

# **A CRYPTOGRAPHIC FRAMEWORK FOR ULTRA COMPRESSION TECHNIQUES USED FOR SECURED CLOUD STORAGE**

**A Thesis**

**SUBMITTED TO THE  
TILAK MAHARASHTRA VIDYAPEETH, PUNE**

**FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY**

**In Computer Science**

**Under the Board of Modern Sciences & Professional Skills**



**BY**

**Mrs. Swati Dinesh Ghule  
(25315085004)**

**UNDER THE GUIDANCE OF**

**Dr. ANUP GIRDHAR**

**DEPARTMENT OF COMPUTER SCIENCE**

**2025**

# **A CRYPTOGRAPHIC FRAMEWORK FOR ULTRA COMPRESSION TECHNIQUES USED FOR SECURED CLOUD STORAGE**

**A Thesis**

**SUBMITTED TO THE  
TILAK MAHARASHTRA VIDYAPEETH, PUNE**

**FOR THE DEGREE OF  
DOCTOR OF PHILOSOPHY**

**In Computer Science**

**Under the Board of Modern Sciences & Professional Skills**



**BY**

**Mrs. Swati Dinesh Ghule  
(25315085004)**

**UNDER THE GUIDANCE OF**

**Dr. ANUP GIRDHAR**

**DEPARTMENT OF COMPUTER SCIENCE**

**2025**

## **CERTIFICATE BY THE SCHOLAR**

This is to certify that the thesis titled, “**A CRYPTOGRAPHIC FRAMEWORK FOR ULTRA COMPRESSION TECHNIQUES USED FOR SECURED CLOUD STORAGE**” submitted by Mrs. Swati Dinesh Ghule, under the supervision of Dr. Anup Girdhar, P.R.N. Number 25315085004 for the award of Ph.D. degree of Computer science from ‘Tilak Maharashtra Vidyapeeth’ University embodies my original work and has not formed the basis for the award of any degree, diploma, fellowship, titles in this or any other university or any other similar institutions of higher learning.

Place:

Date:

Signature

**Mrs. Swati Dinesh Ghule,**

Research Scholar,

Tilak Maharashtra Vidyapeeth, Pune

Enrollment Number 25315085004

## **CERTIFICATE BY THE SUPERVISOR**

This is to certify that the work incorporated in the thesis “**A CRYPTOGRAPHIC FRAMEWORK FOR ULTRA COMPRESSION TECHNIQUES USED FOR SECURED CLOUD STORAGE**” submitted by Mrs. Swati Dinesh Ghule is an original piece of research work carried out by the candidate under my supervision. Literary presentation is satisfactory and the thesis is in a form suitable for publication. Work evinces the capacity of the candidate for critical examination and independent judgment. Such materials as has been obtained from other sources have been duly acknowledged in the thesis.

Place:

Date:

Signature



**Dr. Anup Girdhar,**

Ph.D Research Guide,

Tilak Maharashtra Vidyapeeth, Pune

## DECLARATION

I hereby declare that this Ph.D. thesis entitled “A Cryptographic Framework for Ultra Compression Techniques Used for Secured Cloud Storage” was carried out by me for the degree of Doctor of Philosophy in Computer Science under the guidance and supervision of Dr. Anup Girdhar, Tilak Maharashtra Vidyapeeth, Pune, India. The interpretations put forth are based on my own experimentation and understanding. The books, articles and websites, which I have made use of are acknowledged at the respective place in the text. All content is plagiarism free. All the information gained is from genuine sources and the results are obtained following the detailed procedure. For the present thesis, which I am submitting to the University, no degree or diploma or distinction has been conferred on me before, either in this or in any other University.

Place:

Date:

Signature

**Mrs. Swati Dinesh Ghule,**

Research Scholar,

Tilak Maharashtra Vidyapeeth, Pune.

Enrollment Number 25315085004

## ACKNOWLEDGEMENTS

I wish to sincerely thank all those who have contributed in one way or another to this study. Words can only inadequately express my deep gratitude to my guide, **Dr. Anup Girdhar**, for his meticulous care, kindness and generosity. His fruitful comments and insightful suggestions have been a crucial formative influence on the present study. He has supported me in every possible way since the beginning of my research. His critical and careful reading of my writing has saved me from a lot of errors. Without his guidance and encouragement, my research would have never come out in the present form. Furthermore, it has been a memorable and enjoyable experience for me to work with him. I thank my family for their continuous trust, encouragement and faith in me through the ups and downs of my research work. I am grateful to my better half Mr. Dinesh Ghule who supported and inspired me and helped me throughout this journey. I would also like to thank my daughter Miss Varada Ghule who constantly co-operated with me in every possible way. I indeed owe a special debt to them. I would also like to acknowledge my parents Dr. Krishnarao Gade and Mrs. Rama Gade, my brothers Dr. Aniket Gade and Mr. Nitiket Gade for their moral support. I am also indebted to my in-laws for their support. A special thanks to my friend **Mrs. Rama S. Bansode** for helping me in this treacherous journey. Also a special thanks to **Dr. Asmita Namjoshi**, Head of TMV, Computer Science Department for their support and facilities provided. I wish to express my sincere gratitude to **Dr. Sunanda Yadav**, the Director of the place of research, for her expert guidance and invaluable suggestions.

I would also like to the Management, **Dr. G. R. Ekbote Chairman, P.E.Society, Pune** for their valuable support. I thank **Dr. (Mrs.) K. R. Joshi, Principal** and **MCA Department HOD, teaching and non-teaching staff** of my Institute, P. E. Society's Modern College of Engineering, Pune for supporting me through out my research work. Last, but not the least I would like to extend my gratitude to all those individuals who have helped me directly or indirectly in the completion of this research activity.

**Mrs. Swati Dinesh Ghule,**  
Research Scholar, TMV.

# **LIST OF FIGURES**

- Figure 1.1: Classification of Data Compression techniques
- Figure 1.2: Classification of compression techniques based on data type.
- Figure 1.3: Classification of compression Technique based on Application
- Figure 1.4: Encryption after Compression
- Figure 1.5: a & b Order of applying Encryption
- Figure 1.6: Cloud Security Challenges
- Figure 1.7: Blockchain Architecture
- Figure 1.8: Structure of the Blockchain
- Figure 1.9: Torrent Architecture
- Figure 1.10: HDFS Architecture
- Figure 1.11: Structure of Thesis
- Figure 2.1: Original File and its Zip file
- Figure 2.2: Size of Zip file
- Figure 2.3: Original files
- Figure 2.4: Size of zip file
- Figure 3.1: Research Methodology
- Figure 3.2: Location of organizations
- Figure 3.3: Companies Covered, location and role of respondents during Data Collection and Analysis
- Figure 3.4: Title level held in the organization
- Figure 3.5: Degree of security responsibility
- Figure 3.6: Technical Reason for Availing Cloud Service
- Figure 3.7: Bandwidth require for data upload on cloud.
- Figure 3.8: Data Transmission mode used by organization
- Figure 3.9: Authentication method used by CSP
- Figure 3.10: Access Control Policies used by CSP
- Figure 3.11: Type of Cloud Services
- Figure 3.12: Percentage of budget allocated for Cloud Service
- Figure 3.13: Level of dependency of Cloud Services
- Figure 3.14: Size of Data uploaded on Cloud
- Figure 3.15: Frequency of data upload on cloud by organisation
- Figure 3.16: Users in Organization use cloud services

Figure 3.17: Cloud Deployment Model used by Organization

Figure 3.18: Response given by CSP after threat detection

Figure 3.19: Common Reason for availing Cloud service

Figure 3.20: Level till which devices get affected

Figure 3.21: Backup Drive used by Organization

Figure 3.22: Data backup frequency on cloud in organization

Figure 3.23: Data Encryption Standard by CSP to your Organization

Figure 3.24: User uploads sensitive data on cloud

Figure 3.25: Level of Security gain by organization

Figure 3.26: Threat Faced by Organization during cloud usage

Figure 3.27: Factors hindering use of cloud services

Figure 3.28: Representation of Huffman tree creation

Figure 3.29: Applying Huffman coding compression algorithm

Figure 3.30: Original file before compression

Figure 3.31: Compressed file after applying Huffman coding Compression algorithm

Figure 3.32: Decompressed file after applying Huffman coding decompression algorithm

Figure 3.33: Screen shot showing the file size

Figure 3.34: Code for RSA key generation.

