registration requests to data owners with the attribute set containing details like user ID, email ID, department, contact details, etc. The user attribute list is represented as $\{A_1, A_2, A_3, \ldots, A_n\}$. The data owner takes global parameters $GP(p, \mathbb{G}, g, e, \mathbb{G}_T, H)$ and master key MK as input, assigns attribute policy to the user, and generates a secret key SK and public key $User_{id} = \{A_1 \epsilon A_n\}$. The owner executes Algorithm 4.1 of the proposed architecture to complete the key generation process. The mathematical notation of the key generation process is given below:

1. Global parameters $GP(p, \mathbb{G}, g, e, \mathbb{G}_T, H)$

2. Each user attribute list is represented as: $A_i = \{A_1, A_2, A_3, \ldots \ldots, A_n\} = \{1,2,3, \ldots \ldots, n\}$

3. Then parameter list is denoted as: $params = [p, g, e, g_2, h, Y = e(g, g_2)^{\alpha}, T_1, T_2, T_3, \ldots, T_n]$ $where\ \alpha \overset{R}{\leftarrow} \mathbb{Z}_p, g_2, h, T_i \overset{R}{\leftarrow} \mathbb{G}$

4. Data owner public key is represented as: $PK = \{\mathbb{G}, g, g^a, g^b, e(g, g)^b, H\}$

$$where\ (a, b)\ \in \mathbb{Z}_p$$

5. Data owners' master keys are represented as: $MK_i = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$

6. The secret key of each user having A attributes can be generated as:

$$SK_A = [A, d_1 = g^r, d_2 = g_2^{\alpha} h^r, d_i = T_i^r, \forall i\ \in A], where\ r \overset{R}{\leftarrow} \mathbb{Z}_p$$

$\boldsymbol{Encrypt}(\boldsymbol{GP}, \boldsymbol{F}, \boldsymbol{SK_A})$: The owner uses the CP-ABE encryption algorithm to encrypt the data. The encryption algorithm takes global parameters GP, file F, and secret key SK_A as input and generates ciphertext CT. Before the owner uploads the file on the cloud, the file is divided into multiple shards, and $Merkle_{root}$ is calculated using the $SHA256$ algorithm and maintaining this Meta information in the blockchain network. The user details, application binary interface owner public key, are represented in the form of the JSON object, and the following steps are involved for uploading the file F.

1. The owner uses the Diffie-Hellman algorithm to exchange common key SK by using user public key and identifier. Then, a common key is utilized to perform a systematic encryption technique using an AES algorithm. This information is stored on the blockchain by executing Algorithm 4.2.

2. The owner divides the file into multiple shards to calculate the $Merkle_{root}$ by following the details given in Algorithm 4.2. The calculated $Merkle_{root}$ stores on the blockchain network.

3. The owner then uses the encryption algorithm to encrypt the file F with the common secret key using AES symmetric encryption algorithm. The user ID $User_{id}$ (public key) is stored with the encoding output and F_{id} on the cloud server $\{User_{id,}F_{id}, CT\}$. The encryption process follows the Equation 4.1.

$$Encrypt(params[p, g, e, g_2, h, Y = e(g, g_2)^{\alpha}, T_1, T_2, .., T_n], W = i_1 \wedge i_2 \wedge \ldots \wedge i_k, F \in \mathbb{G}_T)$$

$$CT = [W, C_1, C_2, C_3], where\ C_1 = F\,.Y^s, C_2 = g^s, C_3 = \left(h\prod_{j=1}^{k} T_{ij}\right)^s, s \overset{R}{\leftarrow} \mathbb{Z}_p \qquad (4.1)$$

$\boldsymbol{Decrypt(GP, CT, SK_A)}$: The decryption algorithm requires the global parameters GP, encrypted text CT, and common shared key SK_A. The owner first obtains the $User_{id}$ and F_{id} from the blockchain structure and checks to see if the requested file exists in the blockchain network. If the requested file does not exist, then the decryption process is discarded. Otherwise, the owner allows a user to follow the decryption process and obtain the plaintext using the generated common key, global parameters, and ciphertext. If the attribute set satisfies the access policy condition, then decryption results in plaintext; otherwise, it will fail. The decryption process of the CP-ABE along with correctness proof is as follows:

1. $Decrypt(params, SK_A = [A, d_1, d_2, d_i, \forall i \in A], CT = [W, C_1, C_2, C_3])$

- If $\{i_1, i_2, \ldots., i_k\} \not\subset A, decryption fails$
- If $\{i_1, i_2, \ldots., i_k\} \subset A, then d = d_2 \prod_{j=1}^{k} d_{ij}\ and$

$$F = \frac{C_1.\,e(d_1, C_3)}{e(d, C_2)}$$

2. $Correctness(SK_A, CT)$

- $SK_A = [A, d_1 = g_r, d_2 = g_2^{\alpha} h^r, d_i = T_i^r, \forall i \epsilon A]$
- $CT = \left[W, C_1 = F.Y^s, C_2 = g^s, C_3 = \left(h\prod_{j=1}^{k} T_{ij}\right)^s\right],$

  $where\ Y = e(g, g_2)^{\alpha} and\ d = d_2 \prod_{j=1}^{k} d_{ij}$

$$\frac{C_1.\,e(d_1, C_3)}{e(d, C_2)} = \frac{F.\,e(g, g_2)^{\alpha s}\,.\,e(g^r, \left(h\prod_{j=1}^{k} T_{ij}\right)^s)}{e(g_2^{\alpha} h^r \prod_{j=1}^{k} T_{ij}^r, g_s)}$$

$$= \frac{F.e(g,g_2)^{\alpha s}.e(g^r,\left(h\prod_{j=1}^{k}T_{ij}\right)^s)}{e(g_2^{\alpha},g^s)\,.\,e\left(h\prod_{j=1}^{k}T_{ij},g\right)^{rs}}$$

$$= F$$

4.2.2 Smart Contract Function

This section describes the logic and algorithms of the proposed architecture. In the proposed blockchain network, operations are designed in the Java programming language. Various smart contract functions are created and executed in this architecture to provide different functionalities. The function defined the multiple variables and structures when they are created:

1. The data owner is the controller of the proposed architecture, and its public key is treated as the owner's address. All network addresses that execute different functions are considered as a sender.

2. The proposed blockchain structure maintains Meta details of users and data. The data owner utilizes this information to maintain the access control and integrity validation process. The data owner takes the user and file details to sustain the Metadata in the blockchain structure using a hashing algorithm.

3. The proposed architecture facilitates the two main structures: storing the actual file in the cloud servers and storing user and file Meta information in the blockchain network to provide an effective and reliable environment.

The following functions are designed in the proposed blockchain-based architecture:

User Registration Function: Each data owner executes the key generation algorithm to generate public and master keys. The system utilizes the CP-ABE algorithm by using the bilinear mapping technique. After generating the master key, the data owner uses the same algorithm to generate the secret key for the users. The generated secret key is shared with the user using the Diffie-Hellman algorithm based on the $User_{id}$ (public key) shared in the attribute list. We design the registration flow of data owner and user as shown in Fig. 4.2.

i. Data owner sends a registration request to the blockchain network.

$$DO \rightarrow Blockchain: Registration_{request}$$

ii. Blockchain sets up the global parameters and shares with the data owner in response.

$$Blockchain \rightarrow DO: Key_{generation}(p, \mathbb{G}, g, e, \mathbb{G}_{T,} H)$$

iii. Data owner generates public and master keys by using shared global parameters and the registration process is completed.

$$DO: PK_i = \{\mathbb{G}, g, g^a, g^b, e(g, g)^b, H\}; MK_i = \{a, b \in \mathbb{Z}_p\}$$

iv. User sends a registration request to the data owner with attributes details.

$$User \rightarrow DO: Registration_{request}(A_1, A_2, .., A_n)$$

v. Data owner stores user details in the blockchain system and executes key generation algorithm with master key and attributes as parameters.

$$DO \rightarrow Blockchain: Store_{Request} \leftarrow Key_{generation}(MK_i, A_j)$$

vi. Data owner generates a public key and common secret key by using the key generation algorithm.

$$DO: SK_j = \{p, \mathbb{G}, g, e, \mathbb{G}_T, MK_i, A_j\}; User_{id} = \{A_1 \in A_j\}$$

vii. Data owner exchanges the generated common secret key with the user using Diffie-Hellman key exchange protocol.

$$DO \rightarrow User: Key_{exchange}(SK_{j,} PK_j)$$

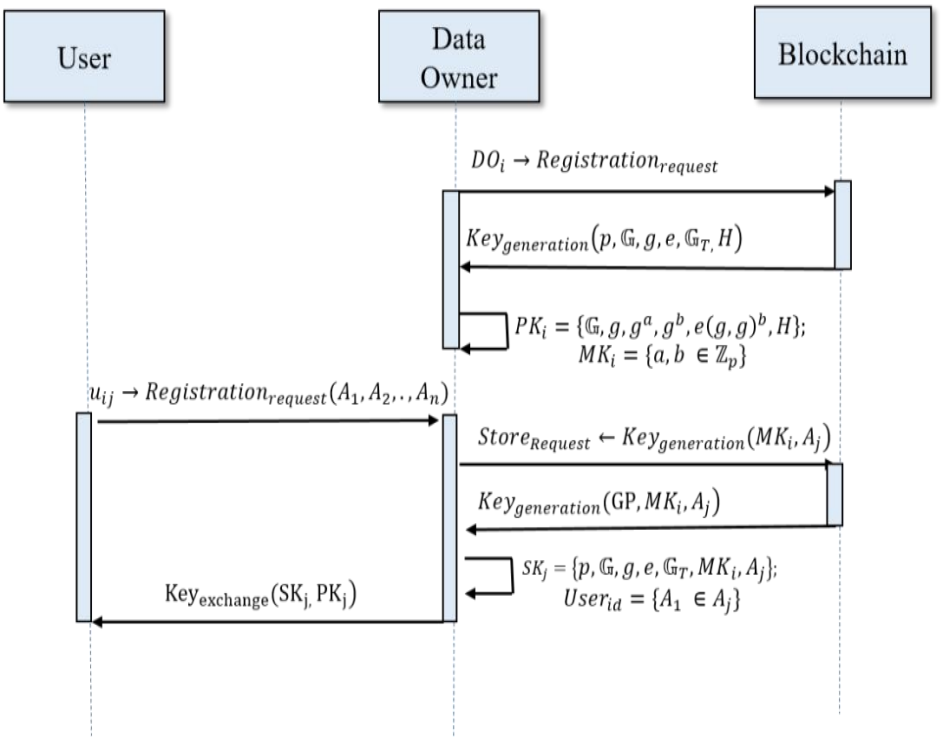


Figure 4.2: Registration Flow of Data Owner and User

Algorithm 4.1: Key Generation

$Key_{generation}(GP)$

Input: Data Owner Set $DO = \{DO_1, DO_2, DO_3, \ldots, DO_i\};\ 1 \leq i \leq n$

User Set $u = \{u_1, u_2, u_3, \ldots, u_j\}\ where\ DO_i \in \{u_1, u_2, u_3, \ldots, u_j\}$

$u_j \leftarrow Attribute_{list}(A_1, A_2, \ldots, A_j)$

Output: Master Key MK and Secret Key SK

1. $for(i = 1; i \leq n, i++)$
2. Generate Master Keys for Data Owner, DO_i: $MK_i \leftarrow Key_{generation}(GP)$
3. $end\ for$
4. $for\ (i = 1; i \leq n; i++)$
5. $for\ (j = 1; j \leq n; j++)$
6. Generate Secret Keys SK_{ij} for User $u_{ij} \leftarrow Key_{generation}(GP, MK_i, A_j)$
7. $end\ for$
8. $end\ for$

In Algorithm 4.1, line 2 generates a master key for the data owner using a key generation process. In contrast, line 6 generates the secret keys for the users belonging to different data owner groups, e.g., $DO_1 = \{u_{11}, u_{12}, \ldots,), DO_2 = \{u_{21}, u_{22}, \ldots\}$ etc. The time complexity of the algorithm is $O(n^2)$.

Data Outsourcing Function: The data owners execute this function to store the user data on the cloud storage and maintain Meta details on the blockchain structure. Each data owner is responsible for calculating the $Merkle_{root}$ by dividing the file into multiple shards, and this information is stored on the blockchain. This function uses a $SHA256$ hashing algorithm to generate a hash and root hash. The overall flow of the data outsourcing process is presented in Fig. 4.3, and the explanation is given below:

i. Data owner requests blockchain to outsource data on cloud storage.

$$DO \rightarrow Blockchain: Upload_{request}()$$

ii. In response, blockchain requests Metadata details to be stored on network.

$$Blockchain \rightarrow DO: Request_{metadetails}()$$

iii. Data owner generates data Meta information using a $SHA256$ hashing algorithm. First, data is divided into multiple shards, and hashing algorithm is used to generate a hash of each shard and calculate the $Merkle_{root}$. Data owner also stores mapping details of user $User_{id}$ and data F_{id} in the blockchain structure and cloud. $DO: F_{ij} = \{f_1, f_2, . . , f_n\}$; $F_{id} \leftarrow SHA256(F_{ij})$;

$$Hash_i \leftarrow SHA256\{f_1, f_2, . . , f_n\};\ Merkle_{root}(Hash_i)$$

iv. Data owner sends generated Meta details to blockchain for storage.

$$DO \rightarrow Blockchain: Store\ (Merkle_{root},\ F_{id})$$

v. After storing Meta details, the data owner generates ciphertext using a shared secret key-based AES encryption algorithm.

$$DO: CT \leftarrow Encrypt(GP, F, SK_A)$$

vi. Data owner sends generated ciphertext with user and file mapped information to blockchain network for cloud storage.

$$DO \rightarrow Blockchain: Outsource_{data}(F_{id},\ User_{id},\ CT)$$

vii. Blockchain sends the data to the cloud for storage.

$$Blockchain \rightarrow Cloud: Outsource_{data}(F_{id},\ User_{id}, CT)$$

viii. Cloud stores and optimizes the resource using Honeybee optimization algorithm to minimize execution and response time as explain in Algorithm 4.5.

$$Cloud: Optimize_{resources}\ (S)$$

ix. On successful storage of data cloud return a true value to the blockchain system.

$$Cloud \rightarrow Blockchain: Return\ True$$

x. Blockchain network returns transaction ID to the data owner.

$$Blockchain \rightarrow DO: Return\ TransID$$

Algorithm 4.2 represents the steps of the data outsourcing process. Line 2-4 depicts the Meta details generation steps where lines 7 and 8 denote the encryption and outsourcing process. Line 9 represents calling the optimization process used by the cloud storage to minimize the transaction process time. The time complexity of the Algorithm 4.2 is $O(n)$.

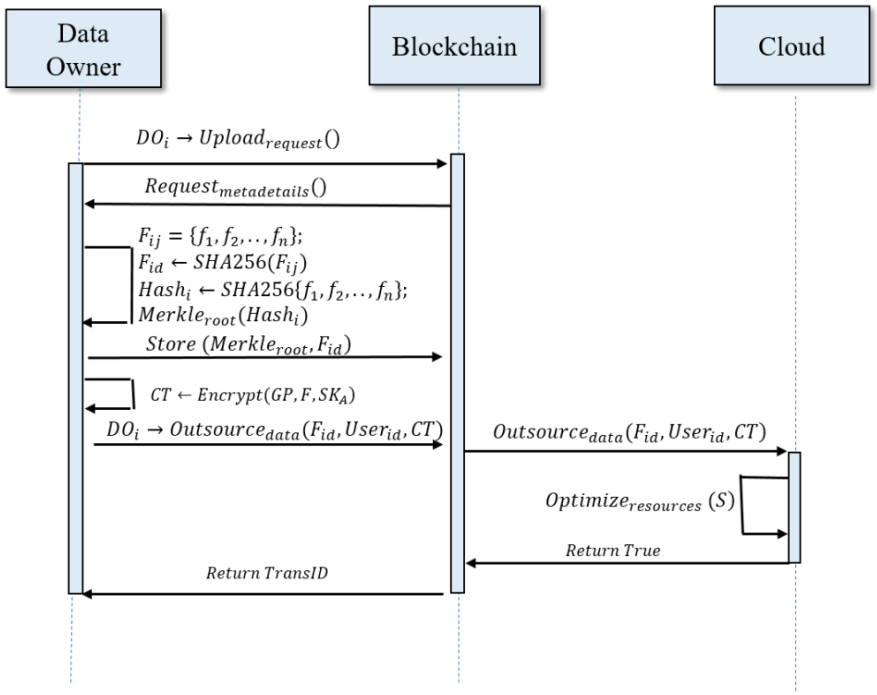


Figure 4.3: Data Outsource Workflow

Algorithm 4.2: Data Outsourcing

$Upload_{request}()$

Input: Data Owner Set $DO = \{DO_1, DO_2, DO_3, \ldots., DO_i\};\ 1 \leq i \leq n,$

File Set $F = \{F_{11}, F_{12}, F_{13}, \ldots, F_{ij}\}$ *where* $i \epsilon$ *DO number and* $j \epsilon$ *User member*

Output: Hash File, Merkle Root $Merkle_{root}$, Ciphertext CT

1. $for\ Each\ File\ in\ F\ do$
2. DO_i Divide File into Multiple shards, $F_i = \{f_1, f_2, \ldots., f_n\}$
3. Calculate File Hash, $HashFile_i \leftarrow SHA256(F_i)$
4. Generate and Store Merkle Root, $Merkle_{Root}(HashFile_i)$
5. $end\ for$
6. $for\ Each\ File\ in\ F\ do$
7. Encrypt File Data F, $CT \leftarrow Encrypt(GP, F, SK_A)$
8. Outsource Data, $Outsource_{Data}(CT)$
9. Optimize Resources, $Optimize_{resources}()$
10. $end\ for$

Access Control Function: This function allows the data owner to check the access control policy before decrypting the requested data. Users are permitted to access the data stored on the cloud, and the data owner first checks whether users can access the data according to the access policy stored on the blockchain. If the users have rights means user shared attributes are a subset of the stored attribute list, then the requested file is converted to plaintext following the decryption process; otherwise, it will discard the request. The access control overall flow is represented in Fig. 4.4.

i. User sends an access request to the data owner.

$$User \rightarrow DO: Access_{request}(F_{id}, User_{id})$$

ii. Data owner retrieves user access policy from the blockchain network.

$$DO \rightarrow Blockchain: Retrieve_{details}(User_{id})$$

iii. Blockchain sends requested details to the data owner.

$$Blockchain \leftarrow DO: u_j(i_1, i_2, i_3, \ldots, i_k\}; A_j$$

iv. Data owner checks if requested user attributes are a subset of the stored list, means it satisfies the access policy and access is allowed.

$$DO: if \{i_1, i_2, \ldots, i_k\} \subset A_j$$

v. After access policy verification, the data owner requests ciphertext to the blockchain system.

$$DO \rightarrow Blockchain: Access(F_{id}, User_{id})$$

vi. Blockchain sends ciphertext fetch request to cloud storage.

$$Blockchain \rightarrow Cloud: Fetch(F_{id,} User_{id})$$

vii. In response, cloud storage returns the requested ciphertext to the user.

$$Cloud \rightarrow Blockchain \rightarrow DO \rightarrow User: Return\ (CT)$$

viii. User utilizes a secret key to decrypt the received ciphertext using an AES algorithm.

$$User: Decrypt(GP, CT, SK_A)$$

Algorithm 4.3 represents the access control process. Line 2-3 represents the access policy checking process by the data owner. Line 4 denotes the decryption of the retrieved ciphertext. Algorithm 4.3 requires a constant time complexity for executing the simple statements.

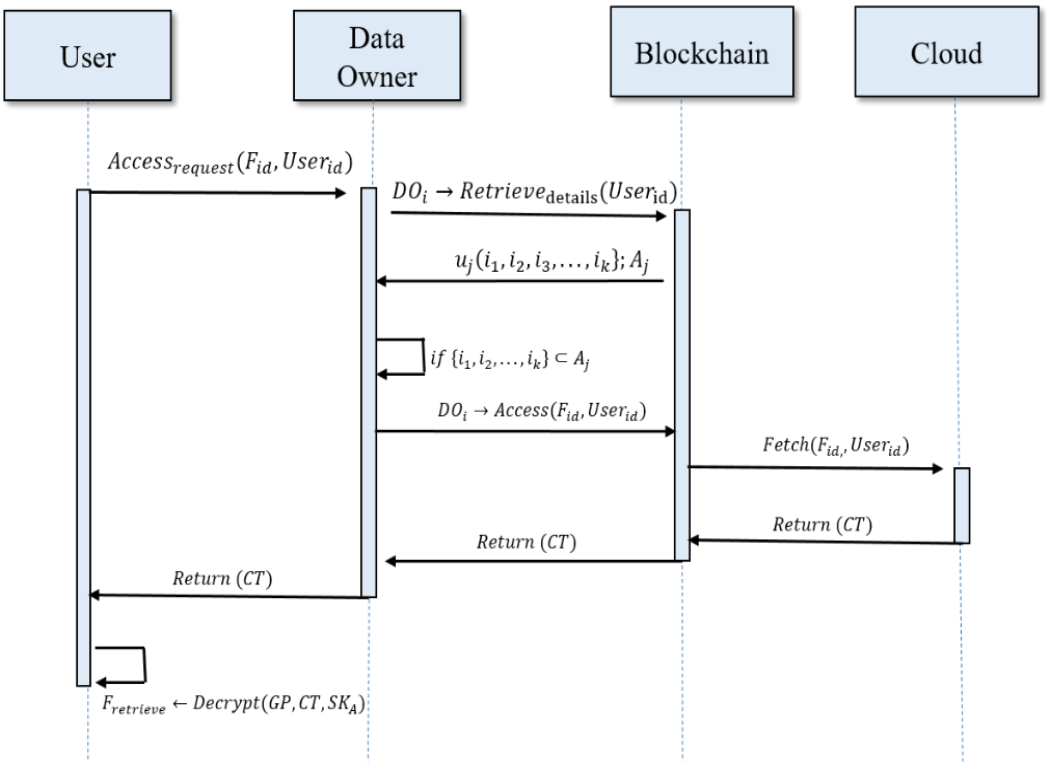


Figure 4.4: Access Control Workflow

Algorithm 4.3: Access Control

$Access_{request}(F_{id}, User_{id})$

Input: Data Owner Set, $DO = \{DO_1, DO_2, DO_3, \ldots, DO_i\};\ 1 \leq i \leq n,$

User Set, $u = \{u_1, u_2, u_3, \ldots, u_j\}\ where\ DO_i \in \{u_1, u_2, u_3, \ldots, u_j\},$

Attribute List, $Attribute_{list}(A_1, A_2, \ldots, A_n)$, Ciphertext CT, File ID F_{id}

Output: File, $F_{retrieve}$

1. User Send Access Request, $u_i \leftarrow Access_{request}(F_{id,} User_{id})$
2. DO Retrieve u_i Attribute Details $attribute_{detials}\{i_1, i_2, i_3, \ldots, i_k\} \in User_{id}$
3. $if\ \{i_1, i_2, \ldots, i_k\} \not\subset A, Access\ fails$
4. $if\ \{i_1, i_2, \ldots, i_k\} \subset A\ then\ F_{retrieve} \leftarrow Decrypt(GP, CT, SK_A)$
5. $u_i \leftarrow access(F_{retrieve})$

Integrity Checking Function: This function ensures the integrity of stored cloud data. Users can request the integrity check for the file from the data owner. Data owner accesses the stored user and file details from the blockchain and follows the verification process. The data

owner retrieves the stored Merkle root of the requested file, recalculates the Merkle root of the file, and performs the match. If the results are matched, then the integrity is verified; otherwise, it returns false. The following steps are involved in the integrity verification process, as shown in Fig. 4.5.

i. User sends integrity checking request to the data owner.

$$User \rightarrow DO: CheckIntegrity_{request}\ (F_{id})$$

ii. Data owner retrieves requested file details from the blockchain structure like Merkle root.

$$DO \rightarrow Blockchain: Retrieve_{details}(F_{id})$$

iii. Blockchain returns requested information.

$$Blockchain \rightarrow DO: Merkle_{root}(F_{id})$$

iv. For the verification process, the data owner, requests the ciphertext from the blockchain, which requests from the cloud storage.

$$DO \rightarrow Blockchain \rightarrow Cloud: Access(F_{id})$$

v. In response, the cloud returns the requested ciphertext.

$$Cloud \rightarrow Blockchain \rightarrow DO: Return(CT)$$

vi. Data owner performs the decryption process using a secret key and recalculates the Merkle root of the fetched data.

$$DO: Decrypt(GP, CT, SK_A)$$

vii. Data owner matches the generated Merkle root with stored Merkle root; if both match, return true to the user; otherwise return false.

$$DO: Match\ (Merkel_{retrieve},\ Merkel_{root})$$

Algorithm 4.4 presents the functionality of the integrity verification process. Line 2 denotes the retrieval of stored Merkle root corresponding to file ID. Line 3 and 4 represent the decryption of retrieved ciphertext and recalculation of Merkle root. Line 6 performs the matching process of both stored and generated Merkle roots; if both are matched, then integrity is verified; otherwise, not. Algorithm 4.4 requires constant amount of time complexity for executing the simple statements.