Figure 3.35: Generated public and private keys

Figure 3.36: code for file encryption using RSA

Figure: 3.37: Encrypted compressed file.

Figure 3.38: Code for file decryption using RSA

Figure 3.39: decrypted file using RSA

Figure 3.40: Code for splitting a file

Figure 3.41: Code for applying SHA256 hash on splitted file

Figure 3.42: Code for creating a block chain and uploading it on server

Figure 3.43: System Model for Proposed Framework

Figure 3.44: Registration Flow of Data Owner and User

Figure 3.45: Data Outsource Workflow

Figure 3.46: Access Control Workflow

Figure 3.47: Integrity Checking Process

Figure 4.1 System Architecture

Figure 4.2: Multivariate Regression Feature Performance plot

Figure 4.3: Decision Tree Classifier Feature Performance plot

Figure 4.4: Random Forest Feature Performance plot

Figure 4.5: SVM Feature Performance Plot

Figure 4.6: ANN Feature Performance Plot

Figure 4.7: UCCS architecture

Figure 4.8: GFS Architecture

Figure 4.9: HDFS Architecture

Figure 4.10: RSA Encryption Flowchart

Figure. 4.11: Comparison of proposed enhanced CP-ABE with RSA method and existing methods for encryption time

Figure. 4.12: Proposed enhanced CP-ABE with RSA method for decryption time with existing methods

Figure. 4.13: Proposed enhanced CP-ABE with RSA method for execution time with existing methods.

Figure 4.14: Structure of the Merkle Tree

Figure 4.15: Transmission Time Comparison between Proposed Work without Optimization and Block Secure

Figure 4.16: Transmission Time Comparison between Proposed Work with Optimization and Block Secure

Figure 4.17: Run-Time Comparison of Proposed Architecture with the Secure Access Control

Figure 4.18: Encoding Time Comparison between Proposed Work and Deduplication with Blockchain

Figure 4.19: Decoding Time Comparison between Proposed Work and Deduplication with Blockchain

Figure 4.20: Proposed framework for Decentralized Secure Storage

Figure 4.21: Comparison of performance between a centralized and a distributive system

Figure 4.22: Traditional Database Processing Structure

Figure 4.23: Hadoop Cluster

Figure 4.24: Hadoop Master-Slave Architecture

Figure 5.1: Menu of UCCS sever.

Figure 5.2: Seedbox Web User Interface

Figure 5.3: Distribution of load on 4 nodes aggregated for 1 hour.

Figure 5.4: A part Blockchain of five blocks generated by Implemented model

Figure 5.5: Comparison of Ethereum Blockchain database architecture without Indexing and with Indexing on top of Hadoop.

# **LIST OF TABLES**

Table 1.1: Classification of various coding scheme

Table 3.1: Qualitative versus Quantitative Research

Table 3.2: List of Parameters

Table 3.3: Desired Confidence Level

Table 3.4: Comparison of some cloud storage system

Table 3.5: Count of locations of respondents

Table 3.6: Type of organizations of respondents

Table 3.7: Count of title level of respondents

Table 3.8: Count of degree of responsibilities of respondents

Table 3.9: Count of Technical Reason for Availing Cloud Service

Table 3.10: Bandwidth required for data upload on cloud

Table 3.11: Data Transmission mode used by organization

Table 3.12: Authentication Method used by CSP

Table 3.13: Access Control Policies used by CSP

Table 3.14: Frequency of data upload on cloud by Organization

Table 3.15: Percentage of IT budget allocated to cyber security

Table 3.16: Level of concern for Cloud Services.

Table 3.17: Size of Data uploaded on cloud

Table 3.18: Users in Organization use cloud Service

Table 3.19: Data Backup Frequency in organization on cloud

Table 3.20: Cloud Deployment Model used by Organization

Table 3.21: Backup Drive used by Organization

Table 3.22: Count of Technical Reason for Availing Cloud Service

Table 3.23: Data Encryption Standard provided by CSP

Table 3.24: Sensitive data uploaded on cloud by user

Table 3.25: Count of common reason for availing cloud service in organizations

Table 3.26: Count of level till which devices get affected

Table 3.27: Level of Security Gain by Organization

Table 3.28: Threat Faced during the use of cloud service

Table 3.29: Response given by CSP after threat detection

Table 3.30: Factors hindering use of cloud services

Table 3.31: Notation and Description

Table 4.1: Statistical models and its accuracy

Table 4.2: Comparison of UCCS with GFS and HDFS1

Table 4.3: The comparison of encryption time with existing methods for different key lengths

Table 4.4: The comparison of decryption time with existing methods for different key length

Table 4.5: The comparison of execution time with existing methods for different key length.

Table 4.6: Table showing result of Huffman coding algorithm

Table 4.7: Compression Ratio of Different File Formats

Table 4.8: Comparison of Checksum and Markel Root

Table 4.9: Comparison of MFA, OAuth2.0 and Kerberos

Table 4.10: Comparison of security aspect of UCCS, GFS and HDFS

# **LIST OF ABBREVIATIONS**

|          |                                                              |
|----------|--------------------------------------------------------------|
| CP-ABE   | :Ciphertext Policy Attribute-based Encryption                |
| RSA      | :Rivest-Shamir-Adleman                                       |
| IT       | :Information Technology                                      |
| URL      | :Uniform Resource Locator                                    |
| DC       | :Data Compression                                            |
| LZ       | :Lempel Ziv                                                  |
| BWT      | :Burrows-wheeler transform                                   |
| RLE      | :Run Length Encoding                                         |
| LZW      | :Lempel-Ziv-Welch                                            |
| AES      | :Advanced Encryption Standard                                |
| DES      | :Data Encryption Standard                                    |
| ABE      | :Attribute-based encryption                                  |
| SaaS     | :Software as a Service                                       |
| CSP      | :Cloud Service Providers                                     |
| IP       | :Internet Protocol                                           |
| XML      | :Extensible Markup Language                                  |
| ABEY     | :Advanced Blockchain for Enhanced Yields                     |
| VM.      | :Virtual Machine                                             |
| GB       | :Giga Byte                                                   |
| IPFS     | :Inter Planetary File System                                 |
| DAG      | :Directed Acyclic Graph                                      |
| I/O      | :Input / Output                                              |
| P2P      | :Peer-to-Peer                                                |
| GFS      | :Google File System                                          |
| HDFS     | :Hadoop Distributed File System                              |
| MB,      | :Mega Byte                                                   |
| CPU      | :Central Processing Unit                                     |
| SHA,     | :Secure Hash Algorithm                                       |
| SNR      | :Signal-to-noise ratio                                       |
| BCP-ABE- | :Blockchain-based CP-ABE scheme with partially hidden access |
| PHAS     | structures                                                   |
| AWS      | :Amazon Web Services                                         |

|        |                                                |
|--------|------------------------------------------------|
| IaaS   | :Infrastructure as a Service                   |
| Paas   | :Platform as a service                         |
| JSP    | :Java Server Pages                             |
| SQL    | :Structured Query Language                     |
| SOA    | :Service Oriented Architecture                 |
| SUP    | :Storage Upkeep Protocol                       |
| VMC    | :Vulnerability Management Center               |
| KVM    | :Kernel based Virtual Machine                  |
| WAN    | :Wide Area Network                             |
| TPA    | :Third Party Assistance                        |
| HTTP   | :HyperText Transfer Protocol                   |
| NDN    | :Named Data Network                            |
| TCP/IP | :Transfer Control Protocol / Internet Protocol |
| S4DS   | :Society for Data Science                      |
| MAC    | :Mandatory Access Control                      |
| DAC    | :Discretionary Access Control                  |
| RBAC   | :Role Base Access Control                      |
| RAC    | :Rule Based Access Control                     |
| ABAC   | :Attribute Based Access Control                |
| TB     | :Tera Byte                                     |
| Mbps   | :Mega bits per second                          |
| KB     | :Kilo Byte                                     |
| PK     | :Public Key                                    |
| UCCS   | :Ultra Compression Cloud Storage               |
| MK     | :Master Key                                    |
| SK     | :Secret Key                                    |
| GP     | :Global Parameter                              |
| CF     | :Compressed File                               |
| DO     | :Data Owner                                    |
| CT     | :Ciphertext                                    |
| JSON   | :Java Script Object Notation                   |
| SVM    | :Support Vector Machine                        |
| ANN    | :Artificial Neural Networks                    |

|      |                                           |
|------|-------------------------------------------|
| MSE  | :Mean Square Error                        |
| RMSE | :Root Mean Square Error                   |
| AI   | :Artificial Intelligence                  |
| CT   | :Compression Time                         |
| CR   | :Compression Ratio                        |
| ACL  | :Access Control Lists                     |
| API  | :Application Programming Interface        |
| KDC  | :Kerberos Key Distribution Center         |
| TGT  | :Ticket-granting ticket                   |
| SSO  | :Single Sign-On                           |
| LDAP | :Lightweight Directory Access Protocol    |
| MFA  | :Multifactor Authentication               |
| SSL  | :Secure Socket Layer                      |
| SASL | :Simple Authentication and Security Layer |
| JVM  | :Java Virtual Machine                     |
| UI   | :User Interface                           |
| SSH  | :Secure Shell                             |
| FTP  | :File Transfer Protocol                   |
| SFTP | :SSH FTP                                  |

# **TABLE OF CONTENTS**

## **CHAPTER 1: INTRODUCTION**

|                                                             |    |
|-------------------------------------------------------------|----|
| 1.1 Introduction to the Study.....                          | 1  |
| 1.2 Data Compression Techniques.....                        | 2  |
| 1.2.1 Based on Data Quality.....                            | 3  |
| 1.2.2 Based on Coding Scheme.....                           | 4  |
| 1.2.3 Based on Data Type.....                               | 5  |
| 1.2.4 Based on Application.....                             | 6  |
| 1.2.5 Data compression with cryptography.....               | 6  |
| 1.2.6 Huffman Coding compression Technique.....             | 7  |
| 1.3 Encryption Algorithms used on compressed Data .....     | 8  |
| 1.3.1 Compression – RSA Technique.....                      | 9  |
| 1.3.2 Limitations of AES and DES encryption.....            | 9  |
| 1.3.3 Access Control with Encryption - CP-ABE with RSA..... | 10 |
| 1.3.4 Cryptographic Hash – Markle Root.....                 | 11 |
| 1.4 Secure Cloud Storage.....                               | 11 |
| 1.4.1 Cloud storage issues.....                             | 12 |
| 1.4.1.1 Data Privacy and Integrity.....                     | 12 |
| 1.4.1.2 Data recoverability and Vulnerability.....          | 13 |
| 1.4.1.3 Improper media Refinement.....                      | 13 |
| 1.4.1.4 Data Backup.....                                    | 13 |
| 1.4.2 Identity Management and Access Control.....           | 13 |
| 1.4.2.1 Malicious insider.....                              | 14 |
| 1.4.2.2 Outside Intruder.....                               | 15 |
| 1.5 Blockchain Technology.....                              | 16 |
| 1.5.1 Key characteristics of Blockchain.....                | 17 |
| 1.5.2 Smart Contract.....                                   | 18 |
| 1.6 Data Compression using P-2-P technique.....             | 19 |
| 1.6.1 Tracking Server.....                                  | 20 |
| 1.6.2 Client.....                                           | 20 |
| 1.7 What is GFS.....                                        | 21 |
| 1.8 What is HDFS.....                                       | 22 |
| 1.8.1 NameNode.....                                         | 23 |
| 1.8.2 DataNode.....                                         | 23 |
| 1.8.3 Block.....                                            | 23 |
| 1.8.4 Features of HDFS.....                                 | 24 |
| 1.9 Motivation for the Study.....                           | 24 |
| 1.10 Need of the Study.....                                 | 26 |
| 1.11 Scope and Contribution of the study.....               | 26 |
| 1.12 Research Methodology used in the study.....            | 28 |
| 1.13 Structure of the Thesis.....                           | 29 |
| 1.14 Keywords used.....                                     | 32 |

## **CHAPTER 2: LITERATURE REVIEW**

|                                        |    |
|----------------------------------------|----|
| 2.1 Introduction to topic.....         | 33 |
| 2.2 Analysis of Literature Survey..... | 33 |
| 2.3 Data Compression Techniques.....   | 34 |
| 2.4 Data Encryption Techniques.....    | 36 |
| 2.5 Cloud Computing Concept.....       | 38 |

|                                                                  |     |
|------------------------------------------------------------------|-----|
| 2.6 Understanding cloud architecture and working .....           | 39  |
| 2.7 Decentralized Data Storage.....                              | 41  |
| 2.8 File System Security Related CVE's.....                      | 42  |
| 2.9 Security Aspects for Data.....                               | 43  |
| 2.10 Secure Cloud Storage.....                                   | 45  |
| 2.11 Understanding Blockchain Technology.....                    | 46  |
| 2.12 Working of P-2-P Technique.....                             | 48  |
| 2.13 Working of emulator like VMware Workstation.....            | 49  |
| 2.14 Secure Cloud Framework.....                                 | 51  |
| 2.15 Distributed File Storage.....                               | 52  |
| 2.16 Centralized File Storage.....                               | 54  |
| 2.17 Studying digital currencies like Bitcoins and Ethereum..... | 56  |
| 2.18 Hadoop Distributed File System.....                         | 57  |
| 2.19 Gap Analysis.....                                           | 58  |
| <b>CHAPTER 3: RESEARCH METHODOLOGY</b>                           |     |
| 3.1 Introduction: Research Approach.....                         | 63  |
| 3.2 Type of Research Methods.....                                | 63  |
| 3.3 Method of Data Collection .....                              | 65  |
| 3.4 Research Methodology Applied.....                            | 68  |
| 3.5 Survey Questionnaire.....                                    | 74  |
| 3.5.1 Survey on demographic details.....                         | 80  |
| 3.5.2 Survey on technical details.....                           | 84  |
| 3.5.3 Survey on cloud service details.....                       | 88  |
| 3.5.4 Survey on security details of cloud service.....           | 96  |
| 3.6 Key Understandings.....                                      | 103 |
| 3.7 Experimental Identification of the Problem.....              | 104 |
| 3.7.1 Hypothesis for qualitative analysis.....                   | 104 |
| 3.7.2 Hypothesis of Study.....                                   | 104 |
| 3.7.3 Testing of Hypothesis.....                                 | 105 |
| 3.7.3.1 H01-Case Study-1.....                                    | 105 |
| 3.7.3.2 H02-Case Study-2.....                                    | 112 |
| 3.7.3.3 H03-Case Study-3 .....                                   | 118 |
| 3.7.4 Key Findings of the Hypothesis Experiments.....            | 122 |
| 3.8 Proposed Framework.....                                      | 123 |
| 3.8.1 Objective of the Study.....                                | 123 |
| 3.8.2 Significance of Research.....                              | 123 |
| 3.8.3 Need of the Study.....                                     | 124 |
| 3.8.4 Benefit of the Study.....                                  | 126 |
| 3.8.5 The Framework of the Present Study.....                    | 127 |
| 3.8.6 Conceptual Framework.....                                  | 128 |
| 3.8.7 Proposed model for UCCS.....                               | 128 |
| 3.8.8 Objective of Framework Implementation.....                 | 133 |
| 3.8.9 Result and Expected outcome.....                           | 133 |
| 3.9 Algorithms used for Building the Framework.....              | 134 |
| 3.9.1 The Ultimate Algorithm.....                                | 134 |
| 3.9.2 Smart Contract Function.....                               | 138 |
| 3.10 Example Scenario.....                                       | 149 |
| 3.11 Security Analysis.....                                      | 150 |