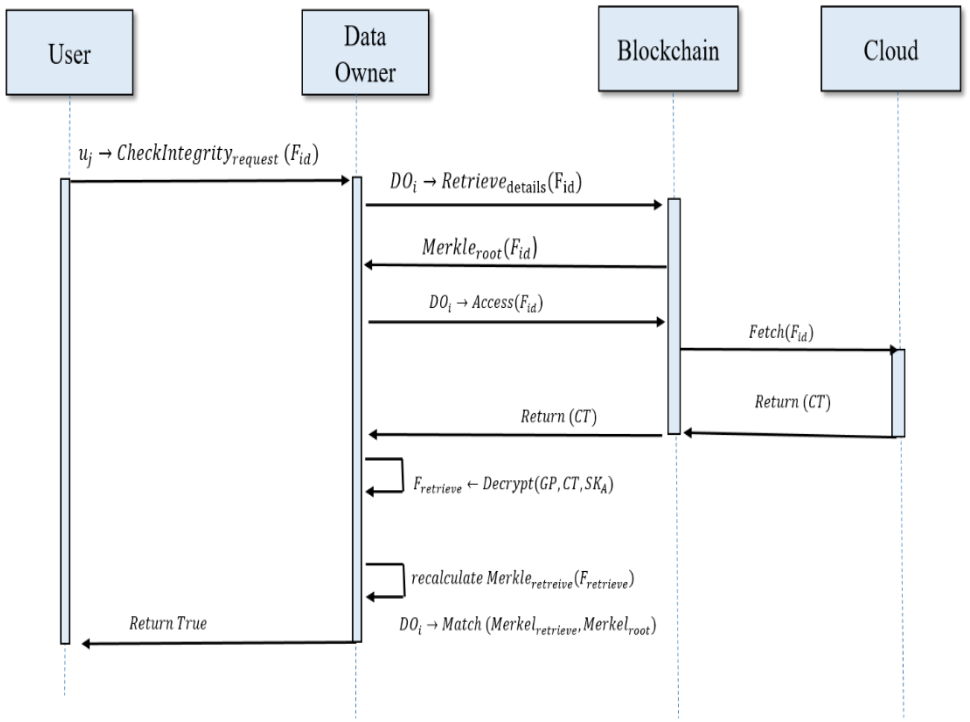


Figure 4.5: Integrity Checking Process

Algorithm 4.4: Integrity Checking

$CheckIntegrity_{request}(F_{id})$

Input: Data Owner, $DO = \{DO_1, DO_2, DO_3, \ldots, DO_i\};\ 1 \leq i \leq n,$

User Set, $u = \{u_1, u_2, u_3, \ldots, u_j\}\ where\ DO_i \in \{u_1, u_2, u_3, \ldots, u_j\},$

Merkle Root, $Merkle_{root}$

Output: $True/False$

1. User Request Integrity Checking, $u_i \leftarrow CheckIntegrity_{request}(F_{id})$
2. DO Retrieve Merkle Root, $Merkle_{root}(F_{id})$
3. DO Decrypt Ciphertext, $F_{retrieve} \leftarrow Decrypt(GP, CT, SK_A)$
4. DO Recalculate Merkle Root, $Merkle_{retrieve} \leftarrow Merkle_{root}(F_{retrieve})$
5. $Match_{result} \leftarrow Verify\ (Merkel_{retrieve}, Merkel_{root})$
6. $if(Match_{result} = true)\ then\ integrity\ verified$
7. $else\ return\ false$

Optimization Function--This function deals with the cloud servers' optimization process to effectively utilize the resources and minimize the response time. Multiple users or multiple

authorities have access to the cloud data in various forms such as data uploading, data downloading, data request, returned results, and so forth. Here, a Honeybee optimization model is deployed to optimize the transactional data process and efficiently reduce time. This algorithm aims to leverage the memory allocation of each server (virtual machine). Therefore, a fair bandwidth is allocated for each server, which helps to prevent task overloading.

Algorithm 4.5: Optimization

$Optimize_{resources}(S)$

Input: Number of Servers S

Output: Resources Optimized

1. Initialize Server Number, $Server_{number} \leftarrow S$
2. Split Servers into Multiple Blocks (Virtual Machines), $S = \{B_1, B_2 \dots \dots B_n\}$
3. Initialize Best Block Size, $Best\ block\ size \leftarrow B_{best.block}$
4. Initialize Average Block Size, $Average\ block\ size \leftarrow B_{avg.block}$
5. Initialize $NonGroup_{foragerbees}$ to Select from $B_{avg.block}$ as B_{NGFA}
6. Declare $Set_{neighbourservers}$ as N_{server}
7. Set $Maximum_{iterationnumber}$ as Max_{iter}
8. Set $Error_{rate}$ as err
9. Estimate $Fitness_{value}\ (S)$
10. Rearrange, $Rearrange\ (Fitness_{value}(S))$
11. $While\ current\ iteration\ i\ \leq\ Max_{iter}$
12. Perform $Neighborhood_{search}(Group_{foragerbees}, NonGroup_{foragerbees})$
13. Select $Best_{foragerbees}$ as $average\ block\ size$ as B_a
14. Re-estimate $Fitness_{value}$ for each $Average_{datablock}$
15. Sort Results of $Average_{datablock}$
16. Reallocate Remaining $forager\ bees$ to other Locations
17. Evaluate $Fitness_{value}$ for each $Average_{datablock}$
18. Arrange $bees$ from $local$ and $global$ search
19. Process Continue until Max_{iter}

Initially, Algorithm 4.5 selects a cloud environment with S servers, and each server is divided into n blocks (virtual machines). Thus, the time complexity of the algorithm is $O(n)$. The designed algorithm identifies the blocks/virtual machines in the overloaded server and finds the less overloaded blocks, evenly distributing the load and minimizing the response time. The task can be taken as a bee, and it is looking for a less loaded block (food source) when it finds the suitable block assignment of the task to block occurs. The next task also tries to assign the same block, which continues until the capacity on the block reaches the threshold value. Once the threshold value is obtained, the process finds the less resource utilized block, and the task is redirected to that block.

The algorithm starts by randomly sending the S scout bees to selecting sites (less or highly overloaded). The fitness value is evaluated for each site and sorted from highest to lowest. The local search finds the best location or sites that are m fitting locations. The m best sites are categorized into two groups; $\mathrm{Group}_{\mathrm{foragerbees}}$ and $\mathrm{NonGroup}_{\mathrm{foragerbees}}$. The number of forager bees is set as $\mathrm{B}_{\mathrm{GFA}}$, and the number of not forager bees as $\mathrm{B}_{\mathrm{NGFA}}$. The local search process starts with recruiting the forager bees in the best sites neighborhood. The global search process is a random search process in the non-best sites. Finally, the overall locations are sorted according to their fitness value, and the process continues until the global optimum is found [84].

4.3 Example Scenario

Consider an organization, a university (ABC University), that wants to manage the organization's data on the cloud in a distributed manner without involving any third party. First, it would create a blockchain network to provide a registration process to its employees. ABC university heads of departments like CSE, ME, EE, ECE, CE, etc., act as data owners represented as $\{DO_1, DO_2, DO_3, \ldots, DO_n\}$ in the proposed blockchain network. Faculty working in the department represent as users $\{u_1, u_2, u_3, \ldots, u_n\}$ who belong to a particular department and work under the head or data owner. Therefore, the proposed architecture allows data owners to register in the network and then be responsible for registering their department faculty/users using the proposed system's key generation algorithm $Key_{generation}()$. Heads/data owners maintain the faculty and department file details. Heads and employee details are saved in the blockchain network. Head also maintains the access policy for their employees or users and provides access according to attribute details $attribute_{detials}\{i_1, i_2, i_3, \ldots, i_k\} \epsilon\ User_{id}$. The access control $Access_{request}()$ and

integrity checking $CheckIntegrity_{request}()$ algorithms provide the privacy-preserving environment and ensure that users can access the concerned department data via respective data owners. Therefore, there are many departmental heads or data owners in the proposed architecture, and each department contains multiple users.

4.4 Security Analysis

This section analyzes the security features of the proposed architecture to address the security threats. The proposed architecture deploys associated mitigation functionality to ensure security and privacy features to the cloud users. The following security solutions are provided by the proposed architecture to tackle the security threats:

Authentication: Users of the proposed architecture follow the registration process to obtain the public keys. Public keys are treated as the users' addresses for the users' authentication to execute the proposed architecture's basic functionality. The blockchain network verifies users before allowing access to the services.

Privacy Protection: The proposed architecture ensures the confidentiality of the data stored by the data owner. It ensures that the unauthorized user will not access the data owner's data. Also, it utilizes encryption techniques to transmit the data among the data owner, user, blockchain, and cloud—the data owner stores the encrypted user and data information in the blockchain network. Therefore, unauthorized users are not allowed to access the data stored on the network. The blockchain structure ensures all operations to be non-destructive and non-repudiation. The access policy and user details are securely stored in the blockchain network, ensuring data confidentiality and privacy.

Key Security: The proposed architecture deploys the traditional CP-ABE algorithm in the blockchain network. It registers multiple data owners to generate user keys instead of a single authority or key management authority, thereby reducing the burden and providing a reliable environment. The respected data owner generates the data user key. Considering the risks involved while sharing the keys through the transmission medium, both the data owner and the user use the key exchange protocol Diffie-Hellman algorithm to share the key. Therefore, the proposed architecture guarantees the security of the keys.

Access Control: The proposed architecture implements a fine-grained access control method and removes a single authority, ensuring a more dispersed access mechanism. It utilizes an attribute-based access control scheme to authorize user access. The users provide attribute

details during the registration process to the data owner. The data owner stores the attribute list to the blockchain network and describes the attribute policy using the subset operator. If the shared user's attributes are a subset of the stored attribute list, access is allowed to the requested user. After access is allowed, users follow the decryption process to obtain the plaintext using a secret key.

Integrity Protection: The user can request the integrity validation of the data from the data owner. The proposed architecture utilizes the Merkle root concept to check the integrity of the data. The data owner maintains the Meta details of the data during the blockchain network's uploading process. Therefore, the data owner obtains the stored Merkle root, recalculates the Merkle root of the requested data, and performs the verification. If both hash matches, then the data's integrity is maintained; otherwise, it returns false to the user.

4.5 Result Analysis

This section presents the details of the proposed architecture experimental setup, evaluates storage and system performance, and compares the proposed work with the existing techniques.

4.5.1 Experimental Setup

We implemented the proposed architecture in the Java programming language. The experiments are conducted on the Window 10 operating system, with Intel® Core ™ i7CPU, 2.5GHz, and 8GB RAM. We used Netbeans 7.0 IDE to implement the proposed architecture with JDK 1.7. The external auxiliary Java Pairing-based Cryptography is used to implement bilinear pairing-based cryptography in the proposed architecture. For simulating the cloud storage environment, the CloudSim-3.0.3 framework is used. CloudSim provides the cloud computing component system and behavioral modeling. Simulation of cloud environment offers useful insights to explore such dynamic, distributed, and scalable environments. The jar folder cloudsim-3.0.3.jar is used for integrating a java-based blockchain network with the simulated cloud environment. There are many more mature blockchain networks available, like Ethereum and Hyperledger. However, these blockchain networks are not directly deployed in the proposed architecture because the block header is more complicated than the traditional blockchain network. In contrast, we define a minimized block header while maintaining the Meta details. Thus, we have designed and implemented our minimal block

structure to combine multi-server cloud storage to ensure an easy, secure and reliable environment.

4.5.2 Storage Performance

CloudSim is used to simulate and model a cloud computing environment. The computing resources and virtual nodes are simulated to evaluate the performance of the optimization algorithm. Initially, we created a CloudSim environment with 20 data centers, 50 virtual machines, and 100-1000 transactions (tasks). Fig. 4.6 presents an average response time of the algorithm while executing each transaction ranges from 2000 to 20000 bytes by varying the number of transactions. The average response time of tasks expresses the amount of time between submitting a request and the first response produced by a transaction in seconds. It is observed that the average response time starts at about 6 s in all situations. Still, this value increases slightly to reach 18 s when the task's executable instruction length equals 2000 bytes. However, it is highly increased to reach 100 s when the executable instruction length is equal to 20000 bytes. This is due to an increase in the number of tasks and how the instruction length affects the system load. Then, the response time is also increased.

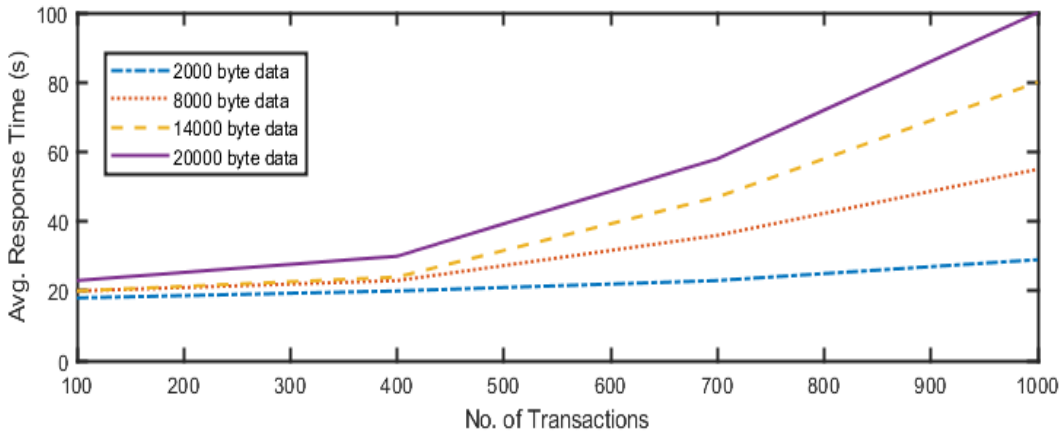


Figure 4.6: Average Response Time

Fig. 4.7 presents an average execution time of the proposed work while executing each transaction ranges from 2000 to 20000 bytes at different transactions. The average execution time of transactions denotes the amount of time between the start and the transaction's finish time in seconds. It is observed that the average execution time starts at about 8 s in all situations. This value increases to 15 s when the task's executable instruction length equals 2000 bytes. However, it is highly increased to reach 95 s when the executable instruction length equals 20000 bytes.

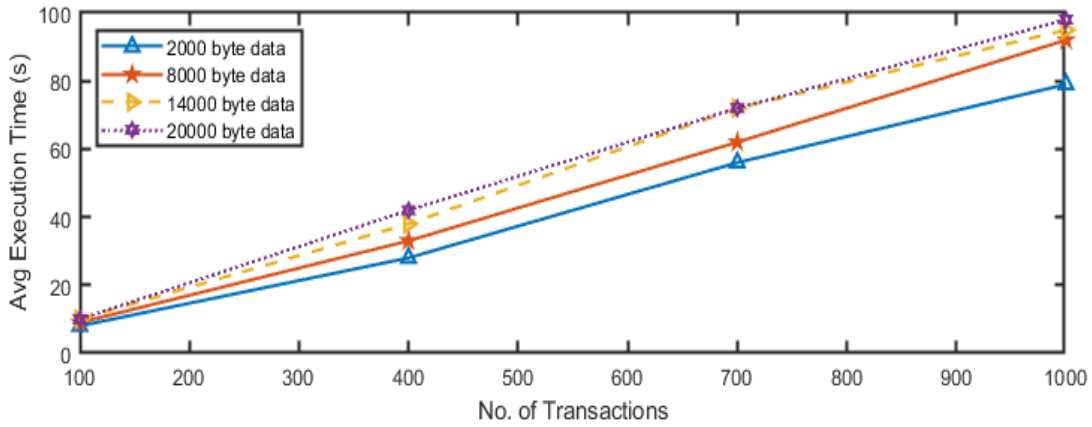


Figure 4.7: Average Execution Time

The proposed system's resource usage is calculated based on the use of computational resources. The computing workload is directly proportional to resource consumption, which rises as the number of transactions rises. Equation 4.2 is used to measure the average resource consumption. As shown in Fig. 4.8, the proposed scheme maximizes resource efficiency by making resources as busy as possible compared to the proposed architecture without optimization implementation. Resource usage is lower when there are fewer transactions; but, as the number of transactions grows, resource utilization rises. This implies that for the larger task sets, the resource would be fully used. Fig. 4.8 shows that the minimum device utilization ranges from 74% to 80% for small task sets but reaches 100% when the computational task demands increase in the proposed solution. In contrast, the proposed architecture device utilization ranges from 70% to 84%, without optimization algorithm deployment. Thus, the proposed architecture efficiency is increased in terms of resource utilization rate with the implementation of optimization algorithm as compared to without optimization.

$$Resource_{utilization} = \sum_{i=1}^{n} \frac{Transaction\ processing\ time\ by\ a\ resource}{active\ time} \qquad (4.2)$$

Where $Resource_{utilization}$ denotes the total number of computing resources engaged in processing task sets, and n is the combined utilization of all computational resources expended on processing tasks workload.

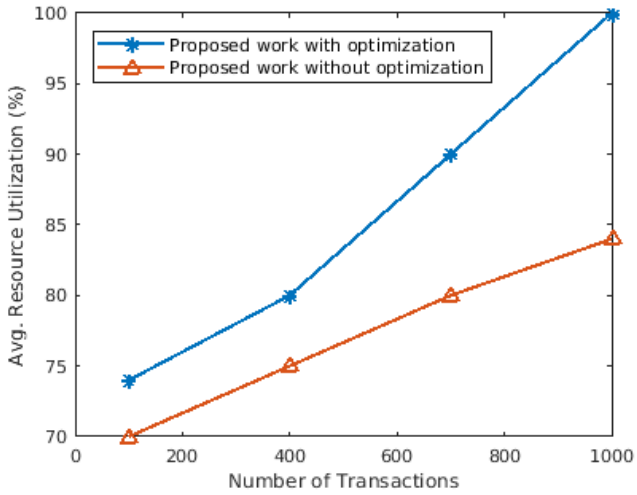


Figure 4.8: Average Resource Utilization

4.5.3 System Performance

This section analyses the proposed architecture based on computation time, throughput (total packets received/time frame), and delay parameters. By utilizing JMeter, we conducted the evaluation process to measure the performance parameters of the proposed architecture. The number of transactions is in the range of 100-1000 who execute the proposed architecture's different operations. The proposed architecture supports three types of transactions, i.e., data, access, and validation transactions. The data transactions include the functions required to upload the data on cloud storage. The access transactions include the operations necessary to access the uploaded data from the cloud storage. The validation transaction consists of the integrity checking function of the proposed architecture. During the experiment, we executed a series of transactions and evaluated the system performance. A maximum of 1000 transactions are executed on the proposed application, and, in the end, the throughput computation time and delay parameters are assessed. In JMeter, the Data/time (KB/sec) units indicate the throughput. The proposed application throughput refers to the amount of data transferred from one location to another in a unit amount of time. The evaluation is monitored with a breakpoint after 100 transactions, and the total throughput of the proposed network is calculated for different transactions, as shown in Fig. 4.9.

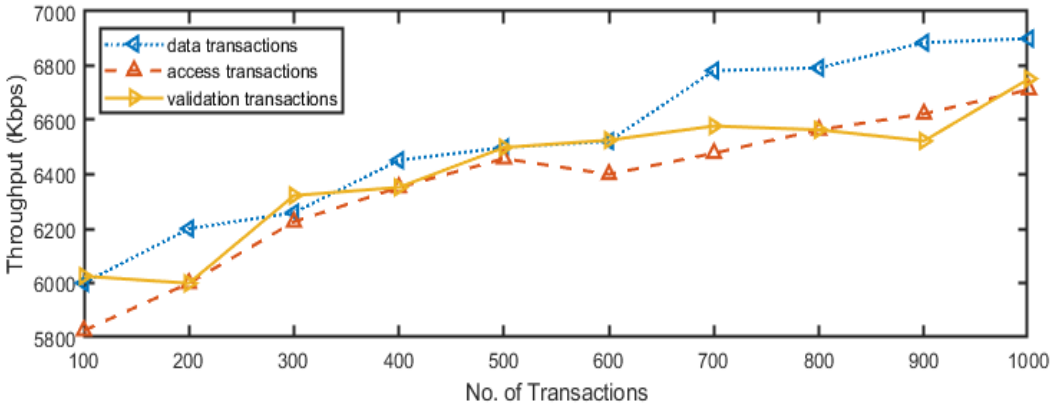


Figure 4.9: Throughput of the Proposed Work

The computation time is the average time taken to execute the series of transactions. As shown in Fig. 4.10, the computation time increases with the number of transactions. The user initiates a thousand transactions in the proposed application to analyze the average computation time interval of a hundred transactions. The average computation is calculated separately for each type of transaction, as shown in Fig. 4.10. Delay denotes the time required by the system to send the transaction request to the other nodes. The average delay represents the delay required to execute the 1000 transactions in the proposed architecture. We have evaluated the proposed application average delay by using the Jmeter tool and measuring it in a second. As shown in Fig. 4.11, the proposed application recorded the highest delay in validation transactions, i.e., 4.5 seconds.

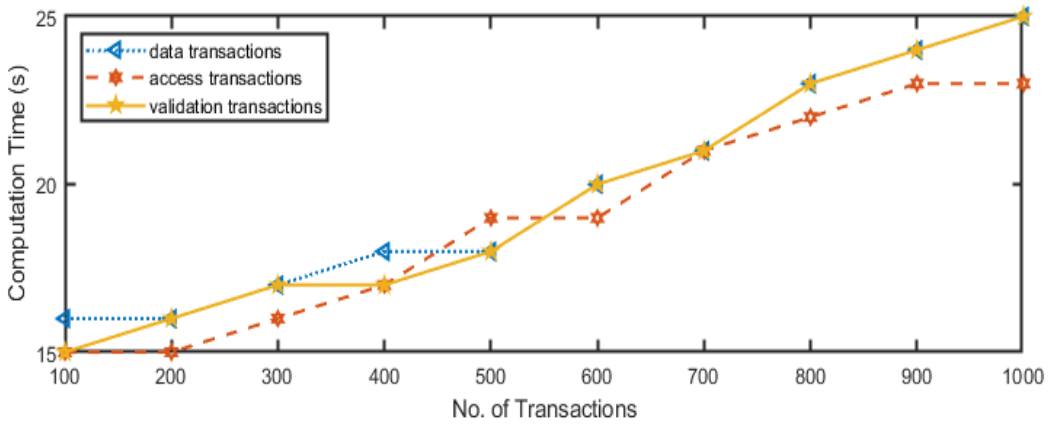


Figure 4.10: Computation Time of the Proposed Work

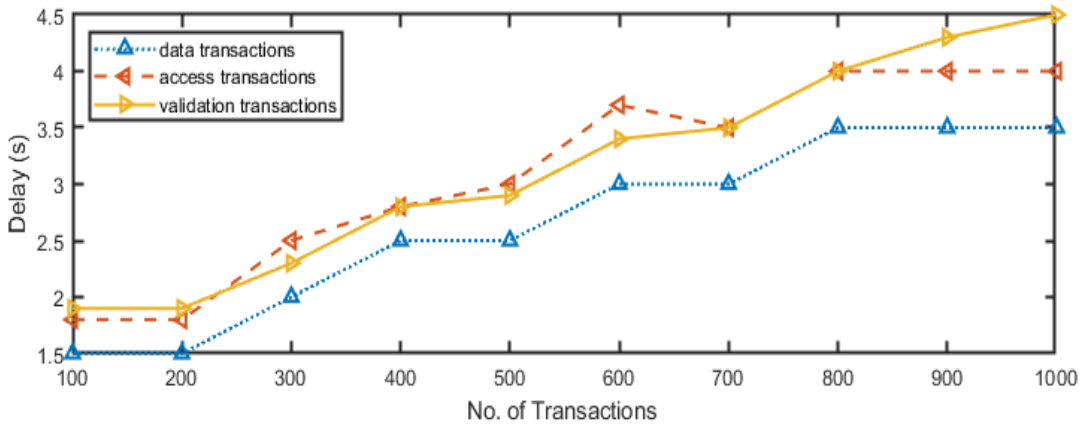


Figure 4.11: Delay of the Proposed Work

4.5.4 Comparative Analysis

This section compares the proposed architecture with state-of-the-art techniques such as block secure [73], secure access control [74], and deduplication with blockchain [76]. The different aspects of the proposed architecture are evaluated and compared with the existing work. The network performance of the proposed architecture is analyzed by comparing it with the block secure [73]. The proposed architecture access control algorithm efficiency is evaluated and compared with the existing work [74]. Also, the proposed architecture storage performance is measured and compared with the work deduplication with blockchain [76]. For comparative analysis, we have selected three different literature pieces to measure the various aspects of the proposed architecture by using different measuring parameters. First, the proposed architecture is compared with the literature [73] based on transmission time, considering two different architectures with or without optimization algorithm implementation. Next, the proposed architecture performance is analyzed by varying the number of user attributes to evaluate the run time of the transactions and compared with the secure access control [74]. Last, the performance of the proposed architecture is measured based on encoding and decoding time and compared with blockchain-based deduplication architecture [76]. This comparison shows the significance of the proposed architecture from existing architectures.

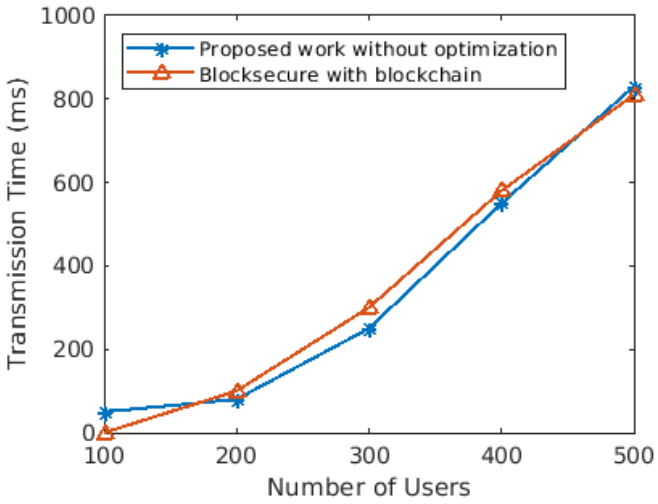


Figure 4.12: Transmission Time Comparison between Proposed Work without Optimization and Block Secure

The block secure architecture utilizes the two architectures for storing the user data on the cloud. One is blockchain architecture with multiple data centers, and another is multiple data centers with genetic optimization [73]. As shown in Fig. 4.12 and 4.13, the proposed architecture compares with both block secure architectures using transmission time as the parameter. The transmission time is defined as the amount of time required for storing and accessing the files on the cloud. The proposed architecture without optimization implementation utilizes the same amount of transmission time required by the block secure with blockchain solution without optimization by varying the number of users as depicted in Fig. 4.12. Thus, the proposed architecture maintains the effectiveness of the existing work.

As shown in Fig. 4.13, the proposed architecture with optimization implementation compares with block secure architecture with genetic implementation. The proposed architecture requires less transmission time for storing and accessing the files from the cloud database than block secure [73]. The block secure architecture requires more time on calculations in genetic algorithm, whereas proposed architecture does not require much analysis. Thus, the proposed architecture performance is better than the block secure architectures in terms of transmission time.

The proposed architecture performance is compared to the secure cloud storage with access control literature using run time as the measuring parameter. The proposed architecture performance is analyzed by measuring the run time of access transactions with varying the number of attributes of the users. As shown in Fig. 4.14, the proposed architecture access

transactions execution requires less time than the secure access control [74]. The execution time of both architectures increases as the number of attributes increases. The formulation of simple access policies in the proposed architecture decreases the run time compared to the literature.

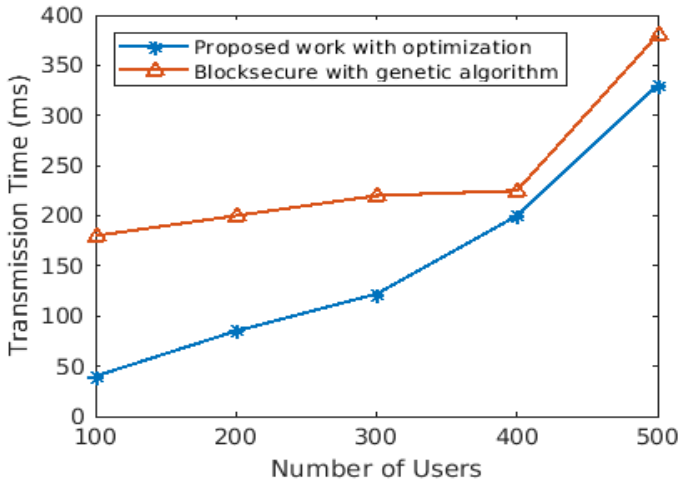


Figure 4.13: Transmission Time Comparison between Proposed Work with Optimization and Block Secure

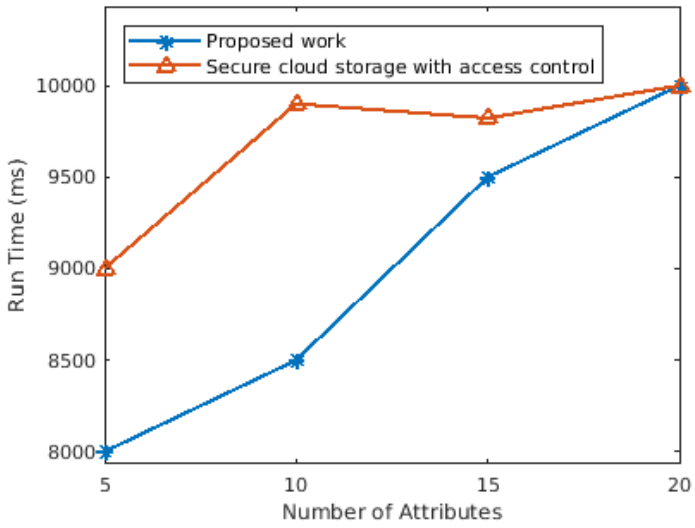


Figure 4.14: Run-Time Comparison of Proposed Architecture with the Secure Access Control

The proposed architecture's encoding and decoding time are compared with the literature [76]. As shown in Fig. 4.15 and 4.16, the proposed architecture requires similar encryption and decryption time compared to blockchain-based deduplication architecture. The decoding time for both architectures is less than the encoding time. As depicted in the figures, the

proposed architecture and literature take the same time for both processes. Therefore, the performance evaluation and comparative analysis show that the proposed architecture maintains the effectiveness of the existing work and provides a better and secure solution.

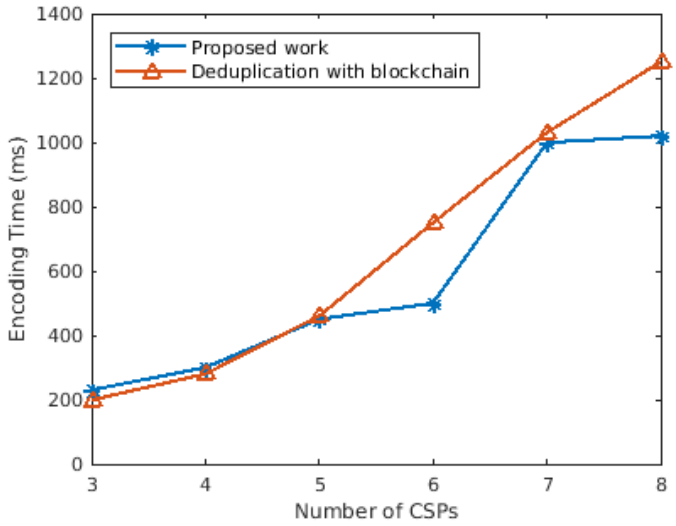


Figure 4.15: Encoding Time Comparison between Proposed Work and Deduplication with Blockchain

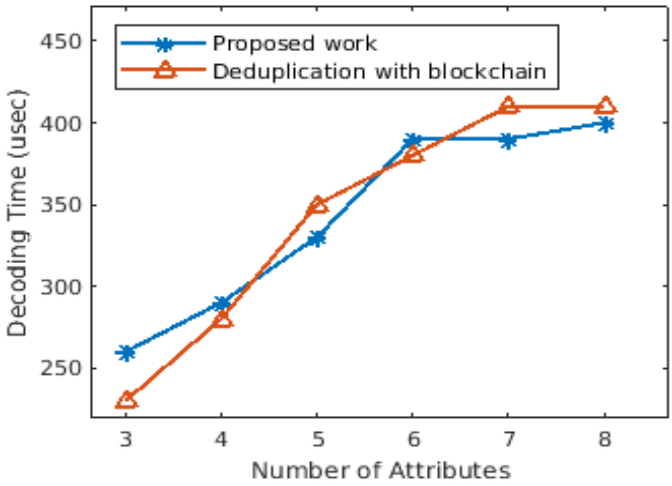


Figure 4.16: Decoding Time Comparison between Proposed Work and Deduplication with Blockchain

4.6 Conclusion

This chapter represents distributed and privacy-preserving blockchain-based cloud storage architecture with access control and integrity checking features. The proposed architecture deploys a CP-ABE approach in blockchain structure to provide key generation processes

using bilinear pairing. The implemented key generation process uses distributed system to prevent complete reliance on a single authority and deliver a more secure environment. It also provides an access control mechanism that ensures cloud data are only accessible by the authenticated user with data owner nodes' help. The proposed blockchain structure maintains the integrity of the cloud data using the Merkle root concept. Furthermore, it uses Honeybee optimization techniques at the cloud storage system to optimize resource utilization and minimize transaction processing time. For future work, we plan to include the user revocation process in the existing architecture to provide reliable, safe, and dynamic removal of the user at any time and manage key management and access control approach accordingly.

CHAPTER 5

ACCESS CONTROL AND USER REVOCATION PROCESS IN CLOUD STORAGE USING BLOCKCHAIN

This chapter introduces the enhanced access control and user revocation features in the proposed blockchain-based cloud storage architecture. The proposed architecture registers data owners and attribute authorities using a key generation algorithm. The data owners and attribute authorities store the public information in the blockchain structure, set access policies, and generate the user's secret key to resolve key escrow problems. The deployed architecture attains the fine-grained access control with the user revocation process. The performance evaluation demonstrates that the proposed scheme provides a more efficient cloud environment.

5.1 Introduction

With the tremendous growth of Information Technology, a massive amount of information is generated every year, and it is difficult to handle this information locally. Therefore, the cloud storage system is considered the most suitable and economical way to resolve this issue [85]. However, the storage of confidential information on the cloud servers creates trust issues. Thus, promising cryptographic solutions are required to protect user privacy, achieve access control, perform user revocation, and encode the data before uploading to the cloud. Traditionally used cryptographic solutions cannot provide flexible information sharing processes and fine-grained access control; therefore, a new encryption technique, Attribute-based Encryption (ABE), was developed by Sahai and Waters [86]. The ABE algorithm utilizes the attribute set to specify access policy and allow decryption of the data according to

the set policy. Since then, many variations of ABE have been proposed by researchers to specify access policy to the ciphertext or decryption key [87-89]. Nevertheless, the traditional ABE algorithms lack efficiency, require more computational cost, and increase linearly with the complexity of access policy. The access control mechanism in the existing cloud system requires one or more completely trusted attributes or central authorities to maintain the access policy. If the central authority is compromised, the whole structure is affected. Also, the user revocation where re signature generation is required remains a major task. Blockchain is a relatively advanced computer technology development where members can immediately capture and share transactions with other members [90]. Several research studies have used the blockchain to achieve attribute-based access control and revocation process in the cloud storage system. For example, in [91], the authors present a blockchain-based data sharing scheme using smart contract and ABE to achieve user revocation process. The proposed architecture provides privilege management using attribute-level revocation during the data sharing process. It involves trusted authority for the key management process and encrypts or decrypt data; thus, the user data's security is at risk. In case of failure of key management center, users cannot access their information, and the entire structure will get affected.

Furthermore, Ning et al. [92] develop a cryptcloud framework to secure cloud storage systems. The designed architecture uses the CP-ABE algorithm to support the white box traceability, provide auditability, and revocation process. It lacks the utilization of blockchain technology to provide a more reliable environment. Similarly, in [93], the authors design a multi-authority-based CP-ABE approach to ensure the revocation process in the Named Data Network (NDN). The suggested method deploys the proxy-based access control technique with forward and backward security. The involvement of proxy servers increases the response time. Also, Fan et al. [94] implement the access control mechanism using proxy servers for smart cities. It also uses the CP-ABE scheme to achieve the security features and user revocation process. The proposed scheme requires more processing overhead due to the usage of proxy servers. Wang et al. [95] design a secure cloud storage system using blockchain technology. The designed framework develops using the Ethereum platform to set a valid access period for cloud users and achieves an access control mechanism. It uses the smart contract functionality to store ciphertext in the blockchain network. Similarly, Saini et al. [96] propose a smart contract-based access control framework using blockchain technology for the healthcare system. The designed system uses the Elliptic Curve Cryptography to encrypt the healthcare data before storing it on the cloud. These works suffer

from many disadvantages, such as the involvement of a semi-trusted party which affects the overall security of the architectures.

To deal with the disadvantages and challenges mentioned above, we designed Java-based blockchain architecture with access control and user revocation process for the cloud storage to ensure data security. The proposed architecture implements CP-ABE to provide a key generation mechanism using bilinear mapping-based cryptography and perform data access control mechanisms. Data owners and attribute authorities manage the key related and user access policy details in a distributed manner by utilizing the blockchain structure and providing a more robust environment. The designed architecture stores publicly accessible information in the blockchain network while also controlling stored cloud data.

5.2 Proposed Work

In this section, the entire proposed scheme has been explained in detail.

5.2.1 System Model

Fig. 5.1 shows the decentralized secure architecture for blockchain-based cloud storage. The proposed system model consists of five core entities:

- **Blockchain:** Blockchain maintains the non-tamper and transparent data transfer between the users of the proposed architecture. The blockchain network ensures access control and revocation functionality using various smart contract functions such as key generation, encryption, re-encryption, decryption, and key update functions.
- **Attribute Authority:** Attribute authority produces the user keys using the CP-ABE algorithm. Attribute authority has the right to generate, distribute, and modify user attribute keys. Attribute authority also manages user’s access rights based on their attributes. Attribute authority employs a re-encryption process to attain fine-grained access control to apply the attribute level's user revocation method.
- **Data Owner:** The data owner outsources the data on cloud storage to effectively manage data distribution. The data owner also defines the attribute-based access policies and uses these policies to encrypt the user files.
- **User:** Users can access the data when their attributes satisfy the ciphertext's access policies. The user can use the decryption algorithm to decrypt the file using the secret

key and obtain the plaintext. If the data owner removes any user, the user cannot access that group data.

- **Cloud Storage:** Cloud storage stores encrypted files uploaded by the data owners. It manages the access of stored data and gives relevant services.

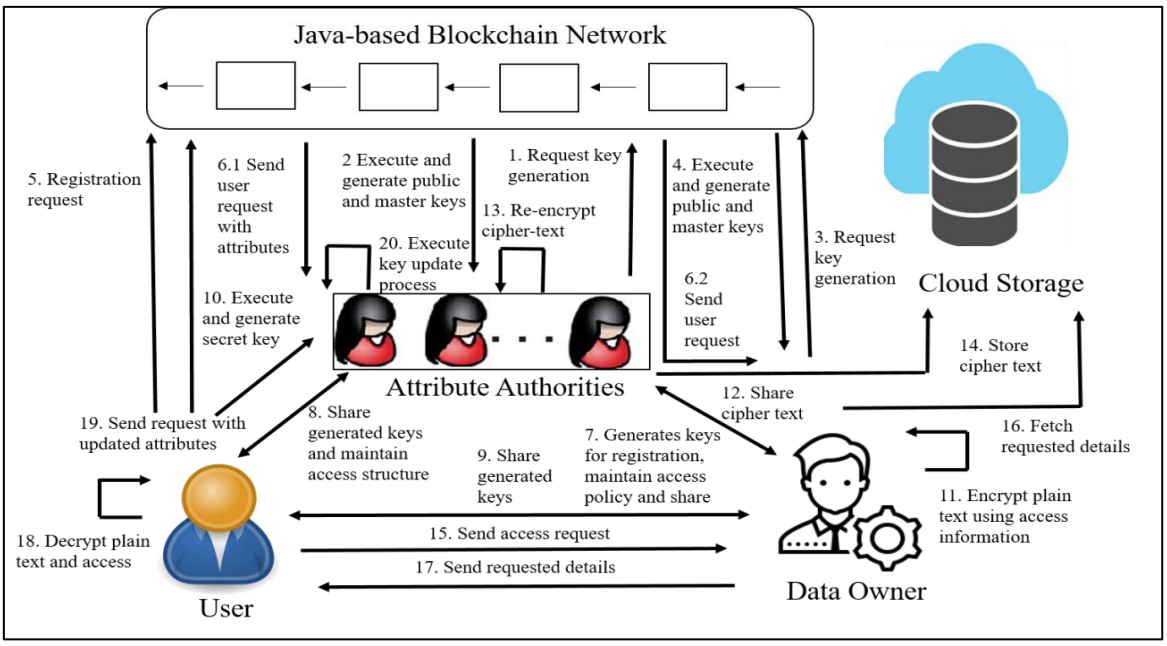


Figure 5.1: System Model of the Proposed Architecture

The proposed architecture workflow, as shown in Fig. 5.1 explanation, is as follows:

1. Attribute authorities and data owners send key generation requests to generate global parameters and public and master keys to register in the proposed architecture.
2. The blockchain network executes the key generation function and generates a public key and master key for the data owner and attribute authority.
3. The user sends a registration request to the blockchain network, which sends the user details to the data owner and attribute authority with attribute list like user ID, department, email ID, contact number, address, DOB, etc.
4. Using the CP-ABE algorithm, the data owner and attribute authority save user details and generate an access policy corresponding to the user's attribute list. The generated keys of the data owner and attribute authority are shared with the user. The user executes the key generation algorithm to generate a secret key using data owner and attribute authority generated keys.
5. The data owner encrypts the plaintext using the access structure to generate the ciphertext. The generated ciphertext shares with the attribute authority for the re-

encryption process to provide user revocation and generates header information so that revoked users cannot access the encrypted ciphertext.

6. After the re-encryption process, the ciphertext with header information is outsourced to the cloud server.
7. The user sends an access request to the data owner. In response, the data owner fetches the requested details from the cloud server and shares them with the user.
8. Then, the user uses the decryption process to decrypt the plaintext. If the user is authenticated, he can decrypt the data as the ciphertext associated with the header information that only allows active users to access it.
9. The user may send the request to attribute authority to add, remove or modify the attributes.
10. The attribute authority executes the key update function to regenerate the keys for the modified user list and updates the group access policy.

The proposed architecture consists of the following smart contract algorithms:

$Setup(, \mathbb{G}, g, e, \mathbb{G}_{T,} H, H_1) \rightarrow GP$: This algorithm computes the public parameter GP in the setup phase.

$DOkeygenerate(GP) \rightarrow (PK_{do}, MK_{do})$: This algorithm generates public key PK_{do} and master key MK_{do} for data owners.

$AAkeygenerate(GP) \rightarrow (PK_{aa}, MK_{aa})$: This algorithm generates public key PK_{aa} and master key MK_{aa} for attribute authorities.

$DOkeycomp(MK_{do}, U_t)\ and\ AAkeycomp(MK_{aa}, U_t, S_i, S) \rightarrow SK_{do,U_t}\ and\ SK_{aa,U_t}$: These two algorithms are executed by the user U_t. First, data owners take master key MK_{do} and user ID U_t as input and output unique private key SK_{do,U_t} for the user. Then, attribute authorities take a master key MK_{aa}, user ID U_t, confidential data T_i and attributes set T described as input and outputs $SK_{aa,\,U_t}$ user key. Finally, the user gets the final secret key SK_{U_t} by combining these two key generation algorithms.

$Encrypt(F, PK_{do}, PK_{aa}, A_t) \rightarrow CT$: This algorithm takes file F, public keys PK_{do}, PK_{aa}, and access structure A_t, as inputs and outputs ciphertext CT.

$ReEncrypt(CT,\ A_G, PK^*_{aa}) \rightarrow CT'$: This algorithm takes the CP-ABE generated ciphertext CT, attribute group details A_G, attribute group public key PK^*_{aa} and outputs

ciphertext CT'. The only active user who satisfies the updated policy can access the updated ciphertext.

$Decrypt(CT', SK_{U_t}, K) \rightarrow F$: This algorithm is executed by the active users to decrypt the ciphertext and obtains plaintext F.

All the notations and their descriptions used in the proposed scheme are given in Table 5.1.

Table 5.1: Notation Table

Notations	**Description**
GP	Global parameters
PK_{do}	Public key of the data owner
MK_{do}	Master key of the data owner
PK_{aa}	Public key of the attribute authority
PK_{aa}^{*}	Public key of the attribute group
MK_{aa}	Master key of the attribute authority
SK_{do,U_t}	Data owner secret key for the user
SK_{aa,U_t}	Attribute authority secret key for the user
U_t	User ID
SK_{U_t}	Secret key of the user
SK_{aa,U_t}^{*}	User secret key for attribute group
CT	Ciphertext
CT'	Re-encrypted ciphertext
$Header$	Ciphertext header information for the user access
K	Security parameter denotes group size
F	File
T	Attribute set
U_t	User ID
A_t	Access structure
A_G	Attribute group
DO	Data owner
AA	Attribute authority

5.2.2 Smart Contract Functions

The proposed architecture deploys the various smart contract functions using a CP-ABE algorithm to provide various services to the user such as key generation process, data outsource service, access control, and revocation process using re-encryption process. The proposed architecture involves data owner and attribute authority entities to provide the essential services to the user using bilinear mapping. The blockchain network selects a bilinear group $\mathbb{G}$ of prime order p and generator g and two random numbers $a, b \in \mathbb{Z}_p$. Let p be a prime number and $\mathbb{G}$, $\mathbb{G}_T$ be multiplicative cyclic groups of order p. A security parameter *K* denotes the group's size. We also use Lagrange coefficients $\Delta_{i,\mathcal{L}}$ for any $i \in \mathbb{Z}_p^*$ and a set, $\mathcal{L}$, of elements in $\mathbb{Z}_p^*$: define $\Delta_{i,\mathcal{L}}(x) = \prod_{j \in \mathcal{L} j \neq j} \frac{x-j}{i-j}$. The hash functions are also designed $H: \{0,1\}^* \rightarrow \mathbb{G}$ to link each attribute with random group element in $\mathbb{G}$ and $H_1: \mathbb{G}_T \rightarrow \mathbb{Z}_p^*$, to model random group element. A map e: $\mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ satisfying the following properties is called a bilinear map or bilinear pairing.

- $e\ (u^a, v^b) = e(u,v)^{ab}, \forall\, u, v \in \mathbb{G}, a, b \in \mathbb{Z}_p$
- If g is a generator of $\mathbb{G}$, then $e(g,g)$ is a generator of $\mathbb{G}_T$
- $e(u,v)$ is efficiently computable for all $u, v \in \mathbb{G}$
- Let $H: \{0,1\}^* \rightarrow \mathbb{G}$ be a hash function that maps attribute to the random element of $\mathbb{G}$.

Key Generation Function: This phase generates global parameters GP at the initial stage. It uses a bilinear group $\mathbb{G}$ of prime order p and generator g based on security parameters. The two hash functions $H: \{0,1\}^* \rightarrow \mathbb{G}$ and $H_1: \mathbb{G}_T \rightarrow \mathbb{Z}_p^*$ are also selected from the hash functions, universal family.

Algorithm 5.1 is executed by the data owners. Line 1 selects a random number from the finite field over prime p, $\mathbb{Z}_p^*$ and use a generator to generate public and master key for the data owner. It outputs two keys pubic key PK_{do} and master key MK_{do}. Both keys are saved in the blockchain network using the SHA256 hashing algorithm. Algorithm 5.1 requires constant amount of time complexity for executing the simple statements.