## **CHAPTER 4: ANALYSIS AND INTERPRETATION**

|                                                                |     |
|----------------------------------------------------------------|-----|
| 4.1 Introduction.....                                          | 152 |
| 4.2 Quantitative Analysis and its Interpretation.....          | 153 |
| 4.2.1 System Architecture.....                                 | 153 |
| 4.2.2 Parameter Simulation.....                                | 154 |
| 4.2.3 Evaluation of Proposed Framework.....                    | 154 |
| 4.2.3.1 Multivariate Regression.....                           | 155 |
| 4.2.3.2 Decision Tree Classifier.....                          | 156 |
| 4.2.3.3 Random Forest.....                                     | 157 |
| 4.2.3.4 Support Vector Machine (SVM).....                      | 158 |
| 4.2.3.5 Artificial Neural Network (ANN).....                   | 159 |
| 4.3 Algorithms used in proposed framework.....                 | 161 |
| 4.3.1 Compression Technique used in UCCS – Huffman Coding..... | 161 |
| 4.3.2 Encryption Technique used in UCCS – RSA based CPABE..... | 164 |
| 4.3.3 UCCS Optimization Algorithm HoneyBee .....               | 169 |
| 4.3.4 UCCS Integrity Verification Technique – Merkle Root..... | 171 |
| 4.3.5 UCCS Multifactor Authentication Technique.....           | 172 |
| 4.3.6 UCCS Access Control Policy – ABAC.....                   | 172 |
| 4.4 Comparative Study of UCCS with GFS and HDFS.....           | 174 |
| 4.4.1 Distributed File System.....                             | 175 |
| 4.4.2 Decentralized Network.....                               | 184 |
| 4.4.3 UCCS Encryption Technique.....                           | 186 |
| 4.4.4 UCCS Compression Technique.....                          | 187 |
| 4.4.5 UCCS Resource Optimization Technique.....                | 188 |
| 4.4.6 UCCS Integrity Verification Technique.....               | 190 |
| 4.4.7 UCCS Multifactor Authentication Technique.....           | 194 |
| 4.4.8 Access Control Policy used in UCCS.....                  | 197 |
| 4.4.9 Security Technique for storage used in UCCS.....         | 200 |
| 4.5 Deploying UCCS Server on VMWare .....                      | 202 |
| 4.6 System Performance.....                                    | 203 |
| 4.7 Comparative Analysis.....                                  | 207 |
| 4.8 Distributive execution of UCCS Server.....                 | 211 |
| 4.8.1 Introduction.....                                        | 211 |
| 4.8.2 Proposed Distributive UCCS server.....                   | 213 |
| 4.8.3 Centralized Vs Distributive Computing.....               | 214 |
| 4.8.4 Role of Apache Spark in UCCS Server.....                 | 215 |
| 4.8.5 Working of Hadoop Cluster.....                           | 219 |

## **CHAPTER 5: FINDINGS AND CONCLUSION**

|                                                                     |     |
|---------------------------------------------------------------------|-----|
| 5.1 Execution for uploading a sample file on the Framework.....     | 220 |
| 5.1.1 Compressing the file.....                                     | 220 |
| 5.1.2 Key Generation and Encryption of the file.....                | 221 |
| 5.1.3 Checking Integrity of the file.....                           | 223 |
| 5.1.4 Adding file in block.....                                     | 223 |
| 5.1.5 Adding block in blockchain.....                               | 224 |
| 5.1.6 Seedbox testing.....                                          | 225 |
| 5.1.7 Running on VMWare.....                                        | 227 |
| 5.2 Execution for a downloading sample file from the Framework..... | 228 |
| 5.2.1 Reading Block .....                                           | 228 |
| 5.2.2 Reading file.....                                             | 228 |
| 5.2.3 Decrypt the file.....                                         | 229 |

|                                                        |     |
|--------------------------------------------------------|-----|
| 5.2.4 Decompress the file.....                         | 229 |
| 5.3 Static Analysis.....                               | 229 |
| 5.4 Final Results.....                                 | 230 |
| 5.4.1 Results obtained from the Framework.....         | 231 |
| 5.4.2 Comparison with existing centralized system..... | 231 |
| 5.4.2.1 Working on Distributed Framework.....          | 232 |
| 5.4.2.2 Blockchain for record.....                     | 233 |
| 5.4.2.3 Using Ethereum for making Blockchain.....      | 235 |
| 5.4.3 Summary of work done .....                       | 237 |
| 5.4.4 Contribution of Research.....                    | 238 |
| 5.5 Limitations of the Study.....                      | 239 |
| 5.6 Conclusions.....                                   | 240 |
| 5.7 Further Research Potential.....                    | 240 |
| <b>REFERENCES</b>                                      | 242 |

# **ABSTRACT**

The increase in networking technology and the need for computing resources have prompted many organizations to outsource their storage. Cloud computing is the availability of computing service over the Internet where user can use the resource available on cloud. Cloud computing offers easy access and high performance computing and storage infrastructure using web service. This rapid development in clouds has impact on security of data the most interesting question is how to build secure cloud storage ensuring data security and privacy in cloud environments is crucial.

For quicker communication and exchange of data, most of the information comes in the form of electronic data. Most of the computer applications are not secure and these applications exchange lot of confidential data having different file formats. The confidential data of these types need to be transmitted securely and stored efficiently. Thus, along with security, factors like implementation cost and performance of various cryptographic algorithms also needs to be measured for practical implementation.

Data compression is a technique of reducing the size of the data file so that the file should take less disk space for storage. Compression of a file can be determined by encoding of file. In lossless data compression algorithm, no loss of data is there while compressing a file, therefore confidential data can be regenerate if it is compressed using lossless data compression. Compression is used to reduces the redundancy and if a compressed file is encrypted it is gives a better security and faster transfer rate across the network than uncompressed encrypted file.

The demand for blockchain invention and the significance of its application has motivated ever-progressing consideration in various scientific and practical areas. Even though it is still in the early stage of testing, the blockchain is being viewed as a progressive solution to point out 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 guarantee 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 badly 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 four significant contributions in the blockchain-based cloud storage domain.

**First,** we have conducted a literature review of compression, encryption, and use 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 Huffman coding algorithm for data compression and 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 Python programming language. The various experiments are conducted using tools such as SimuCloud and Pytest 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 ensure fine-grained access control. Various experiments analyze the performance and compare the proposed architecture with various existing works.

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.

### **Benefit of the Study**

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.
- 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. 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.

## **Conceptual Model Framework**

Our conceptual model framework is a design created to deal with disadvantages and challenges of cloud storage, we designed python-based blockchain architecture for the cloud storage system to provide a secure environment. The proposed architecture implements Huffman coding to compress data and CP-ABE to provide a key generation mechanism using bilinear mapping-based cryptography and perform data access control mechanisms. The frameworks 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 and file data and tracks all access and validation records. The architecture also implements a Honeybee optimization algorithm to optimize the resource utilization on the cloud storage system and reduce transaction processing time. By using blockchain technology, we achieve decentralization with security without a trusted central authority. It makes available four primary services to ensure security features in a cloud storage system:

1. A registration procedure is designed to register data owners and users using a key generation technique to offer an authentication feature.
2. The data owners store the user and file Meta details in the

blockchain structure and set access rules to maintain authorization.

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

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

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.

### **Objectives of the Framework:**

1. To identify and mitigate risk factors occurring at the time of storage and retrieval of data.
2. Working with live Multifactor authentication model to understand their behavior and security methodology.
3. Understanding the working of Compression and decompression Algorithms for creating compressed file.
4. Understand the working of key generation algorithm.
5. Identify the behavior of CP-ABE with RSA Encryption and decryption Algorithms at the time of creating encrypted file.
6. Optimize the resource to minimize the execution time and overhead while storing it.
7. Performs the integrity verification process using the Merkle root method.
8. Notifies the behavior of distributed server and decentralized server while uploading and downloading the file.
9. Proposing how this framework would work best in a decentralized environment.
10. Analysis of the proposed framework with its limitation and future scope.

# CHAPTER 1 INTRODUCTION

## 1.1 Introduction to the Study

Typically, the majority of data is stored on local networks using servers that can be clustered and share storage. Disasters such as fire, flood, tornado, and hard disk failure can occur unpredictably. These calamities have the potential to obliterate all data saved locally. Users might employ remote backup solutions to safeguard their data from such difficulties. Online backup is an Internet-based system that automatically backs up specified files. These files are kept in an internet location and can be accessed from anywhere. This is particularly advantageous in the event that a local computer or server becomes corrupted or damaged. The advantages of utilizing online storage services extend beyond data protection to include additional amenities. An alternative choice is to save duplicates of data in cloud storage. Cloud storage has become an integral component of the IT sector in recent times. Cloud storage is a specific use case of cloud computing. The cloud application is most advanced in this phase. The provision of storage services to users via the internet depends on factors such as the cluster application, grid technology, and distributed file system. Cloud storage often offers a cost-effective storage service that is very reliable and secure in most situations. The most advanced cloud storage application is online backup or file synchronization.

Cloud storage services facilitate the seamless sharing of content across several devices, including computers and mobile devices. Cloud storage offers several advantages, including cost reduction and effective data loss management. Currently, there is a growing emphasis on third-party review and remote integrity checking in various works, which aim to ensure data dynamics. The remote archive service is accountable for effectively preserving the data. The remote data integrity checking protocol detects instances of data corruption and identifies servers that are not performing properly in cloud storage. The suggested study focuses on analyzing the data partitioning approach and remote data integrity testing. The security technique is emphasized to mitigate irreversible data loss. The storage and retrieval process is streamlined by minimizing the storage space required for data or file storage and retrieval. Cloud storage services enable seamless access and sharing of data/files across multiple devices and locations.

Transferring files, such as photographs and videos, might provide challenges when they are excessively large or when there is a significant quantity of them. One can easily share a URL and distribute files to selected individuals by uploading them to cloud storage.

Effective data management is crucial for sustaining service levels and safeguarding essential corporate information, as all data in a cloud environment is stored in a shared system. The dependability of internet connectivity is crucial for small businesses utilizing cloud computing. However, the intermittent internet failures adversely affect the dependability of cloud computing services. Smaller firms intending to adopt cloud computing may be significantly affected by widespread outages. To enhance the efficiency and efficacy of cloud computing in data management, even when bandwidth is limited and regardless of the device used to access data, one potential approach is to employ data compression in the cloud. Data compression aids in minimizing the utilization of costly resources in this context. For instance, the capacity of storage on a disk or the amount of data that can be transmitted across a network. Data compression is an efficient technology that enhances the speed of transmission.

The transmission of information over an untrusted network presents concerns regarding the security of the information. Cryptography is the method used to transmit messages in a covert manner. By utilizing encryption, one can ensure the security of messages by transforming the data into unreadable formats. In order to address the aforementioned issues in the study, it is necessary to develop a framework for implementing highly efficient compression techniques specifically designed for safe cloud storage.

## **1.2 Data Compression Techniques**

Compression is the skill of expressing information in a condensed manner rather than its initial or uncompressed state. The primary objective of data compression is to identify and remove repetitive patterns using various efficient techniques, so enabling the compressed data to conserve space for storage, reduce time for data transport, and minimize costs associated with data maintenance. In order to eliminate redundancy, the original file is represented using a system of coded notation, resulting in what is referred to as an encoded file. An efficient compression algorithm is characterized by a compressed file size that is smaller than the size of the original file. In order to get the

original file, it is necessary to decode the encoded file (Sankalp Shukla, Ritu Gupta, Dheeraj Singh Rajput, Yashwant Goswami, and Vikash Sharma 2022).

### 1.2.1 Based on data quality

In general, the implementation of a Data Compression (DC) approach has a particular impact on the quality of the data, depending on specific criteria. When a direct current (DC) approach is employed for routine tasks such as messaging or internet browsing, the emphasis on the quality of encoded data is not particularly high. However, in the context of text data compression, altering a single character in a text document is deemed unacceptable since it results in a complete change in meaning. Similarly, in the context of medical imaging or remote sensing picture compression, it is not expected that there will be any significant alterations in the values of individual pixels. The importance of data quality in a DC approach is highly contingent upon the specific type of data or application being utilized.

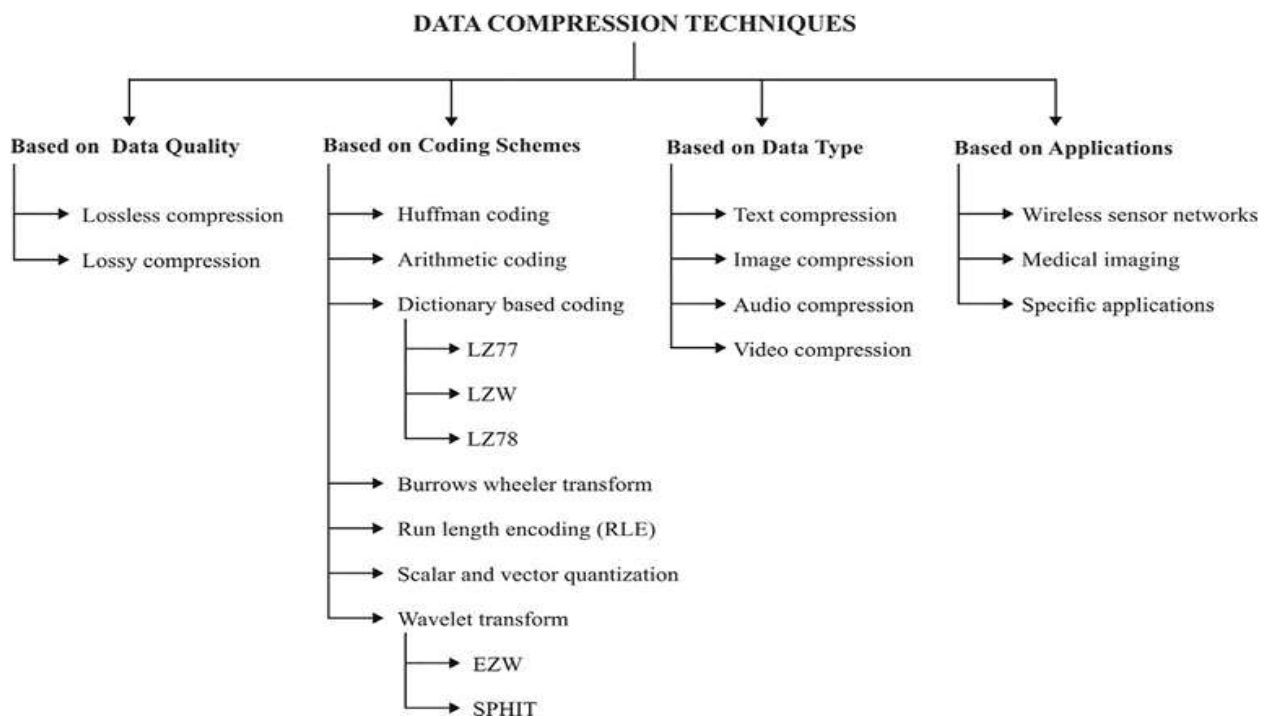


Figure 1.1: Classification of Data Compression techniques.

DC approaches can be classified into lossless and lossy compression based on the need for compressed data quality. Lossless compression, as the name suggests, ensures that no information is lost during the compression process. This means that the compressed data is exactly the same as the original data. It is utilized in applications where data loss is unacceptable, such as text processing, medical imaging, legal forensics, military

images, and satellite imaging. Lossy compression algorithms are advantageous in certain scenarios when the compressed data does not precisely match the original data, but an approximation of the original data is still acceptable. Lossy compression approaches achieve a greater compression ratio compared to lossless compression techniques (Jayasankar, Thirumal and Ponnuram, 2021, p124).