Algorithm 5.1: Key Generation for Data Owner

$DOkeygen(GP)$

Input: Global Parameter GP, Random Exponent a

Output: Public Key PK_{do} and Master Key MK_{do}

1. Begin
2. Let $a \in \mathbb{Z}_p^*$ //selects a random number
3. $\mathfrak{b} \leftarrow g^a$ // g is a generator
4. $PK_{do} \leftarrow \mathfrak{b}$
5. $MK_{do} \leftarrow a$
6. End

Algorithm 5.2 is executed by the attribute authorities. Line 2 selects the random number β from $\mathbb{Z}_P^*$. Similarly, Line 3 selects the random number c from $\mathbb{Z}_P^*$. After the random number selection, the public key is calculated by applying a mapping function using the selected random number β. The master key is generated using the generator function using β, and the attribute group public key is created using c. It generates public keys PK_{aa}, and PK_{aa}^*, and master key MK_{aa}. Algorithm 5.2 requires a constant time complexity.

Algorithm 5.2: Key Generation for Attribute Authority

$AAkeygen(GP)$

Input: Global Parameter GP, Random Exponent β

Output: Public Keys PK_{aa}, PK_{aa}^* and Master Key MK_{aa}

1. Begin
2. Let $\beta \in \mathbb{Z}_p^*$ //selects a random number
3. Let $c \in \mathbb{Z}_p^*$ //selects a random number
4. $PK_{aa} \leftarrow e(g,g)^{\beta}$
5. $MK_{aa} \leftarrow g^{\beta}$
6. $PK_{aa}^* \leftarrow g^c$
7. End

User Key Generation Function: The attribute authority and the data owner participate in the user key generation process. First, the attribute authority and data owner verify the user ID and check if it already exists. After verifying the user ID, the attribute authority and data owner follow the user key generation process using the secure two Party Computation function (2PC) that allows the two parties to jointly compute the function using their inputs without sharing their inputs with the other party [97]. Thus, the proposed architecture utilizes the 2PC protocol to securely generate the user secret key parameters. Then, the user uses the generated parameters to create the secret key. The attribute authority and data owner execute the key generation algorithm to generate one public key and two secret keys, PK_{do,U_t}, SK_{aa,U_t}, and SK^*_{aa,U_t}. The user utilizes these keys to generate his own key SK_{U_t}using Equation 5.1. The user uses the generated key to decrypt the ciphertext.

Algorithm 5.3 generates the secret key for the user using the secret keys generated by the data owners and attribute authorities. First, the data owner and attribute authority authenticate the user details then uses the 2PC function to calculate the value of x using the data owner's N_t and a parameters and attribute authority's M_t and β parameters. Next, the data owner calculates the value of S using the generator function and shares it with the attribute authority using the 2PC protocol. Similarly, the attribute authority utilizes the S to calculate the P parameter.

Based on calculated parameters, attribute authority generates the secret key SK_{aa,U_t} and generates the group key SK^*_{aa,U_t}using PK^*_{aa} key parameter c by applying a hash function on user ID U_t. The data owner uses the attribute set T to calculate the value of D using a generator with a random number and generates the secret key SK_{do,U_t}.

The algorithms generate the secret key for the user by using data owner and attribute authority generated secret keys $SK_{U_t} = \{SK_{aa,U_t}, SK_{do,U_t}\}$ by using Equation 5.1.

$$SK_{U_t} = (g^{\frac{(\beta+N_t)}{a}},\ (\forall i\ \in T: D_i = g^{M_t}.H(i)^{r_i}, D_i = g^{r_i})) \qquad (5.1)$$

The user's secret key is denoted by SK_{U_t}. The user uses the generated key to execute the decryption algorithms to decrypt the ciphertext and obtains the plaintext that the data owner outsources. Algorithm 5.3 requires a constant time complexity.

Algorithm 5.3: Key Generation for User

$AAkeygen(\,) \leftrightarrow DOkeygen(\,)$

Input: User ID U_t

Output: User Secret Key SK_{U_t}

1. Begin
2. $if\ AA$ and DO authenticate $User\ U_t, then$
3. DO selects random exponent $N_t \in_R \mathbb{Z}_p^*$ //unique and secret to the user
4. $x \leftarrow (\beta + N_t)a \leftarrow 2PC\big(DO(N_t, a), AA(\beta)\big)$ //general and secure 2PC function
5. AA selects random exponent $M_t \in_R \mathbb{Z}_p^*$
6. $DO \leftarrow S = g^{\frac{x}{M_t}} = g^{\frac{(\beta+N_t)a}{M_t}}$ //DO computes and send to AA using 2PC
7. $AA \leftarrow P = S^{\frac{1}{a^2}} = g^{\frac{(\beta+N_t)}{aM_t}}$ //AA computes and send to DO using 2PC
8. $SK_{aa,U_t} \leftarrow P^{M_t} \leftarrow g^{\frac{(\beta+N_t)}{a}}$ //attribute authority generates secret key
9. $SK_{do,U_t} \leftarrow (\forall i \in T: D_i = g^{M_t}.H(i)^{r_i}, D_i = g^{r_i}), r_i \in_R \mathbb{Z}_p^*$ //DO generates secret key
10. $SK^*_{aa,U_t} \leftarrow H(U_t)^c$ //AA generates another secret key for attribute group key
11. $else$
12. exit
13. End

Encryption Function: The proposed architecture implements the access policy by represented it in the form of tree structure A_t. The access tree non-leaf nodes are designed using the threshold gate. The proposed architecture uses AND, OR, and $K - Threshold$ gates. The access tree leaf nodes denote attributes for both negated and non-negated user details. The existing CP-ABE encryption algorithm allows using NOT gates only at the tree structure leaf nodes. In contrast, the proposed architecture allows a user to provide non-monotonic access control.

As shown in Fig. 5.2, it illustrates an employee's access structure who needs to give access permission to other employees. The leaf nodes denote the non-negated attributes employee,

CSE, staff, big data, researcher, guest faculty, and one negated attribute student. The first attribute set is {employee, CSE, big data}. It represents the computer science branch employee having research area big data is an authorized access policy set. Whereas the second attribute set {student, postgraduate, graduate, researcher} represents unauthorized access policy set means graduate, postgraduate, and researcher students are not allowed to access employee's data.

The data owner outsources the file on the cloud server. To store the file on the cloud server, the data owner needs the access information $\mathrm{A_t}$ (Fig. 5.2) with universal attributes U and performs encryption to convert plaintext to ciphertext using attribute authority public key PK_{aa}.

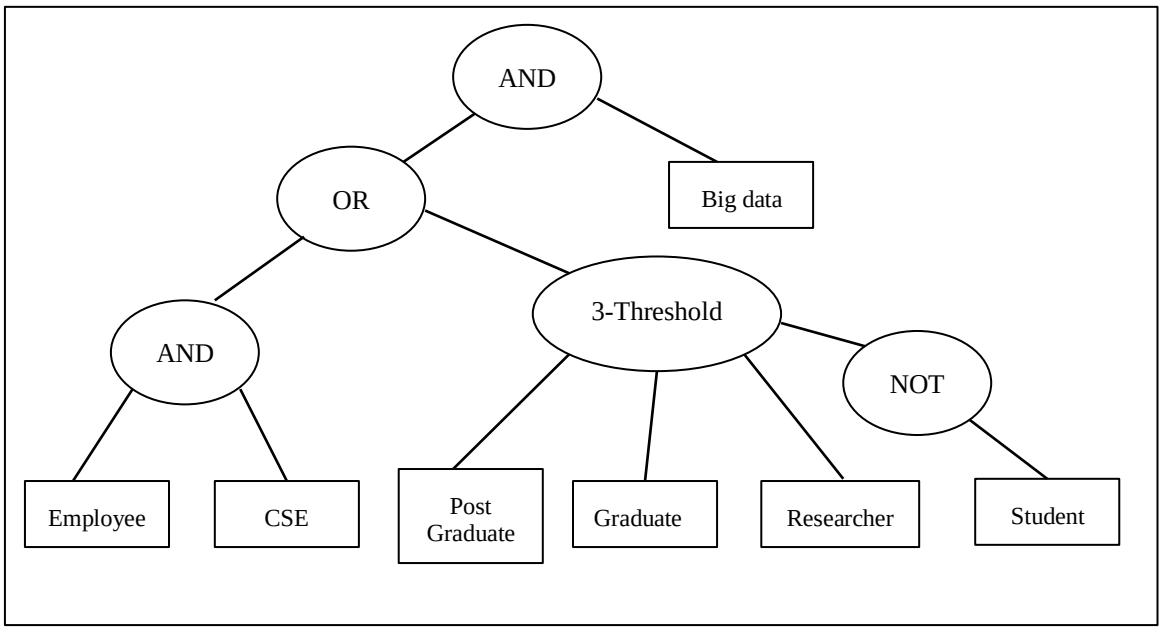


Figure 5.2: Tree of the Access Structure

Algorithm 5.4 selects a polynomial V_n for each node of the access tree A_t using a top-down approach. Then, the algorithm sets the degree d_n for V_x node that is one less than the threshold value of that node i.e., $d_n = K_n - 1$. For the root node R, the random value is selected from $\mathbb{Z}_p^*$ and set $V_R \leftarrow \mathcal{s}$. Next, for any other node of the tree, the algorithm sets V_x by assigning index values. In the end, the algorithm generates the ciphertext using L denotes leaf nodes of access tree A_t, attribute authority public key $e(g,g)^{\beta}$, data owner public key $\mathfrak{b}$, and polynomial function as represented in Line 11. The time complexity of the algorithm is $O(n)$.

Algorithm 5.4: Encryption

$Encrypt(F, PK_{do}, PK_{aa}, A_t)$

Input: File F, Public Key PK_{do}, Public Key PK_{aa}, and Access Structure A_t

Output: Cipher Text CT

1. Begin
2. For each Node $n\ in\ A_t$
3. Select a polynomial V_n //start from the root node
4. $d_n \leftarrow K_n - 1$ //$K_n = V_x$ node threshold value in A_t
5. $if\ x = Root_{node}\ R$
6. Let $\mathcal{s}$, $\mathcal{s}\ \in_R\ \mathbb{Z}_p^*$
7. $V_R \leftarrow \mathcal{s}$
8. $V_x \leftarrow V_{p(x)}\big(index(x)\big)$ //set node x index value to V_x
9. $end\ for$
10. Let $L \leftarrow Leaf\ Nodes(A_t)$
11. $CT = (A_t, C' = Fe(g,g)^{\beta \mathcal{s}}, C_1 = \mathfrak{b}^{\mathcal{s}}\ \forall\ l\ \in L: C_l = g^{V_y^{(0)}}, C_l = H(\delta_l)^{V_y^{(0)}})$
12. End

Re-Encryption Function: The re-encryption phase prevents the revoked user from accessing the plaintext. This function is executed by the attribute authority to re-encrypt the ciphertext by using attribute group A_G before outsourcing to the cloud server—this algorithm control user access according to attributes. Algorithm 5.5 selects the random number K_G from Z_P^* for each group member G_l present in attribute group A_G and calculates the re-encrypted ciphertext CT' as represented in Line 4. Then, the algorithm calculates the header information using randomly selected numbers P^* and R from Z_p^* and attribute group public key PK_{aa}^*. For each group, member belongs to A_G, the algorithm calculates the exponent function P. In the end, the header information is calculated as presented in Line 15. The authorized user, when requesting the cloud server, then responds with CT' and $Header$. Using this information, the user can decrypt the data until the user is not on the revocation list. The algorithm's time complexity is $O(n)$.

Algorithm 5.5: Re-Encryption

$Reencrypt(CT, A_G, PK_{aa}^*)$

Input: Cipher Text CT, Attribute Group A_G, and Attribute Group Public key PK_{aa}^*

Output: Cipher Text CT' and $Header$

1. Begin
2. $For\ each\ G_l\ \ in\ A_G$
3. Let $K_G,\ K_G \in \mathbb{Z}_p^*$
4. $CT' = (A_t, C' = Fe(g,g)^{\beta s}, C_1 = h^s\ \forall\ l\ \in L: C_l = g^{V_y^{(0)}}, C_l = \left(H(\delta_l)^{V_y^{(0)}}\right)^{K_G})$
5. $end\ for$
6. Let $P^*, R\ \in\ \mathbb{Z}_p^*$
7. $x_t = H_1\left(e\left(Q_t^{P^*}, PK_{aa}^*\right)\right) \forall\ U_t\ \in G$
8. $For\ each\ G_l\ \ in\ A_G$
9. $For\ each\ user\ U_i, i \leq n$ //n is the number of users in G_l
10. $i = i * (x - x_i)$
11. $K = K + a_i x^i$
12. $P_i \leftarrow g^{a0}$ // exponent function
13. $end\ for$
14. $f(x) = I = K\ mod\ p$
15. $Header_l = \{K_G.P_0^R, P_1^R \ldots., P_m^R\}$
16. $Header = Header_l$
17. $end\ for$
18. End

Decryption Function: This function works in two stages, as explained below:

1. Group Key Decryption Stage: The user requests the data owner for the data access. In response, the data owner sends a request to the cloud server and provides cipher text $(CT', Header)$ to the user. Then user generates the key using the attributes T associated with the user from the $Header$. For example, a user U_t having attributes λ_j means $U_t \in G_j$, and the user can obtain the attribute key K_l from $Header$ as shown below.

i. Calculates, $x_t = H_1\left(e\left(g^P, PK_{U_t}^*\right)\right)$

ii. Calculates $K_l.P_0^R.\prod_{i=1}^{n}(P_i^R)^{x_t^i} = K_l.g^{Rf^j(x_t)} = K_l$ where n is the number of users in the attribute group G_j.

Then user updates the secret key by using the generated attribute group K_l using the Equation 5.2.

$$SK_{U_t} = \left(SK_{aa,U_t}, SK_{do,U_t}\right) == (g^{\frac{(\beta+N_t)}{a}}(\forall i \in T: D_i = g^{M_t}.H(i)^{r_i}, D_i = (g^{r_i})^{\frac{1}{K_l}}) \quad (5.2)$$

The generated key process is secured as no one can secret key K_l other than the user U_t.

2. Data Decryption Stage: With the help of generated key user can decrypt the ciphertext CT'. The decryption process uses a recursive procedure using node x with its children. The decryption function is defined as $Decryptnode\left(CT', SK_{U_t}, x\right)$ where x denotes the leaf node of access tree A_t as given below.

i. Function $Decryptnode\left(CT', SK_{U_t}, x\right)$

$if\ \lambda x \in T\ and\ U_t \in G_x, then$

$$Decryptnode\left(CT', SK_{U_t}, x\right) = \frac{e(D_x, C_x)}{e(D_x', C_x')} = \frac{e(g^{N_t}.H(x)^{rx}, g^{V_x^{(0)}})}{e((g^{rj})^{\frac{1}{K_l}}, \left(H(x)^{V_x^{(0)}}\right)^{K_y})}$$

$$= e(g,g)^{rtV_x^{(0)}}$$

$If\ U_t \notin G_x\ or\ \lambda_x \notin T$, then the function returns null as shown in below equation.

$$Decryptnode\left(CT', SK_{U_t}, x\right) = e(g,g)^{rtV_x^{(0)}}$$

$$= NULL$$

ii. Function $Decryptnode\left(CT', SK_{U_t}, x\right)$, x is not a leaf node in the access tree A_t

The function recursively call all child nodes of x.

$Q_z = Decryptnode(CT', SK\mathbb{c})\ \{\mathbb{c} \in Childnodes(x)\}$

$\mathbb{L}_x\ denotes\ leaf\ nodes\ set\ of\ \mathbb{c}$

$Q_z = NOT\ NULL\ if\ \mathbb{L}_x \notin \emptyset$

$Q_z = NULL\ if\ \mathbb{L}_x \in \emptyset$

$$Q_x = \prod_{\mathbb{C} \in \mathbb{L}_x} Q_{\mathbb{C}}^{\Delta_{i,p_x(0)}}$$

$$where\ i = \mathbb{C}\ node\ index\ and\ p_x\ is\ \{index(\mathbb{C}) \notin \mathbb{L}_x\}$$

$$e(g,g)^{N_t V_x^{(0)}} = P$$

$$Plain\ text = C' * \frac{P}{SK_{U_t}} = \frac{\left[F * Fe(g,g)^{as^*} e(g,g)^{N_t.S}\right]}{e\left(h^s, g^{\frac{(\beta+N_t)}{a}}\right)}$$

$$= F$$

Key Update Function: If the user changes the attribute list by adding or removing attributes such as an address, email ID, contact number, department, etc., the access permissions for that user should be updated to preserve backward and forward secrecy. The attribute authority executes this smart contract function when the user request is received regarding updating a particular attribute group's attributes. After receiving the user request, the attribute authority first sends the updated attribute group list membership to the data owner to update the stored user-related information at the owner's side. Then, it generates new keys for the updated group attributes, and the updating process is completed. The process does not affect the remaining non-related user's keys due to the changed group attributes. This phase works as follow:

i. The attribute authority selects a random number s' and an attribute key K_l. Perform encryption of CT using PK_{aa}^* as shown in Equations 5.3 and 5.4.

$$CT = A, C = Fe(g.g)^{\beta(s'+s)}, C = h^{s'+s}, C_i = g^{V_i^{(0)}+s'} \tag{5.3}$$

$$C_i = \left(H(i)^{V_i^{(0)}+s'}\right)^{K_i}\ \forall l \in L \setminus \{i\}: C_l = g^{V_y^{(0)}+s'}, C_l = \left(H(l)^{V_l^{(0)}+s'}\right)^{K_l} \tag{5.4}$$

ii. The attribute authority uses a new attribute group to create a polynomial function $f(x)$ for including or excluding users. Then, it creates a new header message by calculating a new $Header_i$ using K_i as given in the Equation 5.5.

$$Header = (g^P, Header_i\ \forall l \in L \setminus \{i\}: Header_l) \tag{5.5}$$

Whenever a user requests the cloud data, the data owner replies with the header information and ciphertext. The user can only decode the ciphertext when the attribute group satisfies.

The above algorithms ensure access permission at various levels and also maintain access restrictions for different users.

5.3 Security Analysis

The proposed architecture defined the following security goals to describe the access and revocation process.

Data Protection: The proposed architecture ensures data protection from unauthorized users as it allows only users to decrypt data if they have enough attributes. If the user is revoked from the group, he cannot access that group's plaintext. The proposed architecture achieved this using the immediate attribute revocation process. The ciphertext is re-encrypted using a group-based attribute access policy instead of the whole access policy. Another possibility of attack may be from the cloud server or attribute authorities. There may be chances that they may share the information for their profit. We deployed two key generation methods to make this process independent of a single authority to resolve this issue. If the user requests the registration process, then the data owner and attribute authority independently generate separate keys and send them to the user. Then the user generates the secret key using these keys. Hence, the proposed architecture guaranteed confidentiality and data protection.

Collusion Tolerance: The proposed architecture avoids a collision attack that is the main security requirement in the ABE algorithm. If multiple users coordinate with each other, they may decrypt the ciphertext by linking the attributes. Therefore to avoid such type of attack, the attribute authority cannot coordinate with the revoked user. Also, the user uses the unique random value to generate the secret key. For the collusion attack, the attacker should recover the $e(g,g)^{\alpha s}$ to decrypt the ciphertext. The attacker cannot perform the decryption process until he gets the random value of the user.

Backward and Forward Security: Backward secrecy means if the new user joins the group, he cannot access the cloud server's data before. Forward secrecy deals with restricting access of revoked users for subsequent cloud data that will be outsourced in the future, except if the user satisfies the access policy to the other valid attributes. The proposed architecture achieves backward and forward security by using an immediate user revocation process instead of timely revocation. If the user discards or updates an attribute in a group, the re-encryption process deploys using a new secret key. Then, the generated key is shared with all the related group users.

5.4 Result Analysis

This section presents the details of the proposed architecture experimental setup, performs performance evaluation, and comparative analysis.

5.4.1 Experimental Setup

The proposed architecture is designed using the Java programming language. The proposed architecture utilizes the org.cloudbus.cloudsim package to simulate the workload, load balancing, and policy-related implementation using different Java classes such as DatacenterBroker and CloudletScheduler Vmallocationpolicy, etc. The proposed architecture created the CloudSim environment of 15 data centers, 50 virtual machines, and 100-1000 tasks (transactions) by implementing the different classes. The proposed architecture is designed so that it can be easily further extendable to include new functionality or deploys in real-world scenarios. The architecture was developed in three parts. The first part implements the graphical user interface to provide essential services to users. The second part deploys the main logic of the architecture using smart contract functions. Different smart contract functions are defined to achieve different services of the proposed work. Each participant of the blockchain network executes the smart contract function in the form of a transaction that follows blockchain procedure to append it as a block in the blockchain structure. Lastly, the back end of the architecture design uses the CloudSim tool to store ciphertext in the cloud storage. The real cloud environment exhibits varying demand-supply patterns and system size depending on the proposed work composition, configuration, and deployment requirement.

Moreover, the users have heterogeneous and competing quality of services requirements. Thus, the proposed work uses the simulation to evaluate the performance and test the services of the architecture in a repeatable and controllable environment free of cost, identify the performance bottleneck and handle the complexities that arise. Furthermore, the proposed architecture design works independently, allowing the update in one part without affecting the main logic of the proposed work. Therefore, the proposed architecture can be easily extended to include the new functionality or utilize the real cloud platforms such as Amazon, Microsoft Azure, etc., by updating the connectivity classes.

5.4.2 System Performance

This section analyses the performance of the proposed architecture and also compare it with the cryptcloud [92] scheme, NDN technique [93], and Proxy technique [94]. We consider the encryption, decryption, key generation, and re-encryption functions' performance time for the comparison process. The encryption time denotes the time required to convert the plaintext to ciphertext, whereas decryption time defines the time required to obtain the plaintext from the ciphertext. The re-encryption time involves the time needed to re-encrypt the ciphertext. Furthermore, the key generation time includes the time required to generate the keys for the user. As shown in Fig. 5.3, we have calculated the key generation time by varying user attributes in the system and compared it with the cryptcloud scheme [92]. We can depict from the analysis that the proposed work requires less time for the key generation process. The proposed scheme used a bilinear-based cryptography approach to generate the secret keys for the user. In contrast, cryptcloud used a semi-trusted key management center to create keys and ciphertext conversion that affected security features. The cryptcloud scheme is a semi-distributed architecture; thus, the proposed scheme provides a better approach.

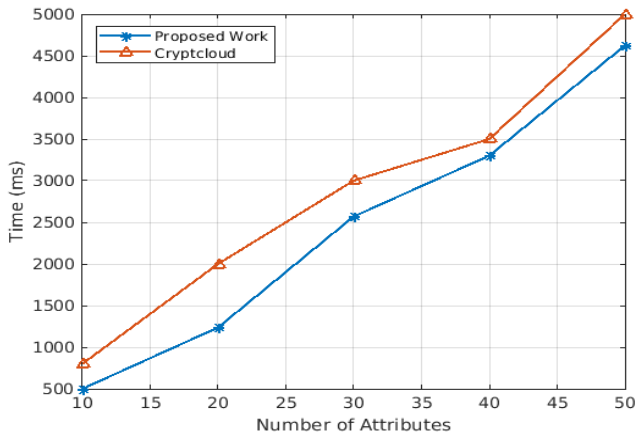


Figure 5.3: Key Generation Time Comparison between Proposed Scheme and Cryptcloud

Fig. 5.4 shows the encryption time comparison between the proposed architecture with cryptcloud [92], NDN [93], and proxy [94] schemes. In all approaches, the encryption time increases with the number of attributes. Therefore, we can analyze that the proposed system requires less time to perform the encryption process than the existing literature. The proposed

scheme uses a robust encryption process with a Pairing-based library and a 160-bit elliptic curve group using a supersingular curve in the 512-bit finite field. In contrast, the cryptcloud technique utilizes the symmetric session key to encrypt the plaintext, thus needing more time to share the same key for both encryption and decryption processes. The NDN and proxy schemes involve the proxy servers to perform the encryption and decryption, which increases the overall time for both processes. Similarly, Fig. 5.5 depicts that the proposed approach's decryption time is less than existing techniques because the decryption process involved in existing work requires more operations to achieve a user-based revocation process.

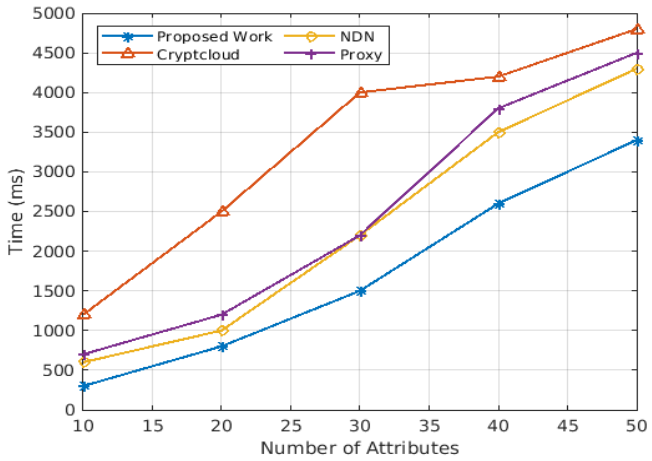


Figure 5.4: Encryption Time Comparison between Proposed Scheme and Existing Techniques

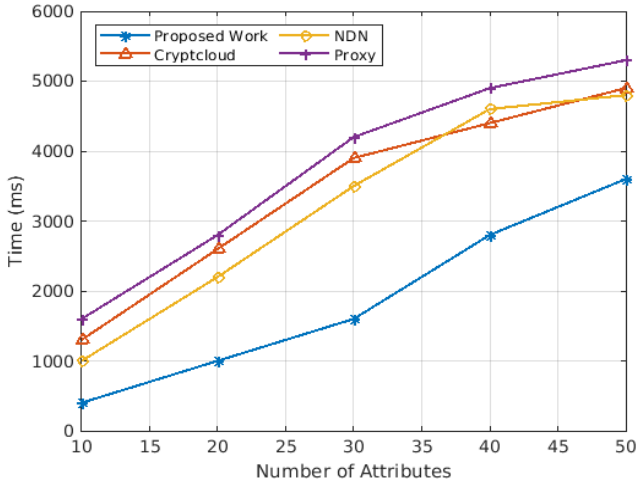


Figure 5.5: Decryption Time Comparison between Proposed Scheme and Existing Techniques

Fig. 5.6 presents the time required by the proposed work for the re-encryption process and compares it with the NDN technique [93] by varying the number of attributes. It is observed that the increment in the number of attributes also increases the re-encryption time in both approaches. Also, the proposed work performs better than the existing literature NDN. The NDN technique involves the proxy re-encryption approach that requires the agent module to respond according to the user's request. Therefore, it requires extra pre-processing before the re-encryption process. In contrast, the proposed work directly executes the re-encryption smart contract function for the user request and reduces the extra overhead. Therefore, the proposed architecture provides a reliable and better solution.

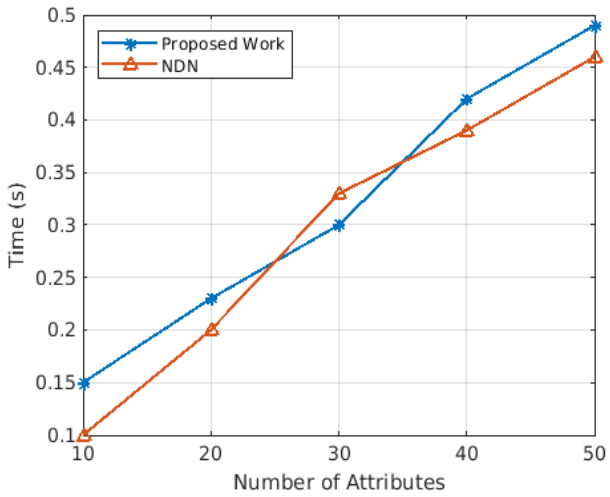


Figure 5.6: Re-Encryption Time Comparison between Proposed Scheme and NDN Technique

5.4.3 Comparative Analysis

This section compares the proposed work with the existing techniques using various parameters.

Table 5.2 compares related work with the proposed scheme based on parameters like authority type, access policy, key escrow, and revocation method. Compared to existing work, our proposed architecture used multiple authority systems to generate keys for the users. Furthermore, the architecture used a non-monotonic access policy. Negative attributes define access set attributes that make the access structure clearer compared to other methods. It also employed the re-keying method to implement an immediate attribute revocation approach rather than time-based attribute revocation. This provides a more secure

environment for cloud data in terms of forward and backward secrecy. The proposed scheme achieves fine-grained access control by using the re-encryption technique and using two authorities to compute keys for the user. Also, the used key generation process has solved the key escrow problem with two authorities' help.

Table 5.2: Comparison of Related Work with Proposed Scheme

Scheme	**Authority**	**Expressiveness**	**Key Escrow**	**Revocation**
[99]	Key	NA	Yes	Time attribute revocation
[100]	Key	Non-monotonic	Yes	Immediate attribute level
[101]	Single	Monotonic	Yes	Time attribute revocation
[102]	Single	Monotonic	Yes	Subset difference revocation
[103]	Multiple	Monotonic	Yes	Immediate attribute level
[104]	Multiple	NA	Yes	Time attribute revocation
[105]	Key	Monotonic	Yes	Immediate attribute level
Proposed Scheme	Multiple	Non-monotonic	No	Immediate attribute level

Table 5.3 compares the proposed scheme efficiency with the related work. We perform a comparative analysis of computation cost for generating various keys. First, the communication cost for sending and receiving data between the data owner/authority and the cloud server is measured using ciphertext key size. Second, the user's storage cost is measured from private key size. Last, the attribute authorities' public key size of the system is measured for comparison. From the comparison, we can conclude that our scheme is the most efficient CP-ABE with direct revocation. The proposed scheme space and computation complexity do not depend on N_u; the total number of users in the system, which is supposed to be huge. Also, the proposed architecture private and public key sizes are smaller than the existing approaches. It requires less computational overhead without involving logarithmic operation. Thus, the proposed work provides a more reliable and efficient environment than the existing techniques. The following notations are used in Table 5.3:

$E_0 \rightarrow Element\ bit\ size\ in\ \mathbb{G};\ E_1 \rightarrow Element\ bit\ size\ in\ \mathbb{G}_T;$

$E_A \rightarrow Bit\ size\ of\ access\ tree\ A_t;\ A \rightarrow Attribute\ count\ in\ A_t;$

$N_u \rightarrow User\ count\ in\ attribute\ group\ G;\ R \rightarrow Number\ of\ revoked\ users.$

Table 5.3: Efficiency Comparison of Related Work with Proposed Scheme

Scheme	Ciphertext Key Size	Private Key Size	Public Key Size	Algorithm
[99]	$(1 + A + \log R)E_0 + E_1$	$((E_A + 1)\log N_u)E_0$	$(\log R + A)E_0 + E_1$	KP-ABE
[100]	$3E_0 + E_1$	$\big((R + 1).E_A\big).E_0$	$(R + A + 1)E_0 + E_1$	KP-ABE
[105]	$(2 + 2A)E_0 + E_1$	$4E_AE_0$	$(3 + 2A + R)E_0$	KP-ABE
[101]	$(2 + A)E_0 + E_1$	$(1 + E_A)E_0$	$(\log N_u + E_A + 3)E_0$	KP-ABE
[102]	$(16A + 64R - 27)E_0 + E_1$	$(5 + 16E_A + 16(\log^2 N_u + \log N_u)E_0)$	$111E_0 + E_1$	KP-ABE
[104]	$4N_uE_0 + E_1$	$8N_uE_0$	$32N_uE_0$	CP-ABE
[103]	$\big(16\sqrt{N_u} + 3A\big)E_0 + E_1$	$\big(2 + 2A + \sqrt{N_u}\big)E_0$	$(5 + 8\sqrt{N_u})E_0 + \sqrt{N_u} + E_1$	CP-ABE
Proposed scheme	$(2A + 1)E_0 + E_1 + E_A$	$(2K + 2)E_0$	$E_0 + E_1$	CP-ABE

5.5 Conclusion

The proposed architecture introduced a blockchain-based fine-grained access control method using the CP-ABE algorithm to provide a robust user revocation process in the cloud storage systems. The proposed methodology utilized the two-authority-based key generation scheme to resolve key escrow issues and make the system independent on a single authority. Thus, it is difficult for the attribute authority or cloud servers to misuse the outsourced data. Furthermore, the proposed scheme ensures the outsourced data's privacy and confidentiality by restricting the users from accessing the data without proper credentials. The proposed architecture deployed the immediate attribute level user revocation process rather than time-based to provide scalable access restriction using the CP-ABE algorithm. The experimental results, performance evaluation, and comparative analysis indicate that the proposed architecture offers a more efficient and scalable cloud environment. For future work, we plan to deploy the proposed architecture in various domains such as healthcare, multimedia, intrusion detection system, etc., to analyze the performance.

CHAPTER 6

APPLICATIONS OF PROPOSED WORK

This chapter highlights the application areas of the proposed blockchain-based cloud architecture. The proposed work is deployed in various domains such as healthcare, intrusion detection, etc., to evaluate its performance. The result analysis and performance evaluation depict that the proposed work provides a more reliable cloud storage environment in different domains.

6.1 Introduction

Blockchain is one of the most hyped advances nowadays and has gained considerable importance as an innovation widely deployed in various areas such as healthcare, supply chain, multimedia, artificial intelligence, etc., [106, 107]. After its commencement in 2008, blockchain has continued to develop as a disruptive advance that might alter the way we interface, make computerized expenses, follow up, and monitor transactions [108]. Blockchain could be cost-effective, removing the centralized authority's need to monitor and regulate transactions and interactions between different members. In the blockchain, every transfer is cryptographically marked and confirmed by other entities holding a copy of the entire record consisting of all the transactions. Furthermore, blockchain technology is an information technology that can be used in software, business, and trade sectors [109].

The widespread deployment of blockchain demonstrates the aspects that change the business community's activities. A lot of new technologies and frameworks have been introduced with the existing keen interest in blockchain technology. Numerous articles were published to explore the advantages of blockchain for existing applications. Examples of these studies include the blockchain technology for business applications [110, 111], healthcare [112, 113], security [114, 115], sharding [116], cloud exchange [117], edge computing [118], and so on.

Certain studies dealt with blockchain obstacles, prospects, and plans for the future. We aim to provide a detailed overview of the proposed blockchain-based cloud architecture usage in various domains such as healthcare, intrusion detection systems, multimedia etc. In this chapter, we implemented the proposed work in healthcare and intrusion detection systems to evaluate the performance of the decentralized cloud storage architecture.

6.2 Blockchain-based Healthcare System

The healthcare system is an information-intensive medical environment where large amounts of data are routinely generated, obtained, and disseminated. Due to the sensitivity of data and restricting factors, such as protection and privacy, storing and distributing this vast volume of data is crucial, as well as significantly challenging [119]. Secure data management is essential for diagnosis in the healthcare sector and clinical settings. In healthcare, the protection of medical information has been innovated in the last decade through many platforms, software, and communication technologies. In [120], authors first translated the health records of paper into Electronic Health Records (EHRs). EHRs must be regularly distributed and exchanged by various healthcare centers, doctors, nurses, healthcare professionals, pharmacy manufacturers, and administration to provide a realistic way for a person's health background to give proper and prompt treatment. In the case of a conventional client-server data management healthcare system, each hospital/healthcare center maintains its database of medical records of a sick person; the delivery of EHRs becomes a slow and costly task. Moreover, cloud-based health information monitoring methods [121, 122] have been presented to solve the accessibility, data usage, and maintenance issues that exist in the client-server architecture. Patient medical information from various hospitals is saved in remote online storage, making it readily available to patients and healthcare professionals. However, the cloud environment faces data security and single point of failure issues.

Therefore, many research articles have employed blockchain technology in existing systems to address the above challenges [123-125]. For example, in [126], the authors proposed a distributed blockchain-based healthcare system, MedRec, that interacts with existing physician data storage solutions and allows for scalability. However, the authors use patient information as a reward for miners, keeping the security of patient information at greater liability. The authors of [127] and [128] proposed a smart contract-based system for accessing health information using Ethereum platform. They ensure access control and preserve medical data privacy by employing advanced cryptographic techniques. Fan et al.

[129] recommend MedBlock, a blockchain-based health information delivery scheme that provides efficient accessibility and extraction of EHRs for an authenticated network. These works [126-129] do not permit patients to transfer data on their health problems and activities to the blockchain network, thus lacking a treatment process.

On the other hand, in [130], the authors suggest a medical data network to exchange health information between health centers and patients through blockchain. This work permits hospitals and physicians to upload patient health records on the blockchain network, giving a complete overview of patients' records. The proposed scheme is specifically designed for patients, thus lacks the involvement of other healthcare professionals. Shen et al. [131] introduce a system to use blockchain and peer-to-peer networks such as MedChain to exchange medical data. This system produces healthcare data via medical inspection and collects patient data from IoT sensors and other mobile applications. The usage of two decentralized networks increases the complexity of the proposed scheme.

Therefore, this research work resolves the challenges mentioned above and proposes a blockchain-based distributed application to protect large-scale healthcare data called Healthify. In the Healthify application, clients are allowed to publish healthcare information and access treatments from doctors. Healthify supports integrity checking and enhances the security of healthcare data. In this architecture, users can upload and publish health data periodically. There is a large amount of medical information with the exponential growth of the hospital's report. It is insufficient to document complete user information in the blockchain, as each node's resource requirements are incredibly high. Considering each blockchain node's limited storage capacity, an IPFS storage system supports sharing the document for high integrity and durability data storage [132]. There is no single repository in IPFS, and the information is circulated and collected in various IPFS nodes throughout the internet.

6.2.1 Proposed Architecture

This section presents the architecture of the proposed application to manage the healthcare data securely. The architecture contains the main components, such as a smart contract, IPFS storage, and Distributed Application (Dapp), as shown in Fig. 6.1. A blockchain-based smart contract is designed to check the authenticity of the users and maintain the integrity of the healthcare data. Smart contract execution triggers automatically whenever the user initiates a request to upload/access the healthcare data to check the user's authenticity. The healthcare

data is stored in the form of blocks in the peer-to-peer IPFS storage network. The application is implemented to guarantee that anybody, including the users themselves, cannot manipulate users' transactions. The application has three types of user transactions: data transactions, data access transactions, and validation transactions. Data transactions are used to upload healthcare data, data access transactions are used for accessing data, and validation transactions are handled to safeguard data integrity.

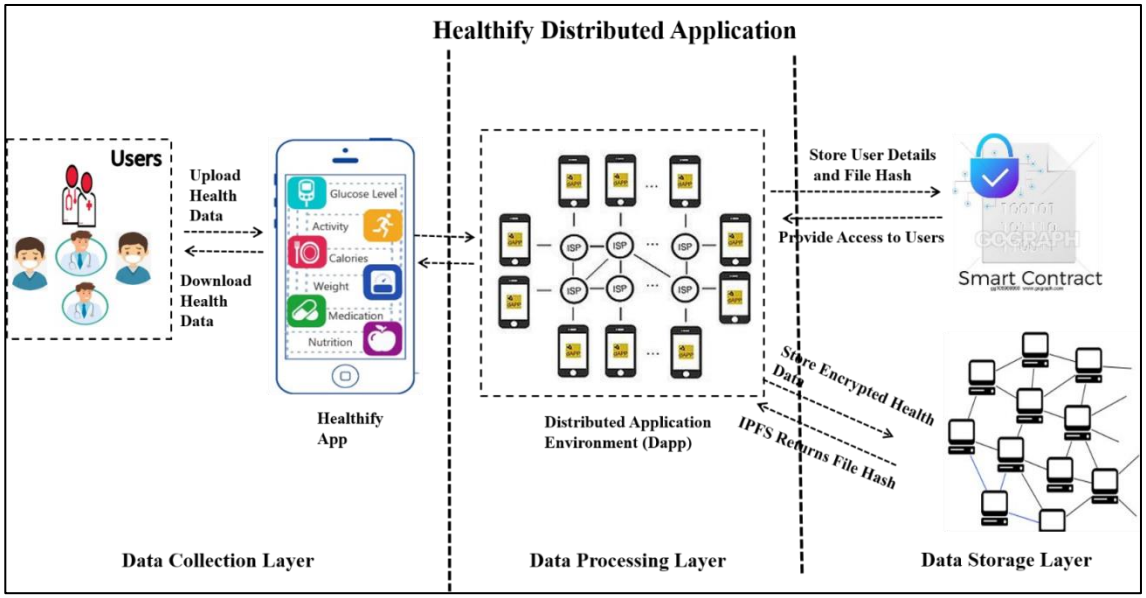


Figure 6.1: Architecture of Blockchain-based Distributed Application for Healthcare

Fig. 6.1 presents a layered architecture of the application. It describes all the application entities in different layers showing how the data flows through them and the functionalities of each layer.

Data Collection Layer: The first layer is the data collection layer. It consists of different users and the user interface of Healthify through which the users interact with the application. Firstly, users register on the platform using a Dapp, and their details are stored on the Ethereum network using the smart contract. The user receives a unique address using which the user interacts on the Dapp. Through the application portal, users may collect their health data. The users may upload the data manually or set a time after which the data will upload. The user sends data on the Dapp, where it accumulates the healthcare data in files, and the user can also visualize the data and registered doctors list.

Data Processing Layer: The second layer is the data processing layer. Healthify utilizes the Ethereum platform for implementation, and the blockchain user utilizes the platform's

functionalities. The users authorize by their public addresses and digital signatures, generated using their private keys, which ensure the authority. User access manages using the public-private keys, and the users can access the data only according to their provided access and authority.

Storage Layer: The last layer is the storage layer, consisting of smart contract and IPFS storage. The smart contract provides a primary application backend that governs all tasks and authorities of the user. A smart contract manages the authenticity of the users by checking if the legit user is using the application. The IPFS storage layer is where the individual health care data are stored, and the user has received a unique hash of the file. The data is encrypted using the AES algorithm before uploading it to the IPFS storage node.

6.2.2 Smart Contract Functions

The smart contract supports user registration, authentication, integrity checking functions, and it is published on the blockchain afterwards. The blockchain also publishes all tasks conducted using the smart contract. In a proposed architecture, the smart contract consists of the various functions, as shown in Fig. 6.2 for each entity. The functions are in such a way that Healthify users can execute and get access to storage services.

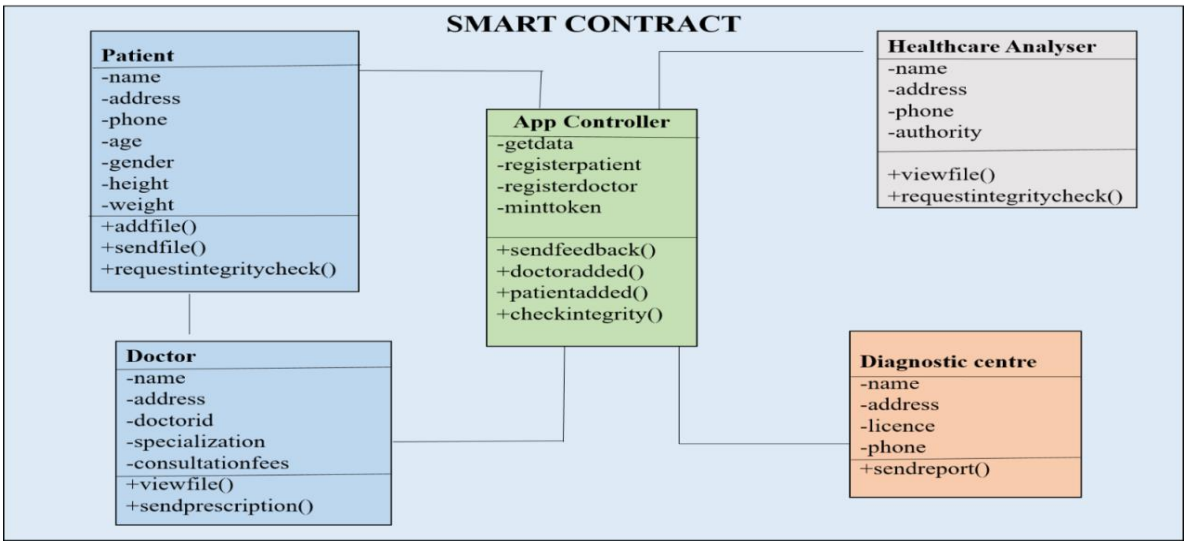


Figure 6.2: Smart Contract Functions

Algorithm 6.1 represents the function to register the user on the application. The conditional statements check if the user is already registered as a patient (or doctor) or not; if yes, then the function reverts with an error message written after the condition. If the task is completed,

it emits an event PatientAdded showing the address of the currently registered user. Similarly, the same function is designed for other users of the application.

Algorithm 6.1: User Registration

AddPatient()

Input: Registered user address

Output: Successful registration of a user

```
1. if (isDoc[msg.sender] == false)
2.     Print "Address is Doctor";
3. end
4. if (isPatient[msg.sender] == false)
5.     Print "Address is already Patient";
6. end
7. isPatient[msg.sender] == true;
8. allPatients.push(msg.sender);
9. emit PatientAdded(msg.sender);
```

Algorithm 6.2 represents the file uploading function. This function is called when the user uploads the health data on the blockchain. This function stores the IPFS hash of the encrypted file on the smart contract. The conditional statements check if the function is called by a valid user only.

Algorithm 6.2: Upload File

AddFile(***_fileHash***)

Input: File Hash

Output: Successful uploading

```
1.  if (isPatient[msg.sender] == true)
2.      Print "Address is not Patient";
3.  end
4.  PatientData[msg.sender].push(_fileHash);
```

Algorithm 6.3 function is called when the user wants to share (or send) the data to any authorized user. It shows the sample function for registered patients. This function also deducts the doctor's fee from the patient's account and reverts if the user has an insufficient token balance. Similarly, the function is called by the doctor/diagnostic center to send a prescription/report to the patient. Once the prescription/report is sent, the user receives the fee stored in the contract.

Algorithm 6.3: Share File

***SendFile*(*address_doc*, *_fileHash*, *_amount*)**

Input: Registered User Address, File hash, Token Amount

Output: Successful Sharing of File

```
1.  if (isPatient[msg.sender] ==true)
2.     Print "You are not Patient";
3.  end
4.  if (isDoc[_doc] == true)
5.     Print "Invalid Doctor";
6.  end
7.  if (_amount == docFee[_doc])
8.    Print "Insufficient fee";
9.  end
10. token.approveContract(addresss, msg.sender, _amount);
11. token.transferFrom(msg.sender, address, _amount);
12. docPatientList[_doc.push(msg.sender);
13. docPatient[_doc][msg.sender]=true;
14. docData[_doc][msg.sender].push(_fileHash);
```

The users call the Algorithm 6.4 functions to check the integrity of the files stored on the IPFS storage. IPFS storage shared the unique hash for each saved record. This unique hash of the file is used to test the validity of the files. The smart contract uses the stored Metadata of the files to audit integrity. The file hash value was registered when the file was uploaded. The user sends an access request to the blockchain network and automatically executes the smart contract function. Then, after checking the user's status, a smart contract requests a file hash from the network. The smart contract selects the requested file to check the integrity by

recalculating the hash and compares the new hash with the previously-stored hash. If they are equal, the data integrity is safe; otherwise, not. In the end, the network returns the result to the user.

Algorithm 6.4: Check Integrity

$\boldsymbol{CheckIntegrity}(\boldsymbol{file}, \boldsymbol{user})$

Input: File Details

Output: Checking of Integrity

```
1.  var reader = new FileReader();
2.  var r = bcrypt.hash(reader.result, salt, function(err, hash){
3.  if(err)
4.    Print "err";
5.  end
6.  else
7.     this.setState({bcryptHash: hash});
8.  end
9.  reader.readAsDataURL(f);
10. varinst = this.props.state.contract;
11.  inst.methods.isUnique(this.state.bcryptHash.toString()).call().then(function(res){
12.  if(res==true)
13.     Print "Integrity completed";
14.     return true;
15.  end
16.  else
17.     return false;}
18.  end
```

6.2.3 Conceptual Scenario

This section presents a model that shows how the user interacts with the Healthify application and all processes functionalities. The application consists of four separate users. The model of interaction for each user is described below:

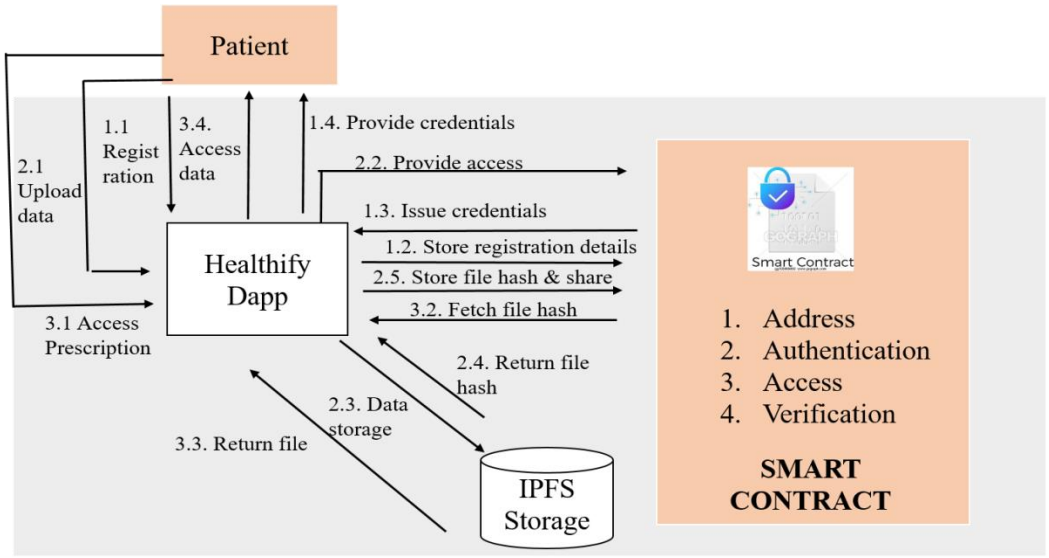


Figure 6.3(a): The Flow Diagram of Patient

Patient: Patient satisfaction is an important aspect of the medical sector and the lifeline for any health-related enterprise or initiative. Personal well-being is a concern for most of us, and that is why the need for the hour is to find the most effective ways to improve health conditions. Therefore, this work introduces the Healthify application to provide instant medical services to users. Fig. 6.3(a) represents the stepwise overall flow of the patient in the application.