### 1.2.2 Based on coding schemes

This section provides a review of popular coding techniques that are considered the predecessors in the field of data compression. These techniques include Huffman coding, Arithmetic coding, LZ coding, Burrows-wheeler transform (BWT) coding, RLE, transform coding, predictive coding, dictionary-based methods, fractal compression, Scalar and vector quantization. The techniques are depicted in Figure 1.1 and their comparison is presented in Table 1.1.

| Reference                  | Coding              | Feature                        | Compression Type   | Versions                                                                                                          | Advantages                                       | Applications                                     |
|----------------------------|---------------------|--------------------------------|--------------------|-------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|--------------------------------------------------|
| Huffman (1952)             | Huffman Coding      | Entropy Based                  | Lossless           | Minimum variance Huffman code, Length Limited Huffman code, Adaptive non-binary Golomb-rice coding, Tunstall code | Effective in all file formats                    | ZIP, ARG, JPEG, MPEG, PKZIP                      |
| Langdon (1984)             | Arithmetic Coding   | Entropy Based                  | Lossy and Lossless | Adaptive arithmetic coding, Binary arithmetic coding                                                              | Flexibility                                      | JPEG multimedia applications                     |
| Ziv and Lempel (1977)      | LZ coding           | Dictionary Based Coding        | Lossless           | LZ77, LZ78, LZW                                                                                                   | Compress all kinds of data                       | TIFF, GIF, PDF, GZip, ZIP, V.42, Deflate and PNG |
| Saupe and Hamzaoui (1994)  | Fractal Compression | Block based coding             | Lossy              | -                                                                                                                 | Suitable for textures and natural images         | Live video broadcasting                          |
| Burrows and Wheeler (1994) | BWT                 | Block sorting compression      | Lossless           | -                                                                                                                 | No need to store additional data for compression | BZip2                                            |
| Capon (1999)               | RLE                 | Employs in High Redundant data | Lossless           | -                                                                                                                 | Faster                                           | TIFF, BMP, PDF and fax                           |

|               |                                |                                                |                    |   |                 |   |
|---------------|--------------------------------|------------------------------------------------|--------------------|---|-----------------|---|
| Sayood (2006) | Scalar and Vector Quantization | Represents larger set of values to smaller set | Lossless and Lossy | – | Less complexity | – |
|---------------|--------------------------------|------------------------------------------------|--------------------|---|-----------------|---|

Table 1.1: Classification of various coding scheme

### 1.2.3 Based on the data type

DC is commonly employed for compressing several types of data, including text, images, audio, video, and particular application data. This section provides an overview of the significant and up-to-date compression algorithms for each data type. The classification hierarchy can be seen in Figure 1.2.

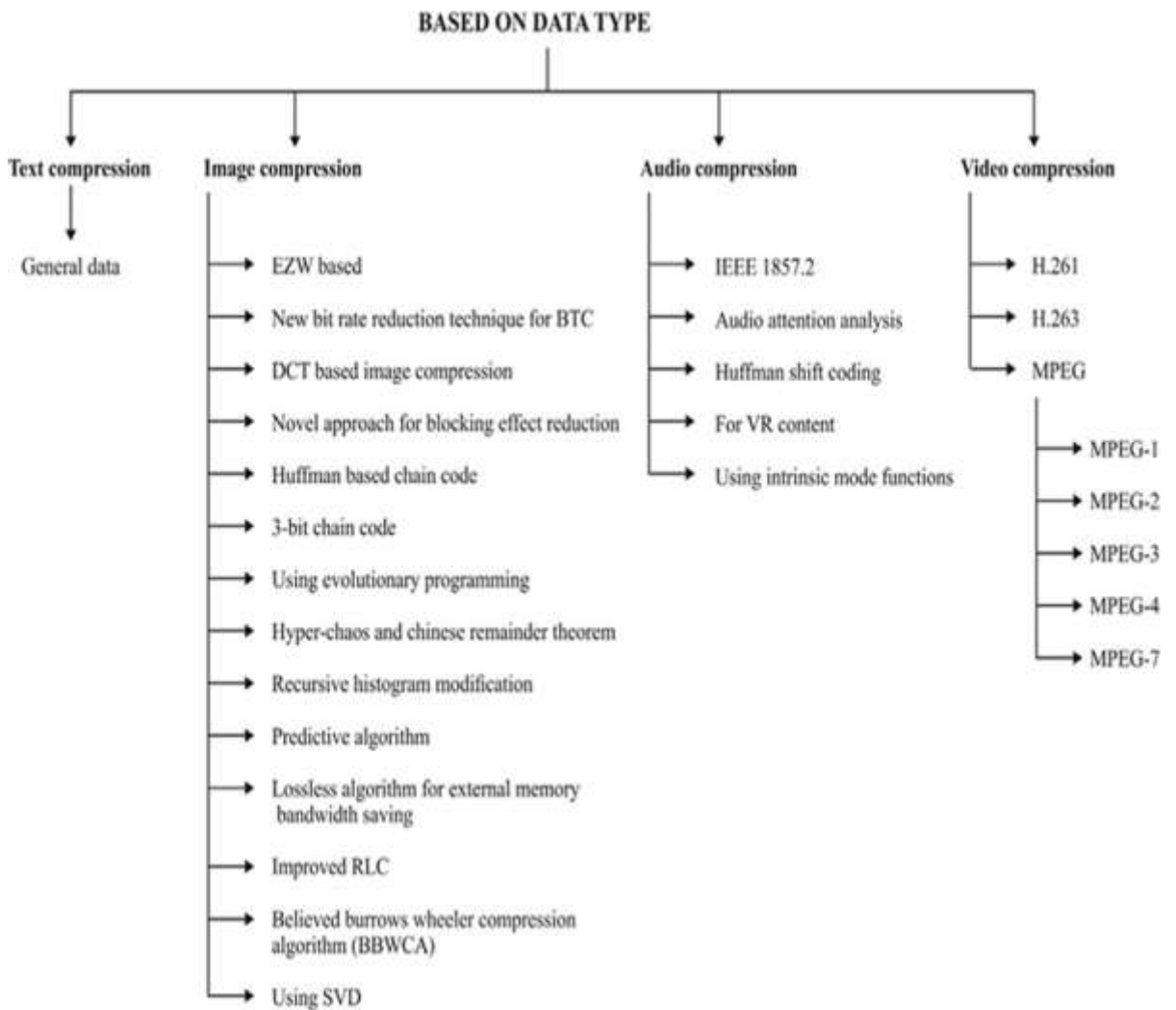


Figure 1.2: Classification of compression techniques based on data type.

### 1.2.4 Based on application

While many basic compression techniques may be relevant to certain applications, the emphasis has primarily been on the technique itself rather than its application. Nevertheless, certain techniques exist where it is impractical to distinguish the methodology from its application. This is because certain strategies rely on the characteristics of the data used in the application. This section examines the compression strategies that have been created for particular purposes. The classification hierarchy can be shown in Figure 1.3.