Step 1: Initially, the patient registers with the application by providing personal information, including his name, age, sex, etc., and data is stored on a smart contract. The registration phase is compulsory until the user can use the functionalities of the software. The user then logs on using their specific address.

Step 2: Upon logging into the application, the user has a few options on the portal. The user can upload the health data using the portal options. Before the data uploading process, data is encrypted using the AES algorithm to provide a more secure environment. After encryption, information is divided into multiple shards and stored in the distributed platform using the IPFS system. In response, the user receives a unique hash corresponding to the uploaded file, which is further utilized by the user to share the file with the doctors or to access the file. The data file for healthcare is created from the user's data over a given period. After this file is submitted to the application, the applicant will continue obtaining the diagnosis/prescription from the application's registered doctors.

Step 3: The user selects the doctor from the registered doctor list and sends the individual stored health data unique hash to the doctor. When sending the request, the selected doctor's fees are deducted from the patient's account and saved on a smart contract. The smart contract automatically transfers the stored tokens into the doctor's account once the patient receives the prescription.

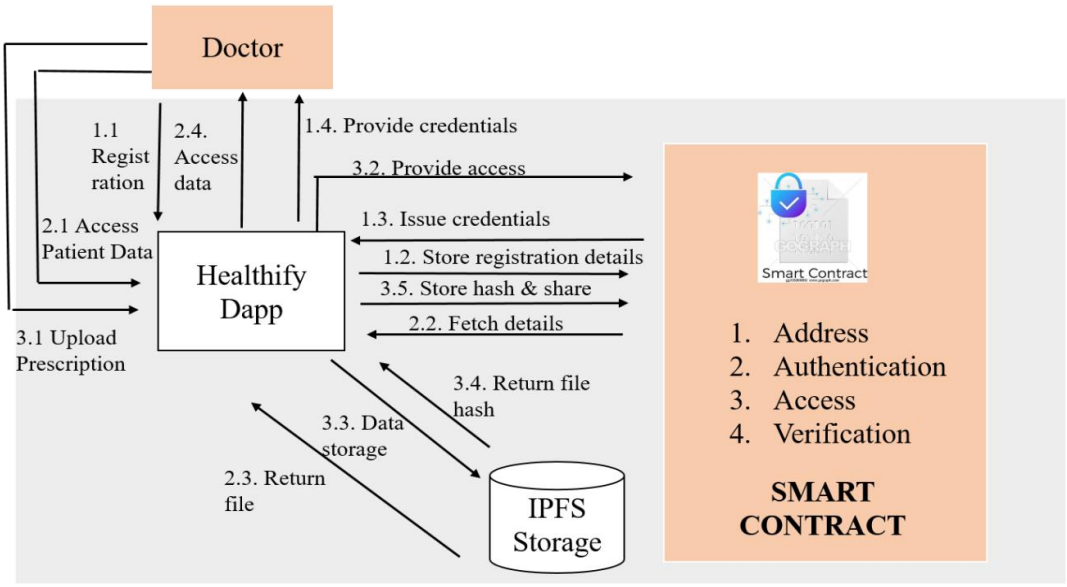


Figure 6.3(b): The Flow Diagram of Doctor

Doctor: The work of a doctor is essential to every medical care process, and we include the provision in our proposal to obtain input from doctors. The patient should be able to report to the selected doctor. But this feature must depend entirely on the customer's decision whether they follow the doctor's feedback. Interaction between the patients and the doctors was a significant obstacle due to physicians' hectic schedules and availability. Therefore, the proposed application allows interaction between the patients and the doctors to combat the issues mentioned above. Fig. 6.3(b) represents the stepwise overall flow of the registered doctor.

Step 1: The doctor registers on the application using the same procedure and then uses his unique credentials to sign in to the application.

Step 2: When the doctor is logged in, they should view a patient's data via a user interface, which allows them to pick the patients. After the patient's selection, the data should be available for review by the doctors. The doctor should add their suggestion or input after reviewing the patient's information. The data used for monitoring will not be editable by

either the patient or the physician. Doctors are allowed to view the files using the file hash shared by the patients.

Step 3: Once the doctor uploads the prescriptions, the prescription is sent to the patients in the same manner, and the doctor receives their fee in the form of tokens, especially designed for the proposed application.

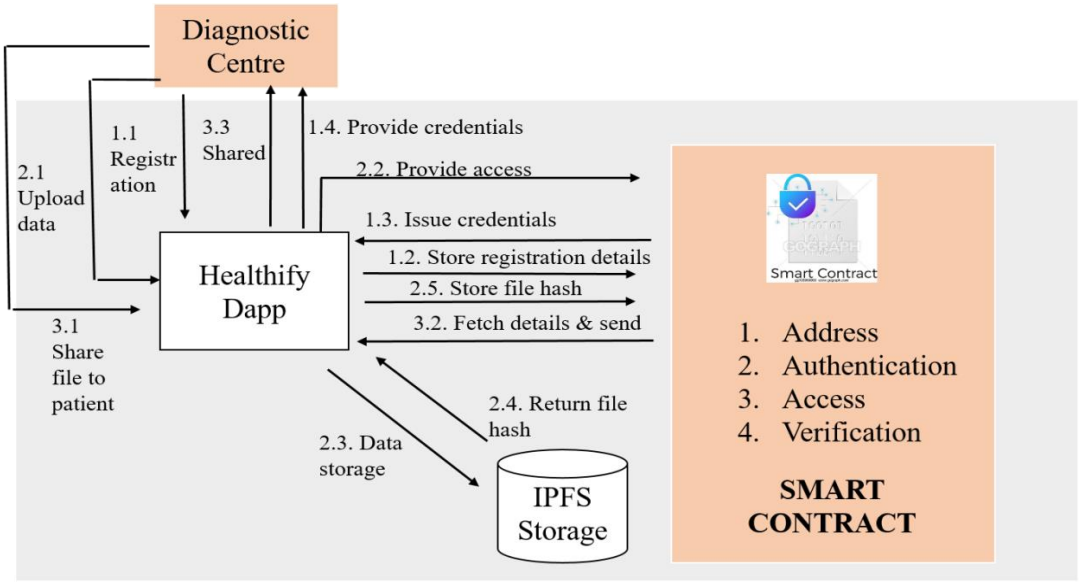


Figure 6.3(c): The Flow Diagram of Diagnostic Centre

Diagnostic Centre: One of the most tedious tasks for everyone is receiving medical records from test centers. Mobile applications make electronic monitoring of their health records simple for patients. Patients may check the reports directly from the centers, which can be exchanged immediately with the doctors. Thus, the proposed application allows diagnostic centers to register at the portal and provide quick services to the users more securely. Fig. 6.3(c) represents the overall flow of the registered diagnostic center.

Step 1: The diagnostic center starts with the registration process and obtains unique credentials.

Step 2: After logging into the system, the diagnostic center may store the generated reports to the IPFS storage and obtain the file's unique hash.

Step 3: The diagnostic center shares the stored report hash to the registered patients by checking the details stored on the smart contract. The diagnostic center also allows checking the integrity of the shared document and ensures security.

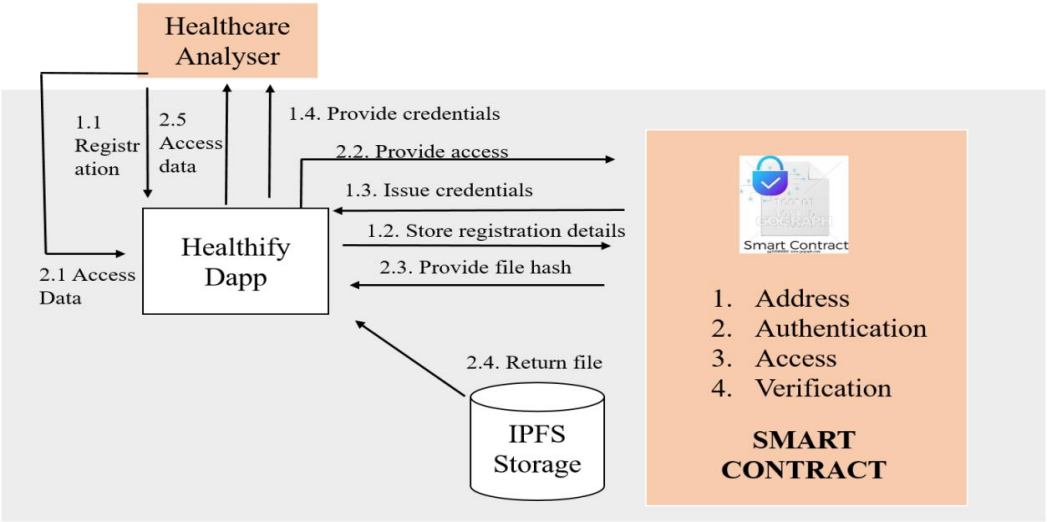


Figure 6.3(d): The Flow Diagram of Healthcare Analyser

Healthcare Analyser: Healthcare experts include a wide range of specialists and practitioners who provide some form of healthcare service, including primary care practitioners such as nurses, doctors, surgeons, physical therapists, social workers, etc. Health analysts play a key and essential role in enhancing health care quality. Thus, the proposed application included the interface for healthcare analyzers to provide a medium for improving healthcare quality. Fig. 6.3(d) shows the stepwise overall flow of healthcare analyzers.

Step 1: The healthcare analyser starts with the registration process and obtains unique credentials.

Step 2: After the registration process, the healthcare analyser may utilize healthcare data to improve healthcare services, tools, medications, and diagnostic methods.

6.2.4 Result Analysis

This section evaluates the performance and validates the efficiency of the proposed application.

6.2.4.1 Experimental Setup

The proposed architecture implements a Dapp that supports a blockchain network with a decentralized file system (IPFS). The Ethereum framework has been used to develop smart contracts for healthcare blockchain. This is an open-source platform and presently one of the largest public blockchain networks with an active community and a large collection of public Dapp. The Dapp can detect discrepancies, unauthorized access to the data, and missing

objects. Ganache tool is used to setting up Healthify's blockchain network to deploy contracts, develop Dapp, and run tests. The proposed application experimental setup consists of a Dapp setup and a smart contract deployment. Thus, the implementation settings are described in two tables to explain each part. Table 6.1 describes the development environment of a Dapp. The user interface is designed using React native as it has excellent compatibility with the Ethereum client. NodeJS is used to connect with the backend, i.e., connection with Ethereum and IPFS. The deployment settings of smart contracts are described in Table 6.2. Smart contracts are developed in Solidity language, which is the primary smart contract language for Ethereum. These are designed by using online compiler remix.ethereum. The key elements of the smart contracts are functions, events, state variables, and modifiers and are written in the Solidity programming language. The remix test network is used to deploy smart contracts on the testnet, and Ethers are utilized to pay the transaction fee.

Table 6.1: Development Environment for the Proposed Application

Component	**Description**
RAM	4 GB
Operating System	Windows 10
Server	Apache Tomcat
Frontend	React Native
Backend	Node JS
Host	Infura
Encryption	AES
Data Storage	InterPlanetary File System

Table 6.2: Development Environment for the Blockchain Smart Contract

Component	**Description**
RAM	4 GB
Operating System	Windows 10
Ethereum	2.0.
IDE	Remix Ethereum
Programming Language	Solidity

6.2.4.2 Performance Evaluation

This section presents the actual results of the work to assess the performance of the proposed application. Several experimental tests were performed using various parameters. The processing time would include the time to send a transaction query to access the health document and the amount of time it takes to upload until the user receives an acknowledgment. We used the different sized health files for this test and noted the time for each file uploading process, as shown in Table 6.3. These are approximate times, and these solely depend on the number of peers and the internet connection speed at the moment.

Table 6.3: Time Required for Uploading Different Sized Files

File Size (x)	Time (in seconds)
x < 1 MB	~ 0.02
1 MB < x < 10 MB	~ 0.1 - ~ 2
10 MB < x < 100 MB	~ 2 - ~ 15
100 MB < x	~ 15 <

The proposed architecture is also evaluated for computation time required for data storage, access, and validation transactions. The computation time is the average time the proposed application takes to execute the series of transactions requested by the users. As shown in Fig. 6.4, the Healthify application calculated the computation time for a series of hundred transactions. Different users initiate a total of five hundred transactions to analyze the computation time of the proposed application for different types of transactions.

There is a requirement to estimate costs associated with deploying smart contracts for healthcare to execute blockchain. In the Ethereum blockchain, all programmable calculations cost some fees to prevent network misuse and solve other computational issues. The fee is listed as Gas in the Ethereum blockchain to run all kinds of transactions. Gas refers to the payment or price value provided by the Ethereum blockchain platform for a successful transaction or execution of a contract, as shown in Table 6.4. The exact gas price is calculated by the network miners, who may decline to handle a transaction if the price of Gas does not reach its mark. Therefore, all functions, computations, message calls, smart contract

creation/deployment, and storage on Ethereum Virtual Machine (EVM) require Gas to execute all of these operations, as shown in Table 6.5. When a user has no legitimate balance account, they cannot carry out any form of operation and is thus deemed invalid. In EVM, Ethers (ETH) are used to buy Gas, and users running the transactions can set their account gas limit for the particular transaction.

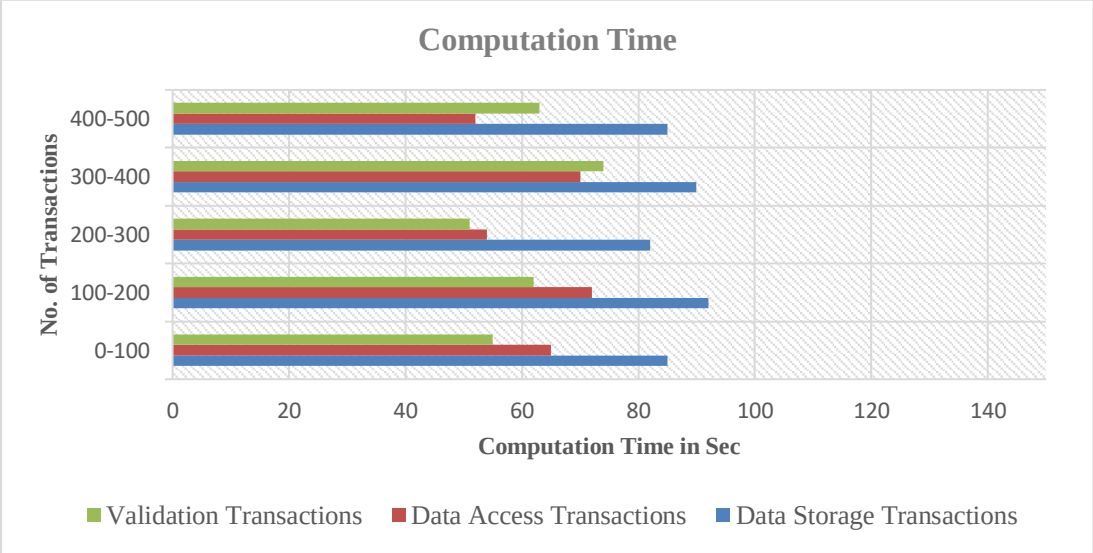


Figure 6.4: Computation Time Required for Completion of Transactions

Table 6.4: Deployment Cost of Contracts

Contract	Cost (ETH)	Cost (Dollar)
Health Token	0.0408706 ETH	$10.94
Healthcare	0.0715839 ETH	$19.16
Total Cost	0.1124545 ETH	$30.1

Table 6.5: Gas Used in Calling/Sending Functions of Smart Contract

Function	Gas (ETH)	Gas (Dollar)
Register Patient	0.002535 ETH	$0.68
Register Doctor	0.003149 ETH	$0.84
Initial Transferring of Coins	0.001997 ETH	$0.53
Patient Adding File	0.003214 ETH	$0.86
Patient Sending File to Doctor	0.007091 ETH	$1.9
Doctor Sending File to Patient	0.006174 ETH	$1.65

6.2.4.3 Comparison Analysis

This section performs the proposed platform's comparative analysis with some of the latest related work. A comparison survey is conducted to illustrate the proposed system's performance and flexibility, and the assessment findings as shown in Table 6.6.

Table 6.6: Comparative Analysis of the Proposed Application with the Existing Studies

Reference	Tokens used	Mining Required	Smart Contract	Blockchain Platform	Integrity Checking	File Storage	Access Policy
[126]	No	Yes	No	Permissioned	Yes	Database Gatekeeper	Yes
[127]	No	Yes	Yes	Permissioned	Yes	EHR DB	Yes
[128]	No	Yes	Yes	Consortium	Yes	Cloud Storage	Yes
[129]	No	No	No	Permissioned	Yes	Blockchain as storage	Yes
[130]	Yes	Yes	Yes	Customized Blockchain	Yes	Blockchain cloud	Yes
[131]	No	No	No	Permissioned	Yes	Healthcare database	Yes
Proposed Work	Yes (Tokens)	Yes	Yes	Permissioned	Yes	IPFS	Yes

The characteristics mentioned above play a crucial role in comparing the existing frameworks for this analysis. It also represents the overall blockchain platform's success and shows the importance of the proposed approach. As shown in Table 6.6, the proposed system offers a more appropriate environment for storing healthcare data as compared with existing works for the following reasons: 1) users may use the Dapp anywhere anytime by using smartphones, 2) utilized the decentralized storage (IPFS) for securely saving users files instead of static databases, 3) designed tokens for providing services to the users. It also prevents malicious activities, 4) designed smart contract to store user-related/token-related information during the communication between the users, 5) the health data of each individual can only be accessed by themselves.

6.2.5 Summary of Healthify

This work introduced a Dapp for secure authentication and access control of broad-scale health data. We have implemented the application to guarantee that patients' medical data are secured to prevent diagnostic conflicts. We designed a smart contract for authentication, access control, file sharing, and token management to obtain secure and flexible healthcare data management. Furthermore, users can validate the integrity of documents to ensure security and privacy at any time. The results, performance evaluation, security analysis, and comparison study show that our plan fulfilled the safety and storage requirements. The proposed application could easily be extended by providing more services to users in the healthcare domain.

6.3 Intrusion Detection System Using Blockchain Technology

The virtual world was built as the global infrastructure when the technology system introduced both computer technology and network technology. Today's virtual world is nearly as powerful as the real economy, placed as the basis for the corporate system. However, when the information, mainly the business information, is exchanged online, a trustworthy authority is essential to ensure the credibility of the data and the actual value that can be grouped into the real world. This type of mode is the internet's company mode now. However, these so-called trusted parties may also be likely and able to do some evil and harmful things knowingly or unknowingly, such as tracking or selling customer data for company use.

Blockchain is suggested as a prospective alternative to the above issue. Blockchain is a relatively advanced technology that allows multiple authoritative domains that do not trust each other to collaborate, cooperate, and coordinate decision-making. The advent of blockchain provides credible information management and exchange methods that can make internet transactions more real and free of third parties [133]. More and more blockchain-based applications are used to deliver business and other services that majorly affect the online business system. Blockchain provides online data transfer with non-modifiable data records, making the transfer of data more reliable. Blockchain plays a more critical role in the future as online privacy and reliability become increasingly essential. Blockchain-based systems will be the basic infrastructure that can provide people with many services [134]. Chain management is applied when there is a need for a data transfer policy to maximize the

efficiency of resource sharing. When one node in the network starts the data transfer to another node in the system, the source node often includes other nodes in the transaction to form a route. The intermediate nodes are referred to as hop in any chain architecture. As shown in Fig. 6.5, the source node aims to transfer the data to the destination node. To maximize resource utilization, one node becomes a data vendor for multiple nodes. This increases the randomness in the network, and the network becomes vulnerable to intruders.

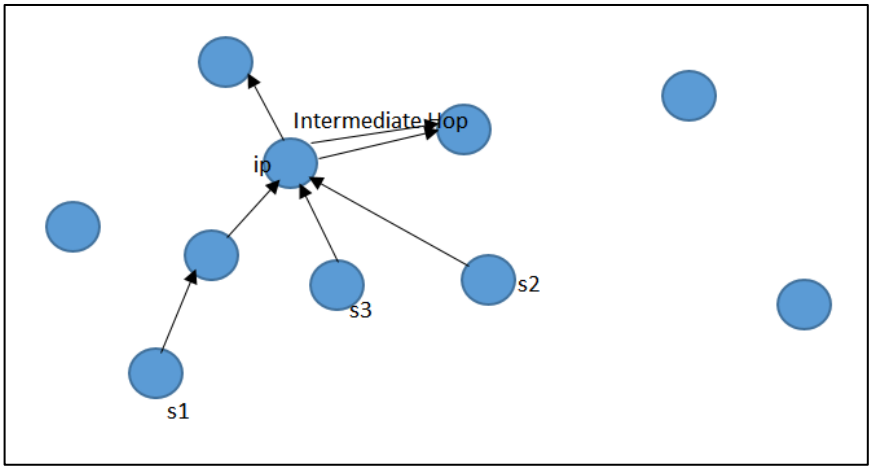


Figure 6.5: The Hop Behavior

As shown in Fig. 6.5, if the intermediate node represented by “ip" receives a lot of data packets or data elements from source nodes s1, s2, s3 … sn and so on, the intruder may also attempt to transfer the data from ”ip" which will further affect the network environment. If described practically, every network has a Network Admin (NA). Still, the NA does not aim to identify the intruder, and it would go to find the node in its system, which is intruded as a user is using AVG antivirus. The antivirus cannot do anything to the virus generator, but it prevents its network from the effects of the virus. This work focuses on this logic and prepares a Layered Voting Rule System (LVRS), described in Section 6.3.1.

6.3.1 Proposed Architecture

The proposed architecture is named LVRS, which contains three layers, a positive layer, a negative layer, and a propagation layer, as shown in Fig. 6.6. The proposed work adopts a behavioral framework of blockchain technology to identify the intruder node in the network as discussed below:

Layer 1: The nodes are deployed in the network referred to as users. Blockchain technology is utilized for deploying the user nodes. The user shares the resources from one end to another in the proposed network. Each user node stored the information in the form of a ledger. Further, ledger information is utilized by the NA for identifying the intruder node.

The voting rule is based on the users, and each user has an independent set of data. The user design creates a semi-autonomous blockchain mechanism as each user is partially affected by another user who votes for him or against him. The user's sustainability depends on the positive votes, which are again further controlled by the NA.

The demanding user is referred to as the destination, and the original resource holder is designated as the source in the proposed work. Each data transfer will also involve intermediate nodes, referred to as hop in the proposed work. Each data transfer is counted as one vote from source to destination vice versa. The intermediate nodes are also benefited if the data is transferred successfully.

Layer 2: Layer 2 is operated by the NA. The voting points are stored with the NA at the Cloud Layer. Positive Layer Repository (PLR) and Negative Layer Repository (NLR) are created from the source to the destination. Also, the power consumption in each simulative iteration of individual nodes and the total transfer is stored.

Layer 3: The third layer propagates the power consumption in combination with the positive and negative sources. This layer decides whether the user is safe for resource sharing or not. LVRS further bifurcates the propagation layer identification mechanism in subsequent steps. The first step is for the propagation mechanism, and the second step uses gradient functions, followed by linear quad architecture for data propagation. LVRS analyses the user's behavior based on the propagation data, which is generated through the overall power consumption in transactions. A unique voting rule is presented, which helps analyze the network when it is scanned for intrusion.

The workflow of the proposed architecture is described as follows:

1. The users are deployed with random locations in the network.

2. The users will share the data as a resource-sharing mechanism.

3. The input data, i.e., the data user wants to transfer or share from one end to another, can be sent only once in one simulation.

4. One data transfer will be considered as one vote to the destination.

5. Source and destinations are assumed to be immune from intrusion as the network considers the vote to affect the immunity positively.

6. The NA has a veto power to reject the vote count of any node if NA feels like the node is compromised.

7. NA uses the blockchain mechanism in three layers of processing and stores the information in the cloud.

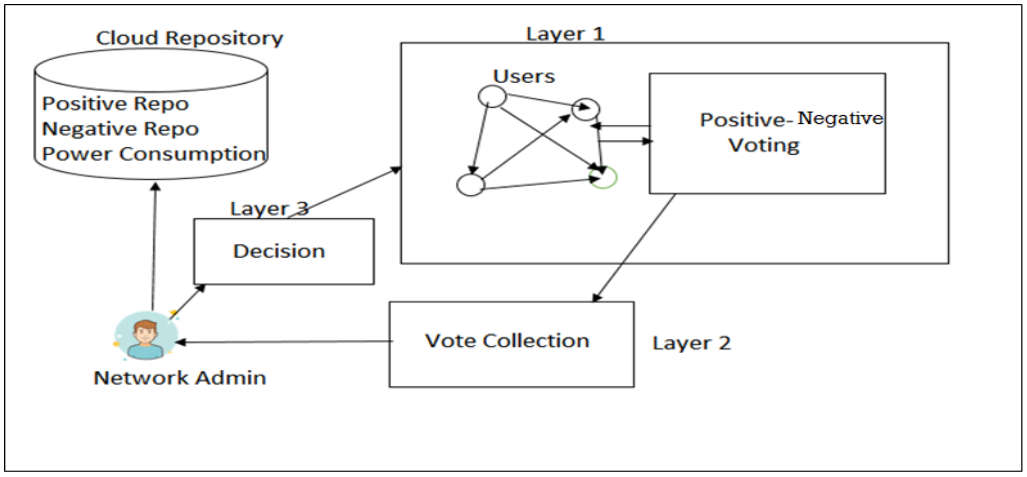
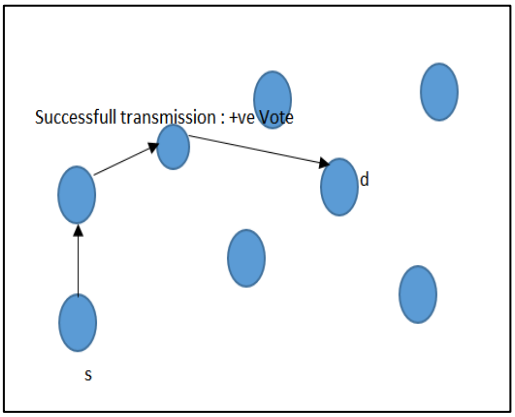


Figure 6.6: Proposed Layer Voting Rule Architecture

It is assumed that every intruded node involved in the data transfer will consume more power, but every high energy-consuming node cannot be considered intruded. The overall proposed structure is demonstrated in Fig. 6.7(a), Fig. 6.7(b), and Fig. 6.7(c) as follows:



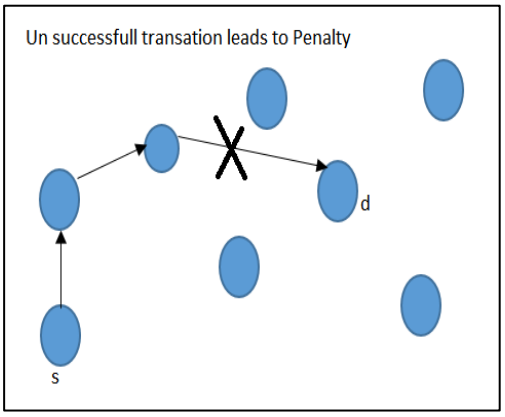


Figure 6.7(a): Positive Vote Figure 6.7(b): Penalty

As shown in Fig. 6.7(a), when a source ‘s' transfer the data to ‘d' successfully, the NA counts it as a positive vote whereas, Fig. 6.7(b) shows that if the node is not able to transfer the data

to 'd' successfully, the node suffers a penalty. It is assumed that 65% of the total transmitted data is received at the destination end. The transaction is said to be a successful transaction. The NA stores the information of the source node and the hops in blocks to rate the node between 0.50 to 1.0. The rating is done based on the total amount of actual data transfer from one end to another. If the node is involved in any other transaction, the average weight of all the previous operations and the current transaction is considered. If the transaction is not successful, i.e., if the received amount is less than 65%, the source node gets a penalty between 0.1 to 0.49, the negative weight is updated. The hops in the network also get a penalty between 0.1 to 0.20. At the time of the network scan, if the node weights more than 0.60, the node is free from the scan. A node with a negative rating of more than 0.20 is scanned twice at the scan time. The nodes are also referred to as users in the proposed model.

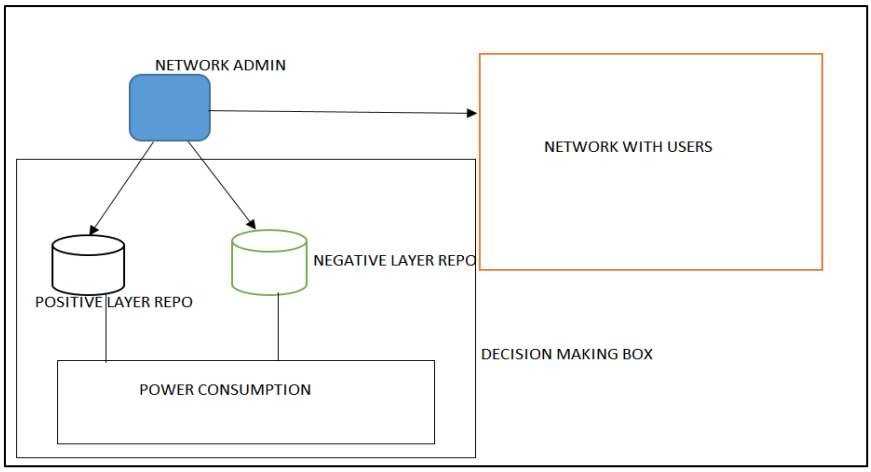


Figure 6.7(c): Decision Making of LVRS

Fig. 6.7(c) represents the decision-making architecture of LVRS, which involves Positive Layer Repo (PLR) and Negative Layer Repo (NLR), as discussed earlier. In addition to the PLR and NLR network, the admin also keeps a record of consumed power in the data transaction through each user in the network and uses it for decision making to find any intrusion in the network. The description of identification is as follows. The users are deployed randomly in the network with a random position. The ordinal measures of deployment are illustrated in Table 6.7.

The proposed model considers a power consumption model when transferring the data from one end to another. When a user transfers the data from one end to another, it will consume an P_t amount of power. Similarly, P_r is the power consumption to receive the data element. NA keeps a record of the transfer and receiving of the data elements and the users involved in

the data transfer. P_t and P_r is heterogeneous, i.e., it is different for every node. A maximum of 1000 simulations is monitored with a breakpoint after every 100 simulations; hence, the window size of identification is 100. The NA analyses the network after every 100 simulations and can use his veto power at any instance. NA uses power consumption as the primary identifying attribute. P_r and P_t has two sub-attributes. The user will consume less power under normal conditions and more power under intruded conditions.

Table 6.7: Deployment Parameters

Parameters	**Values**
Total User Count	40-60
Deployment Mode	Random
Data Type	Bits
Area of Deployment	1000×1000 m
Selection Mode	Random
Total Number of Simulations	100-1000
Deployment Tool	Anaconda
Deployment Framework	Spyder
Language Used	Python

Algorithm 6.5: Deployment of Node

Input: Number of Users

Output: Simulation of Network

1. $\boldsymbol{For_{each}}$ user in $\boldsymbol{user_{count}}$
2. $\boldsymbol{User_x = Deployment_{Mode}.Random()}$
3. $\boldsymbol{User_y = Deployment_{Mode}.Random()}$
4. Generate $\boldsymbol{Power_{Consumption_{Model}}}$
5. Destination = Generate a random destination
6. Source = Look for the resources from the destination
7. $\boldsymbol{Intermediate_{Hops}}$ = Generate Resource Carrier
8. Initialize Network
9. Start Simulation and Data Transfer

Algorithm 6.5 deploys the user with random x and y-axis in the network. Every user contains some resource that is sharable in the network. A power consumption model is also deployed, clarifying the total consumption of power when the user receives a data packet and the total consumed power when a node receives a data packet, as presented in Algorithm 6.6. The data is transferred through intermediate hops, which have any vacant slot to transfer the data. The data are documents that are available to the user. The resources are not editable; only the data owner has the authority to edit the document's data. The rest of the users have read-only permission.

Algorithm 6.6: Identification of Power Consumption

Input: Power Consumption Parameters

Output: Total Consumed Power

1. $\boldsymbol{For_{each}}$ established connection between User1 to User 2
2. $\boldsymbol{Consumed_{Power}} = \sum_{k=0}^{n} \boldsymbol{P_t} + \boldsymbol{P_r}$
3. $\boldsymbol{Prevention_{Input_{Architecture}}} = \boldsymbol{Consumed_{Power}}$
4. Initialize Chain Mechanism with $\boldsymbol{Prevention_{Input_{Architecture}}}$
5. Propagate Chain with Linear and Quad Propagation Model
6. Linear Model follows: $\boldsymbol{ax + b = 0}$
7. Quad Model follows: $\boldsymbol{ax^2 + bx + c = 0}$
8. $\boldsymbol{Chain_{Weight}} = \boldsymbol{Model_{Weight}} + \boldsymbol{Input_{Architecture}}$
9. $\boldsymbol{Chain_{Satisfying_{Element}}} = \boldsymbol{Average_{Gradient}} + \boldsymbol{window_{size}}$
10. $\boldsymbol{Average_{Gradient}} = \sum_{i}^{window_{size}} \boldsymbol{Consumed_{Power}}$

6.3.2 Results

The proposed architecture performance is evaluated using two parameters: throughput and power consumption. The throughput measures the performance in terms of total packets received by the proposed architecture per time frame. In contrast, the power consumption parameter calculates the total power consumed by the proposed scheme in each window. Every result is evaluated with the window defined in Section 6.3.1. Fig. 6.8 represents the throughput of the proposed model with a comparison of the model without blockchain. The

throughput of the proposed architecture is high as compared to the network model with no intrusion model. The proposed work has adopted the new intrusion model, which is adaptive, and hence, the chances of network intrusion are quite low. The throughput is tested for a maximum of 500 simulation iterations. The average throughput for proposed architecture is noticed to be 13000 whereas, for the generalized network architecture, it stands at 8700. A noticeable difference of (13000-8700)/1300= 33% is noticed.

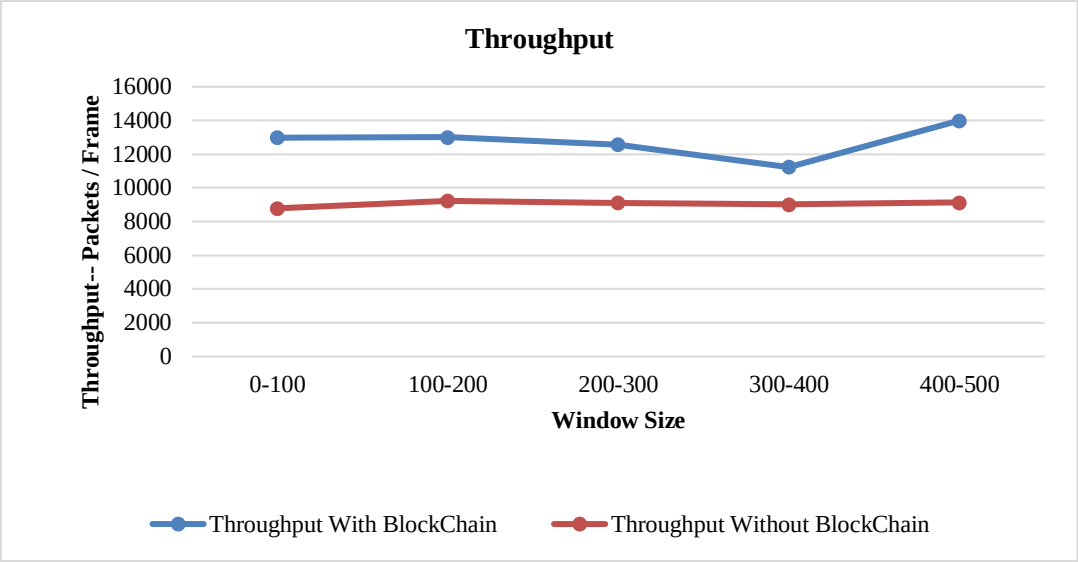


Figure 6.8: Throughput of the Proposed Architecture

The proposed architecture is also evaluated for power consumption, as shown in Fig. 6.9. A total of 500 simulations are tested based on power consumption. A noticeable difference in power consumption is observed. The proposed architecture's power consumption is low compared to the network model with no intrusion model.

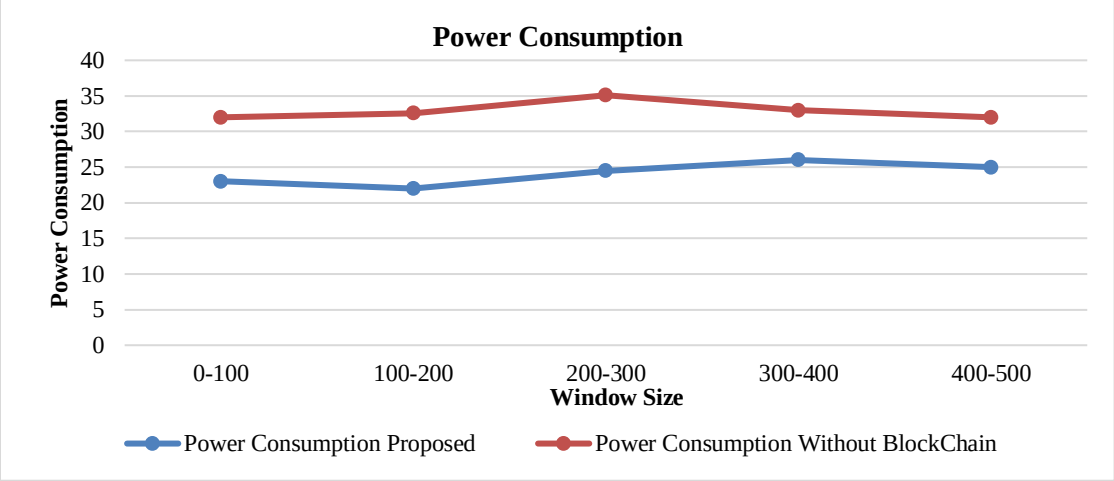


Figure 6.9: Power Consumption of the Proposed Architecture

6.3.3 Summary of LVRS

This work presented intrusion behavior analysis architecture through the adaptive blockchain. Power consumption is kept as the critical aspect of the intrusion analysis. The proposed architecture is named LVRS, which contains different layers for different purposes. A positive layer, a negative layer, and a propagation layer are presented in the proposed work, which utilizes power consumption. LVRS further bifurcates the identification mechanism in subsequent steps. The first step is for the propagation mechanism, and the second step uses gradient functions, followed by linear quad architecture for data propagation. LVRS analyses the user's behavior based on the propagation data, which is generated through the overall power consumption in transactions. A unique voting rule is presented in this work, which helps analyze the network when it is scanned for intrusion. A window size of 100 simulations is used to apply breakpoints and analyze the data through breakpoints. A total of 500 simulations are tested based on Throughput and Power Consumption. A noticeable difference of more than 30% is observed in both throughput and power consumption.

CHAPTER 7

CONCLUSION AND FUTURE WORK

This chapter summarizes the work done in the thesis, presents contributions, and highlights the future directions.

This research work proposed decentralized and secure blockchain-based cloud storage architecture with distributed key management, integrity checking, access control, and user revocation process to ensure privacy in cloud data. The proposed blockchain-based cloud system implements a CP-ABE algorithm with bilinear pairing to provide a privacy-preserving environment. The implemented algorithm utilizes the distributed approach using blockchain technology to maintain keys without involving any trusted authority. The performance of the proposed architecture has compared with the state-of-the-art techniques and has given significant results for different parameters such as computation time, throughput, latency, transmission time, encoding time, decoding time, key generation time, etc., for all techniques being compared. The experimental results, security analysis, and comparative analysis demonstrate the effectiveness and efficiency of the proposed system.

7.1 Summary of Work Done in the Thesis

This section summarizes the thesis to resolve the challenges and issues present in the centralized cloud storage system using blockchain technology.

Firstly, to address the privacy and security challenges of the cloud storage system, this research work proposed novel blockchain architecture for the cloud storage system to achieve security features.

Secondly, to eliminate the need for trusted third parties and single point of failure issues, the proposed blockchain-based decentralization architecture provides a promising solution using

a peer-to-peer network that provides equal validation rights to participants and ensures transparency.

Thirdly, to improve the security of the cloud storage system, the proposed architecture deploys various cryptographic algorithms to provide a key generation mechanism, achieve confidentiality, perform data access control mechanisms, and ensure the integrity of the cloud data. The blockchain network records all transactions that are cryptographically secured and stored in the form of hash values. Furthermore, blockchain transactions are signed by the network participants so that the user interaction with the cloud storage system remains confidential. The blockchain structure manages the key related and user access policy details in a distributed manner and provides a more robust environment. The blockchain network keeps Meta details of user data and tracks all access and validation records.

Lastly, to optimize the resources and minimize the execution time, the proposed architecture deploys a Honeybee optimization algorithm to optimize the resource utilization on the cloud storage system and minimize execution and response time.

7.2 Contributions of the Research

This research work describes the solution approach utilized to address the challenges of cloud storage systems. We have proposed the blockchain-based distributed architecture for the cloud storage system to provide the following four features for organizations or individual end-users:

- **Distributed key generation process using CP-ABE algorithm**
- **Maintain the integrity of stored data using the Merkle root concept**
- **Optimization of cloud storage system**
- **Provide access control and revocation mechanism**

First, this work proposed decentralized blockchain-based cloud architecture. The proposed architecture implements the CP-ABE algorithm using smart contract functionality to provide a complete distributed approach to manage keys, access policies, and the hash of the cloud data. Blockchain technology stores all information in a decentralized manner such that no one alters the data once it has been stored in the form of a block in the blockchain structure.

Second, this research work ensures the integrity of stored cloud data using the Merkle root concept. The proposed architecture divides the files into multiple shards. The Meta details of bifurcated file parts in the form of hash are organized in a Merkle Tree structure to obtain Merkle root for the user data. The root of the Merkle tree is utilized for the integrity verification of the cloud data.

Third, the proposed work implements the optimization algorithm on the cloud storage system to minimize the transaction response and execution time. The effective utilization of cloud resources reduces the storage and time requirement. Various users or authorities access the cloud storage system using different operations such as uploading data, validating data, returning the result, etc. These operations require time for execution and generating response depending on the application requirements. Therefore, reducing the time required to perform these operations optimize the cloud system.

Last, the proposed architecture further enhanced to achieve the access control mechanism by implementing the access policy in a tree structure and attaining fine-grained access control at the system level. This research also uses the CP-ABE algorithm's re-encryption approach to update attribute group keys and deploy immediate user revocation processes.

7.3 Future Work

We plan to deploy the proposed architecture in the multimedia system for future work to provide reliable, safe, and dynamic management of multimedia data and provide key management, access control, and integrity checking features accordingly.

For future work, the proposed architecture functionality will be further extended to provide safe deletion of cloud data with a blockchain-based payment system for utilizing the cloud services.

Publications Related to the Thesis

Papers Published in International Journals:

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "Blockchain Technology for Cloud Storage: A Systematic Literature Review," *ACM Computing Surveys*, vol. 53, no. 4, 2020. **(SCI, IF: 10.2)**
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "Blockchain-based Decentralized Architecture for Cloud Storage System," *Journal of Information Security and Applications*, vol. 62, pp. 2214-2126, 2021. **(SCIE, IF: 3.8)**
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "A Review of Blockchain-based Applications and Challenges," *Wireless Personal Communications: An International Journal*, Springer, vol. 123, pp. 1201-1243, 2022. **(SCIE, IF: 1.6)**
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "Blockchain-based Cloud Storage System with CP-ABE based Access Control and Revocation Process," *Journal of Supercomputing*, Springer, vol. 78, pp. 7700-7728, 2022. **(SCI, IF: 2.4)**
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "A Review of Smart Contract-based Platforms, Applications, and Challenges," *Cluster Computing*, Springer, pp. 1-27, 2022. **(SCIE, IF: 1.8)**

Papers Published in International Conferences:

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "Blockchain-based Integrity Protection System for Cloud Storage," In *Proceedings of the 4th Technology Innovation Management and Engineering Science International Conference* (TIMES-iCON), Bangkok, Thailand, 11th-13th December, 2019, pp. 1-5.
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, "A Preventive Intrusion Detection Architecture using Adaptive Blockchain Method," In *Proceedings of the International Conference for Big Data, Machine Learning and Applications* (BigDML 2019), NIT Silchar, Assam, 16th-19th December, 2019, pp. 25-35.

Published Book Chapters:

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, “Blockchain-based Secure Healthcare Application," In: edited book entitled Blockchain in Digital Healthcare, Taylor & Francis Group, pp. 35-54, 2021.
- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, “Healthify: A Blockchain-based Distributed Application for Healthcare," In: edited book entitled Application of Blockchain Technology in Healthcare, Springer Book Series "Studies in Big Data”, pp. 171-198, 2021.

Paper Communicated in International Journal:

- Pratima Sharma, Rajni Jindal, and Malaya Dutta Borah, “A Comparative Analysis of Consensus Algorithms for Decentralized Storage Systems,” *IT Professional*, IEEE. (Communicated)

References

[1] B. Marr, "How Much Data Do We Create Every Day? The MindBlowing Stats Everyone Should Read," *Forbes*, 2020.

[2] H. Song, J. Li and H. Li, "A cloud secure storage mechanism based on data dispersion and encryption," *IEEE Access*, vol. 9, pp. 63745-63751, 2021.

[3]A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *Journal of Network and Computer Applications*, vol. 79, pp. 88-115, 2017.

[4] Y. Shin, D. Koo, and J. Hur, "A survey of secure data deduplication schemes for cloud storage systems," *ACM Computing Surveys*, vol. 49, no. 4, pp. 1-38, 2017.

[5] M. Du, Q. Wang, M. He, and J. Weng, "Privacy-preserving indexing and query processing for secure dynamic cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2320–2332, 2018.

[6] Y. Zhang, X. Chen, J. Li, D. S Wong, H. Li, and I. You, "Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing," *Information Sciences*, vol. 379, pp.42–61, 2017.

[7] Y. Zhang, D. Zheng, and R. H Deng, "Security and privacy in smart health: Efficient policy hiding attribute-based access control," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130-2145, 2018.

[8] N. Kaaniche, and M. Laurent, "Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms," *Computer Communications*, vol. 111, pp. 120–141, 2017.

[9] Y. Li, K. Gai, L. Qiu, M. Qiu, and H. Zhao, "Intelligent cryptography approach for secure distributed big data storage in cloud computing," *Information Sciences*, vol. 387, pp. 103–115, 2017.

[10] J. Shen, Z. Gui, S. Ji, J. Shen, H. Tan, and Y. Tang, "Cloud aided lightweight certificate less authentication protocol with anonymity for wireless body area networks," *Journal of Network and Computer Applications*, vol. 106, pp. 117–123, 2018.

[11] J. Shen, C. Wang, T. Li, X. Chen, X. Huang, and Z. H. Zhan, “Secure data uploading scheme for a smart home system,” *Information Sciences*, vol. 453, pp. 186-197, 2018.

[12] N. A. Kofahi and A. R. Al-Rabadi, “Identifying the top threats in cloud computing and its suggested solutions: A survey,” *Networks*, vol. 6, no. 1, pp. 1-13, 2018.

[13] R. Lyengar, “Apple to strengthen security after iCloud nude celebrity photos leak." http://time.com/3271667/apple-jennifer-lawrence-icloud-leak660 security/, 2014. Accessed September 4, 2020

[14] D. Marudhadevi, V. N. Dhatchayani and V. S. S. Sriram, “A Trust Evaluation Model for Cloud Computing Using Service Level Agreement,” *The Computer Journal*, vol. 58, no. 10, pp. 2225-2232, 2015.