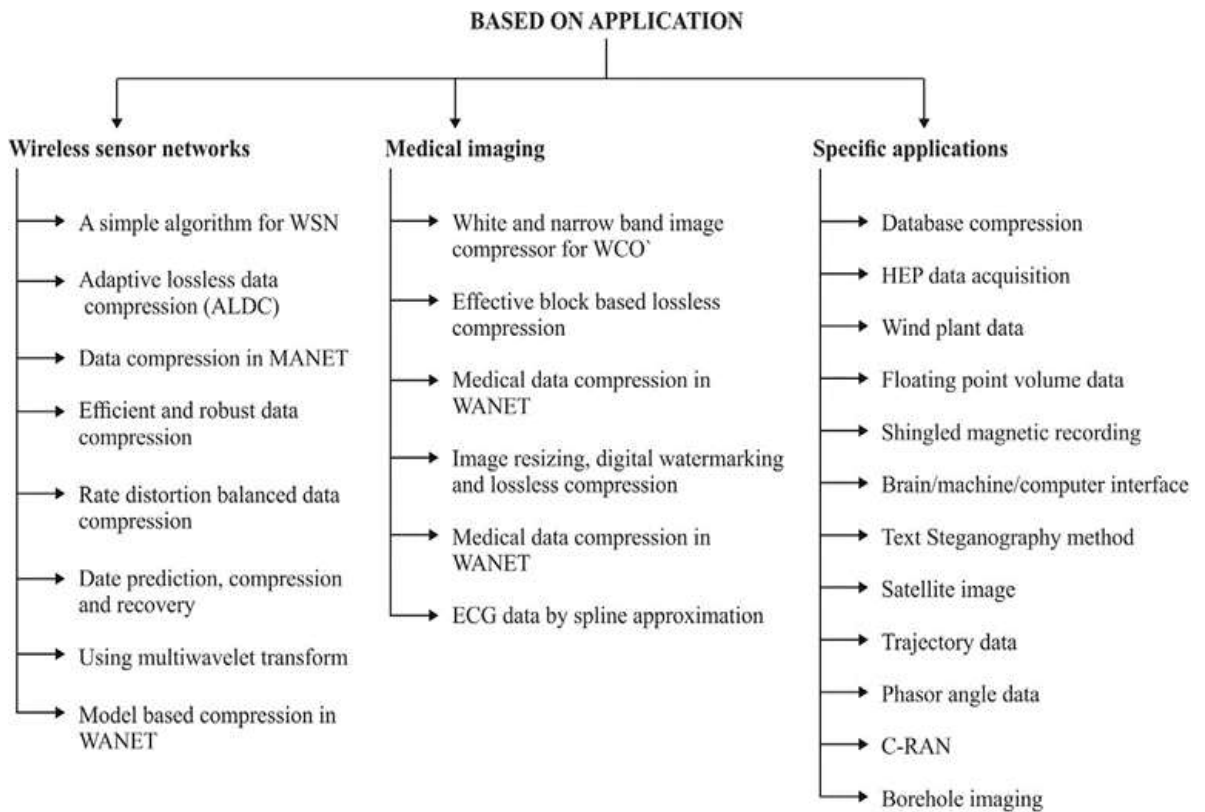


Figure 1.3: Classification of compression Technique based on Application

### 1.2.5 Data compression with cryptography

Today, the method of communication has undergone a significant transformation. The objective of digital communication is to achieve efficient and secure communication. The research in this topic focuses on enhancing communication methods to meet higher standards of efficiency and security. Efficiency is achieved by data compression, while security is ensured through encryption.

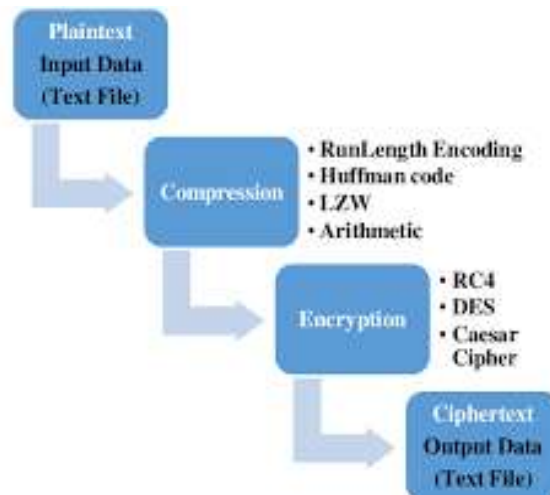


Figure 1.4: Encryption after Compression

Information theory states that there is a maximum level of compression that can be achieved for a digital file, known as the Shannon entropy. The higher the entropy, the lower the potential for compressing the data. Thus, unpredictability is the adversary of compression. However, encryption requires the introduction of randomness into digital data in order to enhance security. It is advisable to always compress data before encrypting it. This is seen in the workflow depicted in Figure 1.4.

### 1.2.6 Huffman Coding Compression Technique

Huffman coding is a frequently utilized encoding technique that is highly popular for compressing all types of files. The approach was invented in the 1950s by D. A. Huffman, for whom it is called. The basis of this is the likelihood or statistical frequency at which a symbol, which could be a character or pixel, occurs. In Huffman coding, when the file is a text file, each character is considered a symbol. Conversely, when the file is an image file, each pixel is considered a symbol. The objective is to utilize a smaller number of bits to encode symbols that occur more frequently. The symbol with the highest frequency is assigned the fewest number of bits in the code. This means that there are less bits available than the minimum required to assign to any other symbol. A Huffman code is a type of prefix code. This implies that the binary code assigned to any symbol does not serve as the beginning of the binary code assigned to any other symbol.

To implement Huffman coding on a file. Sort the symbols in the file in a decreasing order based on their frequency in the file, and then consider the following two observations.

- a. Symbol with higher frequency will be assigned shorter code words compared to symbols with lower frequency.
- b. The two symbols with the lowest frequency will have equal lengths.

The Huffman code is constructed by iteratively merging the least frequent symbols until only two frequencies of two compound symbols remain. This procedure generates a code tree, from which the Huffman code is derived by assigning labels to the branches (Nites Agrawal, 2020).

### **1.3 Encryption Algorithms used on compressed data**

Data compression is a technique used to decrease the size of a data file, resulting in reduced disk space requirements for storage. The compression of a file is determined by the encoding of the file. In a lossless data compression algorithm, no data is lost during the compression process. As a result, if secret material is compressed using lossless data compression, it can be accurately reproduced. Compression decreases redundancy, resulting in improved security and faster transfer rates when encrypting and transferring compressed files compared to uncompressed files. The majority of computer programs lack security measures and frequently communicate a significant amount of confidential data in various file formats, including audio, image, text, and video data. It is imperative to securely transmit and efficiently store this kind of sensitive information. The necessity of combining these two areas has been acknowledged. The dilemma arises over the sequence in which these two processes should be conducted, namely whether Compression should be applied before to Encryption or Encryption should be applied prior to Compression, as depicted in figures 1.5a and b.

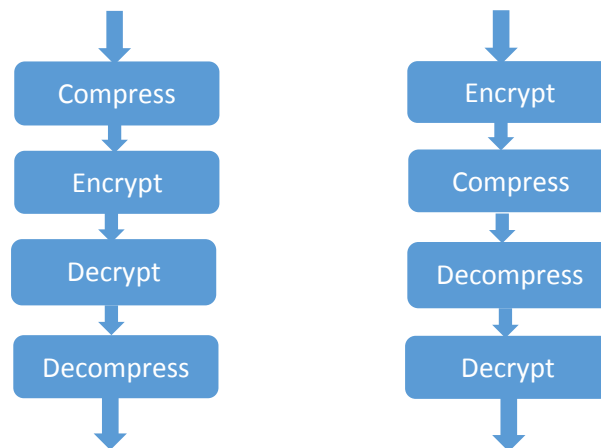


Figure 1.5: a & b Order of applying Encryption

### 1.3.1 Compression - RSA Technique

The efficacy of encryption-decryption operations has emerged as a significant challenge in asymmetric cryptography. The Compression-RSA approach was created to address the issue of inefficiency. The RSA cryptosystem's performance can be negatively impacted by large plaintext due to the exponential operations involved in its encryption and decryption procedures. Furthermore, the compression technique will require additional time, as the RSA cryptosystem with Compression-RSA technique entails performing compression and decompression processes prior to encryption and after decryption, respectively. The overall execution time of the RSA cryptosystem escalated in proportion to the quantity of the initial plaintext. This is a result of the repeated execution of exponential operations. In conclusion, the Compression-RSA technique enhances the speed of the encryption and decryption processes in the RSA cryptosystem.

### 1.3.2. Limitations of AES and DES encryption

- The primary drawback of symmetric encryption methods such as AES and DES is in their reliance on a solitary, exclusive cryptographic key for both encryption and decryption of data.
- Symmetric algorithms have some vulnerabilities, such as the presence of weak keys and the unsafe transmission of secret keys.

- Implementing AES in counter mode in software requires careful consideration of both performance and security because to its complexity.
- Enhancing the security of these algorithms can be achieved by reducing and complicating the key size. This makes the method more suitable for securing data.
- The user's text is a bullet point. Due to the limited length of the encryption key, a brute force assault can be easily used to crack it. The primary issue with DES is the presence of a weak key. It does not provide protection against linear and differential attacks. The DES was not designed with software in mind, resulting in sluggish performance.