[15] Y. Zhang, C. Xu, X. Lin, and X. S. Shen, “Blockchain-based public integrity verification for cloud storage against procrastinating auditors,” *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 923-937, 2021.

[16] Satoshi Nakamoto, “Bitcoin: A peer-to-peer electronic cash system”. Retrieved from https: //bitcoin.org/bitcoin.pdf, 2008.

[17] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, “Untangling blockchain: A data processing view of blockchain systems,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366–1385, 2018.

[18] J. Liu and Z. Liu, “A survey on security verification of blockchain smart contracts,” *IEEE Access*, vol. 7, pp. 77894-77904, 2019.

[19] J. Eberhardt and S. Tai, “On or off the blockchain? Insights on off-chaining computation and data,” In: Service-Oriented and Cloud Computing (ESOCC’17), F. De Paoli, S. Schulte, and Johnsen E. Broch (Eds.), Lecture Notes in Computer Science, Springer, Cham, vol. 10465, 2017.

[20] AWS Blockchain. [Online]. Available: https://github.com/awssamples/non-profit-blockchain. Accessed March 4, 2021

[21] L. Ferretti, M. Marchetti, M. Andreolini, and M. Colajanni, “A symmetric cryptographic scheme for data integrity verification in cloud databases,” *Information Sciences*, vol. 422, pp. 497-515, 2018.

[22] J. Li, J. Li, X. Chen, C. Jia, and W. Lou, “Identity based encryption with outsourced revocation in cloud computing,” *IEEE Transactions on Computers*, vol. 64, pp. 425-437, 2015.

[23] B. Feng, X. Ma, C. Guo, H. Shi, Z. Fu, and T. Qiu, “An efficient protocol with bidirectional verification for storage security in cloud computing,” *IEEE Access*, vol. 4, pp. 7899-7911, 2016.

[24] C. M. Yu, S. P. Gochhayat, M. Conti, and C. S. Lu, “Privacy-aware data deduplication for side channel in cloud storage,” *IEEE Transactions on Computers*, vol. 4, pp. 1-13, 2018.

[25] Y. Yan, L. Wu, G. Gao, H. Wang, and W. Xu, “A dynamic integrity verification scheme of cloud storage data based on lattice and bloom filter,” *Journal of Information Security and Applications*, vol. 39, pp. 10-18, 2018.

[26] G. Xu, M. Lai, J. Li, L. Sun, and X. Shi, “A generic integrity verification algorithm of version files for cloud deduplication data storage,” *EURASIP Journal on Information Security*, vol. 12, pp. 1-15, 2018.

[27] H. Jin, K. Zhou, H. Jiang, D. Lei, R. Wei, and C. Li, “Full integrity and freshness for cloud data,” *Future Generation Computer Systems*, vol. 80, pp. 640-652, 2018.

[28] R. Saxena, and S. Dey, “Cloud audit: A data integrity verification approach for cloud computing,” *Procedia Computer Science*, vol. 89, pp. 142-151, 2016.

[29] J. Mao, Y. Zhang, P. Li, T. Li, Q. Wu, and J. Liu, “A position aware merkle tree for dynamic cloud data integrity verification,” *Soft Computing*, vol. 21, pp. 2151- 2164, 2017.

[30] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu, and X. Zhang, “Efficient public verification of data integrity for cloud storage systems from indistinguishability obfuscation,” *IEEE Transactions on Information Forensics and Security*, vol. 12, pp. 676-688, 2017.

[31] T. Jiang, X. Chen, and J. Ma, “Public integrity auditing for shared dynamic cloud data with group user revocation,” *IEEE Transactions on Computers*, vol. 65, pp. 2363-2373, 2017.

[32] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 608-619, 1 May-June 2020.

[33] J. Li, N. Chen, and Y. Zhang, “Extended file hierarchy access control scheme with attribute-based encryption in cloud computing,” *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 983-993, 2021.

[34] S. Wilkinson and J. Lowry, “Metadisk: Blockchain-based decentralized file storage application,” *Technical Report*, vol. 14, pp. 573–580, 2014.

[35] I. Sukhodolskiy, and S. Zapechnikov, “A blockchain-based access control system for cloud storage, “ In *Proceedings of the IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering* (EIConRus), Moscow, 2018, pp. 1575-1578.

[36] H. Qiu, X. Wu, S. Zhang, V. C. M. Leung and W. Cai, "ChainIDE: A cloud-based integrated development environment for cross-blockchain smart contracts," In *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science* (CloudCom), 2019, pp. 317-319.

[37] D. Li, R. Du, Y. Fu, and M. H. Au, “Meta-Key: A secure data-sharing protocol under blockchain-based decentralized storage architecture,” *IEEE Networking Letters*, vol. 1, pp. 30-33, 2019.

[38] H. Li, H. Tian, F. Zhang, and J. He, “Blockchain-based searchable symmetric encryption scheme,” *Computers and Electric Engineering*, vol. 73, pp. 32-45, 2019.

[39] Y. Zhang, R. H. Deng, X. Liu, and D. Zheng, “Blockchain-based efficient and robust fair payment for outsourcing services in cloud computing,” *Information Science*, vol. 262, pp. 262-277, 2018.

[40] J. H. Lee, “BIDaaS: Blockchain based ID as a service,” Special Section on Intelligent Systems for the Internet of Things, *IEEE Access*, vol. 6, pp. 2274-2278, 2018.

[41] C. Yang, X. Chen, and Y. Xiang, “Blockchain-based publicly verifiable data deletion scheme for cloud storage,” *Journal of Network and Computer Applications*, vol. 103, pp. 185-193, 2017.

[42] P. Wei, D. Wang, Y. Zhao, S. K. S. Tyagi, and N. Kumar, “Blockchain data-based cloud data integrity protection mechanism,” *Future Generation Computer Systems*, vol. 102, pp. 902-911, 2020.

[43] E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa and P. Samarati, "Securing resources in decentralized cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 286-298, 2020.

[44] Y. Zhang, C. Xu, X. Lin, and X. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 923-937, 2021.

[45] S. Wang, X. Wang, and Y. Zhang, "A secure cloud storage framework with access control based on blockchain," *IEEE Access*, vol. 7, pp. 112713-112725, 2019.

[46] R. Awadallah, A. Samsudin, J. S. Teh, and M. Almazrooie, "An integrated architecture for maintaining security in cloud computing based on blockchain," *IEEE Access*, vol. 9, pp. 69513-69526, 2021.

[47] A. Ali, A. Khan, M. Ahmed, and G. Jeon, "BCALS: Blockchain-based secure log management system for cloud computing," *Transactions on Emerging Telecommunications Technologies*, Wiley, pp. 1-20, 2021.

[48] J. Liebenau, and S. M. Elaluf-Calderwood, "Blockchain innovation beyond bitcoin and banking," 2016. Available at SSRN: https://ssrn.com/abstract=2749890

[49] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Applied Innovation*, vol. 2, pp. 6–10, 2016.

[50] L. S. Sankar, M. Sindhu, and M. Sethumadhavan, "Survey of consensus protocols on blockchain applications," In *Proceeding of the 4th International Conference on Advanced Computing and Communication Systems* (ICACCS), 2017, pp. 1–5.

[51] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," In *Proceedings of the IEEE International Congress on Big Data* (BigData'17), 2017, pp. 557–564.

[52] B. Putz and G. Pernul, "Detecting blockchain security threats," In *Proceedings of the IEEE International Conference on Blockchain*, 2020, pp. 313-320.

[53] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. Anyigor Ogah, and Z.i Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet Things of Journal*, vol. 4, no. 6, pp. 1832–1843, 2017.

[54] C. A. Vyas and M. Lunagaria, “Security concerns and issues for bitcoin,” In *Proceedings of the National Conference cum Workshop on Bioinformatics and Computational Biology* (NCWBCB’14), 2014.

[55] A. Wright, and P. D. Filippi, “Decentralized blockchain technology and the rise of lex cryptographic,” 2015. Available at SSRN: https://ssrn.com/abstract=2580664 or http://dx.doi.org/10.2139/ssrn.2580664

[56] Lin, Iuon-Chang, and T. C. Liao, “A survey of blockchain security issues and challenges,” *International Journal of Network Security*, vol. 19, no. 5, pp. 653- 659, 2017.

[57] A. L. Selvakumar and C. S. Ganadhas, "The Evaluation Report of SHA-256 Crypt Analysis Hash Function," In *Proceedings of the International Conference on Communication Software and Networks*, 2009, pp. 588-592.

[58] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, “A survey on the security of blockchain systems,” *Future Generation Computer Systems*, vol. 107, pp. 841-853, 2020.

[59] J. H. Mosakheil, “Security threats classification in blockchains,” *Culminating Projects in Information Assurance*, vol. 48, 2018.

[60] J. Bethencourt, A. Sahai, and B. Waters, ‘‘Ciphertext-policy attribute-based encryption,’’ In *Proceeding of the IEEE Symposium on Security and Privacy* (SP), 2008, pp. 321–334.

[61] S. Nick, “The idea of smart contracts,” 1997. Retrieved from http://www.fon.hum.uva.nl/rob/ Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net /idea.html

[62] S. Nick, “Formalizing and securing relationship on public networks,” 1997, http://firstmonday.org/ojs/ index.php/fm/article/view/548/469 Accessed April, 2020.

[63] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, “SOK: Research perspectives and challenges for bitcoin and cryptocurrencies,” In *Proceedings of the IEEE Symposium on Security and Privacy* (SP), 2015, pp. 104–121.

[64] M. C. K. Khalilov, and A. Levi, “A survey on anonymity and privacy in bitcoin-like digital cash systems,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2543-2585, 2018.

[65] E. Azhir, N. J. Navimipour, M. Hosseinzadeh, A. Sharifi, and A. Darwesh, “Query optimization mechanisms in the cloud environments: A systematic study,” *International Journal of Communication Systems*, vol. 32, no. 8, pp. 1-23, 2019.

[66] A. Singh and K. Chatterjee, “Cloud security issues and challenges: A survey,” *Journal of Network and Computer Applications*, vol. 79, pp. 88– 115, 2017.

[67] M. Du, Q. Wang, M. He, and J. Weng, “Privacy-preserving indexing and query processing for secure dynamic cloud storage*,” IEEE Transactions on Information Forensics and Security*, vol. 13, no. 9, pp. 2320–2332, 2018.

[68] Yinghui Zhang, Xiaofeng Chen, Jin Li, Duncan S Wong, Hui Li, and Ilsun You, “Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing,” *Information Sciences*, vol. 379, pp. 42-61, 2017.

[69] Yinghui Zhang, Dong Zheng, and Robert H Deng, “Security and privacy in smart health: Efficient policy-hiding attribute-based access control,” *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130-2145, 2018.

[70] N. Kaaniche and M. Laurent, “Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms,” *Computer Communications*, vol. 111, pp. 120–141, 2017.

[71] Y. Li, K. Gai, L. Qiu, M. Qiu, and H. Zhao, “Intelligent cryptography approach for secure distributed big data storage in cloud computing,” *Information Sciences*, vol. 387, pp. 103–115, 2017.

[72] Guy Zyskind, Oz Nathan, et al. “Decentralizing privacy: Using blockchain to protect personal data,” In *Proceeding of the IEEE Security and Privacy Workshops* (SPW), pp. 180–184, 2015.

[73] J. Li, J. Wu, and L. Chen, "Block-Secure: Blockchain-based scheme for secure P2P cloud storage,” *Information Sciences*, vol. 465, pp. 219-231, 2018.

[74] S. Wang, X. Wang, and Y. Zhang, “A secure cloud storage framework with access

control based on blockchain," *IEEE Access*, vol. 7, pp. 112713-112725, 2019.

[75] H. Yu, and Z. Yang, "Decentralized and smart public auditing for cloud Storage," In *Proceeding of the IEEE 9th International Conference on Software Engineering and Service Science* (ICSESS), 2018, pp. 491- 494.

[76] J. Li, J. Wu, L. Chen, and J. Li, "Deduplication with blockchain for secure cloud storage," In *Proceeding of the CCF Conference on Big Data*, Springer, 2018, pp. 558-570.

[77] J. P. Cruz, Y. Kaji, and N. Yanai, "RBAC-SC: Role-based access control using smart contract," *IEEE Access*, vol. 6, pp. 12240-12251, 2018.

[78] S. Wang, X. Wang, and Y. Zhang, "A secure cloud storage framework with access control based on blockchain," *IEEE Access*, vol. 7, pp. 112713-112725, 2019.

[79] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38437-38450, 2018.

[80] B. Liu, X. L. Yu, S. Chen, X. Xu and L. Zhu, "Blockchain based data integrity service framework for IoT Data," In *Proceeding of the IEEE International Conference on Web Services* (ICWS), Honolulu, HI, USA, 2017, pp. 468-475.

[81] H. Wang, X. Wang, A. S. Xiao, and Z. Zhou, "Blockchain-based public auditing scheme for shared data," In *Proceeding of the International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, Springer, Cham, 2019, pp. 197-206.

[82] Y. Zhang, C. Xu, J. Ni, H. Li, and X. S. Shen, "Blockchain-assisted public-key encryption with keyword search against keyword guessing attacks for cloud storage," *IEEE Transactions on Cloud Computing*, vol. 9, no. 4, pp. 1335-1348, 2021.

[83] H. G. Do, and W. K. Ng, "Blockchain-based system for secure data storage with private keyword search," In *Proceeding of the IEEE World Congress on Services* (SERVICES), 2017, pp. 90-93.

[84] W. Hashem, H. Nashaat, and R. Rizk, "Honey bee based load balancing in cloud computing," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 12, pp. 5694-5711, 2017.

[85] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, “Cloud storage as the infrastructure of cloud computing,” In *Proceedings of the 2010 International Conference on Intelligent Computing And Cognitive Informatics*, pp. 380–383, 2010.

[86] A. Sahai, and B. Waters, “Fuzzy identity-based encryption,” In: Cramer R. (eds) Advances in Cryptology, EUROCRYPT 2005, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol. 3494, 2005.

[87] F. Guo, Y. Mu, W. Susilo, D. S. Wong, and V. Varadharajan, “CP-ABE with constant-size keys for lightweight devices,” *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 5, pp. 763–771, 2014.

[88] V. Koppula, and B. Waters, “Realizing chosen-ciphertext security generically in attribute-based encryption and predicate encryption,” In: Boldyreva A., Micciancio D. (eds) Advances in Cryptology – CRYPTO 2019, Lecture Notes in Computer Science, Springer, Cham, vol. 11693, pp. 671–700, 2019.

[89] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, “Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption,” In: Gilbert H. (eds) Advances in Cryptology – EUROCRYPT 2010, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol. 6110, pp. 62–91, 2010.

[90] K. Gai, J. Guo, L. Zhu, and S. Yu, "Blockchain meets cloud computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2009-2030, 2020.

[91] W. Ma, J. Ma, Q. Zhang, H. Xue, Y. Li, X. Dang, M. Zhao, J. Zhang, C. Han, and J. Wu, “Attribute revocable data sharing scheme based on blockchain and CP-ABE,” In *Proceedings of the 4th International Conference on Computer Science and Application Engineering* (CSAE 2020), Association for Computing Machinery, New York, NY, USA, 2020, pp. 1–7.

[92] J. Ning, Z. Cao, X. Dong, K. Liang, L. Wei, and K. -K. R. Choo, "CryptCloud+: Secure and expressive data access control for cloud storage," *IEEE Transactions on Services Computing*, vol. 14, no. 1, pp. 111-124, 2021.

[93] Z. Wu, Y. Zhang, and E. Xu, “Multi-authority revocable access control method based on CP-ABE in NDN,” *Future Internet*, vol. 12, no. 1, pp. 1-15, 2020.

[94] K. Fan, J. Wang, X. Wang, and Y. Yang, “Proxy-assisted access control scheme of cloud

data for smart cities," *Personal and Ubiquitous Computing*, vo. 21, no. 5, pp. 937-947, 2017.

[95] S. Wang, X. Wang and Y. Zhang, "A secure cloud storage framework with access control based on blockchain," *IEEE Access*, vol. 7, pp. 112713-112725, 2019.

[96] Saini, Q. Zhu, N. Singh, Y. Xiang, L. Gao and Y. Zhang, "A smart-contract-based access control framework for cloud smart healthcare system," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5914-5925, 2021.

[97] Patra, T. Schneider, A. Suresh, and H. Yalame, "ABY2.0: Improved mixed-protocol secure two-party computation," *Cryptology ePrint Archive*, Report, 2020.

[98] R. Buyya, R. Ranjan and R. N. Calheiros, "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities," In *Proceedings of the International Conference on High Performance Computing & Simulation*, 2009, pp. 1-11.

[99] N. Attrapadung and H. Imai, "Attribute-based encryption supporting direct/indirect revocation modes," In: Parker M.G. (eds) Cryptography and Coding, IMACC, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol. 5921, pp. 278-300, 2009.

[100] N. Attrapadung, B. Libert, and E. de Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," In: Catalano D., Fazio N., Gennaro R., Nicolosi A. (eds) Public Key Cryptography – PKC 2011, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol. 6571, pp. 90-108, 2011.

[101] P. Datta, R. Dutta, and S. Mukhopadhyay, "General circuit realizing compact revocable attribute-based encryption from multilinear maps," In: ISC, vol. 9290 of LNCS, Springer, pp. 336–354, 2015.

[102] P. Datta, R. Dutta, and S. Mukhopadhyay, "Adaptively secure unrestricted attribute-based encryption with subset difference revocation in bilinear groups of prime order," In: Pointcheval D., Nitaj A., Rachidi T. (eds) Progress in Cryptology – AFRICACRYPT, Lecture Notes in Computer Science, Springer, vol. 9646, pp. 325–345, 2016.

[103] Z. Liu and D. S. Wong, "Practical ciphertext-policy attribute-based encryption: Traitor tracing, revocation, and large universe," *The Computer Journal*, vol. 59, no. 7, pp. 983-1004, 2016.

[104] J. M. G. Nieto, M. Manulis, and D. Sun, “Fully private revocable predicate encryption,” In: Susilo W., Mu Y., Seberry J. (eds) Information Security and Privacy, ACISP, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol. 7372, pp. 350–363, 2012.

[105] P. Wang, D. Feng, and L. Zhang, "Towards attribute revocation in key-policy attribute-based encryption," In: Lin D., Tsudik G., Wang X. (eds) Cryptology and Network Security, CANS, Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, vol. 7092, pp. 272–291, 2011.

[106] A. Maxmen. “AI researchers embrace bitcoin technology to share medical data,” *Nature*, vol. 555, pp. 293-294, 2018.

[107] W. Cai, Z. Wang, J. B. Ernst, Z. Hong, C. Feng, and V. C. M. Leung, “Decentralized applications: the blockchain-empowered software system,” *IEEE Access*, vol. 6, pp. 53019–53033, 2018.

[108] Z. B. Herd, “Enlist blockchain to boost conservation,” *Nature*, vol. 548, pp. 523–523, 2017.

[109] J. H. Lee and M. Pilkington, “How the blockchain revolution will reshape the consumer electronics industry [future directions],” *IEEE Consumer Electronics Magazine*, vol. 6, no. 3, pp. 19–23, 2017.

[110] I. Konstantinidis, G. Siaminos, C. Timplalexis, P. Zervas, V. Peristeras, and S. Decker, “Blockchain for business applications: A systematic literature review,” In *Proceedings of the International Conference on Business Information Systems*, Springer, Cham, 2018, pp. 384–399.

[111] Q. E. Abbas, and J. S. Bong, “A survey of blockchain and its applications,” In *Proceedings of the International Conference on Artificial Intelligence in Information and Communication* (ICAIIC’19), 2019, pp. 1–3.

[112] S. Khezr, M. Moniruzzaman, A. Yassine, and R. Benlamri. “Blockchain technology in healthcare: A comprehensive review and directions for future research,” *Applied Sciences*, vol. 9, pp. 1–28, 2019.

[113] A. A. Siyal, A. Z. Junejo, M. Zawish, K. Ahmed, A. Khalil, and G. Soursou, "Applications of blockchain technology in medicine and healthcare: Challenges and future perspectives," *Cryptography*, vol. 3, no. 1, 1–16, 2019.

[114] R. Zhang, R. Xue, and L. Liu, "Security and privacy issues on blockchain," *ACM Computing Surveys*, vol. 52, no. 3, pp. 1–35, 2019.

[115] J. Kołodziej, A. Wilczyński, D. F. Cerero, and A. F. Montes. "Blockchain secure cloud: A new generation integrated cloud and blockchain platforms-general concepts and challenges," *European Cybersecurity Journal*, vol. 8, no. 2, pp. 28–34, 2018.

[116] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu, "Survey: Sharding in blockchains," *IEEE Access*, vol. 8, pp. 14155–14181, 2019.

[117] S. Xie, Z. Zheng, W. Chen, J. Wu, H. N. Dai, and M. Imran, "Blockchain for cloud exchange: A survey," *Computers & Electrical Engineering Journal*, vol. 81, pp. 1–20, 2019.

[118] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 2, pp. 1–22, 2018.

[119] L. Griebel, H. U. Prokosch, F. Köpcke, D. Toddenroth, J. Christoph, I. Leb, I. Engel, and M. Sedlmayr, "A scoping review of cloud computing in healthcare," *BMC Medical Informatics and Decision Making*, vol. 15, no. 1, pp. 1-16, 2015.

[120] E. Jamoom, N. Yang, and E. Hing, "Adoption of certified electronic health record systems and electronic information sharing in physician offices: United States, 2013 and 2014," US Department of Health and Human Services, Centers for Disease Control and Prevention, National Center for Health Statistics, NCHS Data Brief, pp. 1-8, 2016.

[121] A. Bahga, and V. K. Madisetti, "A cloud-based approach for interoperable electronic health records (EHRs)," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 5, pp. 894–906, 2013.

[122] G. Zangara, P. P. Corso, F. Cangemi, F. Millonzi, F. Collova, and A. Scarlatella, "A cloud-based architecture to support electronic health record," *Studies in Health Technology and Informatics*, vol. 207, pp. 380–389, 2014.

[123] V. Patel, "A framework for secure and decentralized sharing of medical imaging data via blockchain consensus," *Health Informatics Journal*, vol. 25, no. 4, pp. 1398–1411, 2018.

[124] A. Juneja, and M. Marefat, "Leveraging blockchain for retraining deep learning

architecture in patient-specific arrhythmia classification," In *Proceedings of the IEEE EMBS International Conference on Biomedical and Health Informatics* (BHI), Las Vegas, Nevada, USA, 2018, pp. 393–397.

[125] K. N. Griggs, O. Ossipova, C. P. Kohlios, A. N. Baccarini, E. A. Howson, and T. Hayajneh, "Healthcare blockchain system using smart contracts for secure automated remote patient monitoring," *Journal of Medical Systems*, vol. 42, no. 7, pp. 1-7, 2018.

[126] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," In *Proceeding of the 2nd International Conference on Open and Big Data* (OBD), 2016, pp. 25–30.

[127] G. G. Dagher, J. Mohler, M. Milojkovic, and P. B. Marella, "Ancile: Privacy-preserving framework for access control and interoperability of electronic health records using blockchain technology," *Sustainable Cities and Society*, vol. 39, pp. 283–297, 2018.

[128] H. Li, L. Zhu, M. Shen, F. Gao, X. Tao, and S. Liu, "Blockchain-based data preservation system for medical data," *Journal of Medical System*, vol. 42, no. 8, pp. 1-13, 2018.

[129] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "MedBlock: Efficient and secure medical data sharing via blockchain," *Journal of Medical System*, vol. 42, no. 8, pp. 1-11 2018.

[130] M. A. Uddin, A. Stranieri, I. Gondal, and V. Balasubramanian, "Continuous patient monitoring with a patient centric agent: A block architecture," *IEEE Access*, vol. 6, pp. 32700–32726, 2018.

[131] B. Shen, J. Guo, and Y. Yang, "MedChain: Efficient healthcare data sharing via blockchain," *Applied Science*, vol. 9, 2019.

[132] N. Nizamuddin, H. R. Hasan, and K. Salah, "IPFS-blockchain-based authenticity of online publications," In *Proceeding of the International Conference on Blockchain*, 2018, pp. 199–212.

[133] W. Meng, E. W. Tischhauser, Q. Wang, Y. Wang, and J. Han, "When intrusion detection meets blockchain technology: A review," *IEEE Access*, vol. 6, pp. 10179-10188, 2018.

[134] M. Banerjee, J. Lee, and K. K. R. Choo, "A blockchain future for the internet of things security: A position paper," *Digital Communications and Networks*, vol. 4, no. 3, pp. 149-160, 2018.