### **1.3.3 Access Control with Encryption - CP-ABE with RSA**

Attribute-based encryption (ABE) is a cryptographic technique that combines access control with encryption features. The ABE, or Attribute-Based Encryption, is a distinct form of public-key encryption that incorporates adaptable access controls for encrypted documents. The ABE technique is utilized in numerous applications and provides as the foundation for implementing the "one to many" sharing of files and fine-grained access control, whereas older systems only give "one to one" encryption. In the CP-ABE (Ciphertext Policy Attribute-Based Encryption) system, the size of cipher texts increases according to the amount of characteristics in the access structure, resulting in increased overheads for the receiver. Moreover, the computing costs of the receiver are increased because to the linear relationship between the number of pairing operations and the number of attributes involved in obtaining the policy during the decryption process. The privacy protection experiences significant shortcomings, particularly in the practical implementation of the Attribute-Based Encryption (ABE) approach, where there are restrictions in terms of bandwidth and processing resources. Several research methodologies exist for safeguarding privacy through access restriction, trust, and encryption. Nevertheless, the current cloud security techniques for accessing outsourced Big Data are disorganized and lack a coherent approach. Therefore, it is necessary to devise an efficient approach to enhance the safeguarding of privacy for cloud data. In order to address these problems, we have created an upgraded CP-ABE system with the RSA algorithm. This system is designed to ensure the security of Big Data that is stored in the cloud. (Basavarajegowda and Sundaram, 2021, pp.47-56)

### **1.3.4 Cryptographic Hash – Merkle Root**

Cryptography is the practice of safeguarding information by transforming the original message into an unintelligible one through encryption. A cryptographic hash function is a type of hash function that accepts a text message of any length as input and transforms it into a fixed-length sequence of encrypted letters that cannot be easily reversed. Hashing has long been a highly successful method frequently employed to compress data for quick retrieval and analysis, as well as to verify the integrity of information. A hash function typically refers to a function that compresses data, resulting in an output that is shorter in length compared to the input. Frequently, a function of this nature accepts an input of variable or almost variable length and transforms it into an output of a predetermined length, such as 160 bits.

Encryption primarily aims to guarantee the confidentiality of data that is either stored on a particular device or transmitted over the internet. A hash function guarantees data integrity by ensuring that even a minor alteration in the original message will result in a different digest. Furthermore, it should be practically impossible for a forger to construct a separate file with the same digest for a given file and digest combination.

The Merkle tree is a basic idea in the construction of a decentralized and secure network. In a decentralized network such as blockchain, all participants in the network are provided with a shared copy of the data. In contrast, a centralized network does not retain exclusive ownership of a single copy of records.

Nevertheless, the process of transmitting an identical copy of the updated record requires ongoing verification and maintenance of data integrity. The Merkle tree facilitates the organization of hashes to ensure network consistency and integrity.

## **1.4 Secure Cloud Storage**

Cloud computing lacks control over the data kept in cloud data centers. Cloud service providers have complete control over the data, enabling them to execute various destructive actions such as duplicating, eradicating, altering, and more. Cloud computing provides a certain degree of control over virtual computers. Due to the absence of control over the data, there are more significant security concerns compared

to the standard cloud computing approach. Encryption provides partial control over stored data, offering a level of security that is superior to plain data. The presence of virtualization and multi-tenancy in cloud computing can increase the potential for attacks compared to the traditional cloud architecture. The figure 1.6 presents several concerns that will be explained in a straightforward manner below.

### **1.4.1 Cloud Storage issues**

#### ***1.4.1.1 Data privacy and Integrity***

Despite the cost and resource management advantages of cloud computing, it is accompanied by certain security concerns. The general cloud computing paradigm must provide the integrity, confidentiality, privacy, and availability of data. However, this model is particularly susceptible to security concerns regarding these conditions. Due to its ease, the number of cloud users is steadily expanding, and the size of applications housed in the cloud is substantial. These circumstances provide increased security risks to clients utilizing cloud services. If any assault compromises a data entity, it will result in a data breach and illegal access to the data of all cloud users. The multi-tenant nature of cloud data might result in the loss of data integrity. SaaS providers, in particular, may also lack access to their technical data, posing a significant risk to data storage. In addition to these hazards, data processing also carries significant risks when data is transferred between several tenants. Virtualization allows for the sharing of physical resources across several users. This leads to the initiation of assaults by malevolent insiders from the Cloud Service Providers (CSP) and/or the company itself. These circumstances may enable a malicious user to carry out attacks on the stored data of other customers while processing their own data. Another significant risk arises when data is entrusted to third-party storage by the CSP. The process of generating and managing cryptographic keys for cloud computing lacks standardization and falls short of expectations. However, the use of standard and safe key management for the cloud does not enable the standard cryptography algorithms to effectively operate in a general cloud

computing paradigm. Cryptography can provide assurance against potential hazards in cloud computing.

#### ***1.4.1.2 Data recoverability and vulnerability***

The cloud ensures that users can access and allocate resources in a flexible and scalable manner, thanks to its resource pooling and elasticity features. The resource allocated to a specific user may be reassigned to another user at a later time. Regarding memory and storage resources, a malevolent user can employ data recovery techniques to acquire the data of prior users. The susceptibility of data recovery presents a significant threat to the security of sensitive user data.

#### ***1.4.1.3 Improper media refinement.***

The storage Media is disinfected for the following reasons:

- (i) The disk can be substituted with a different disk.
- (ii) There is no longer a requirement to maintain the disk.
- (iii) Decimation of services. Inadequate refining poses a significant threat to the integrity of stored data. In a multi-tenant cloud environment, it is not feasible to enhance the performance of a specific tenant beyond its initial state.

#### ***1.4.1.4. Data backup***

Data backup is crucial in both accidental and purposeful disasters. The CSP must regularly perform backups of stored data to ensure data availability. Indeed, the backup data must adhere to security protocols to prevent malevolent actions such as manipulation and unauthorized entry.

### **1.4.2 Identity Management and Access Control**

Access control and identity management are crucial for ensuring the integrity and security of data and services. It is imperative to retain a track record of user identity in

order to prevent unwanted access to stored data. The complexity of identification and access restrictions in cloud computing arises from the separation of the data owner and the stored data across distinct executive platforms. Within a cloud environment, different firms employ a diverse range of authentication and permission policies. Employing many techniques for authentication and authorization creates a complex scenario that evolves over time. Cloud resources exhibit dynamic characteristics and offer flexibility to cloud users. Additionally, IP addresses undergo constant changes when services are initiated or resumed in a pay-per-usage model. The cloud users are allowed to connect to and disconnect from cloud resources as needed, using an on-demand access policy. Efficient and effective access control and identity management are necessary for all of these functions. The cloud must efficiently support the rapid update and administration of identity management for users joining and exiting cloud services. Access control and identity management face several challenges, such as the vulnerability of weak credentials that may be easily reset, the risk of denial of service attacks that can lock an account for a specific duration, the limitation of weak logging and monitoring capabilities, and the potential threat of XML wrap up assaults on web pages.

#### ***1.4.2.1 Malicious Insiders***

An insider threat can arise from individuals who are employed by, contracted with, or are third-party business partners of an organization. Attacks in a cloud environment, namely at the Cloud Service Provider (CSP) level, result in the compromise of user data integrity, confidentiality, and security. This has the potential to result in the loss or unauthorized access of information in both settings. This attack is highly valuable and widely recognized by most organizations. Insiders have a range of attack patterns at their disposal due to their greater knowledge of an organization's internal data storage structure. Many organizations disregard this attack due to the difficulty of protecting against it and the impossibility of finding a comprehensive remedy. This attack highlights the significant danger of data breaches and the potential loss of confidentiality at both the organizational and cloud levels.

#### 1.4.2.2 Outside Intruder

Attacks originating from external sources are referred to as outsider assaults. Ensuring data security is a crucial concern in the field of cloud computing. Service providers lack authorization to access the physical security system of data centres. However, they are dependent on the infrastructure provider to ensure complete data security. Within a virtual private cloud scenario, the service provider is solely responsible for remotely establishing the security configuration, and the precise details of its complete implementation remain unknown. In this process, the infrastructure provider aims to achieve the following objectives:

The two main objectives are:

- (1) Ensuring confidentiality for secure data transfer and access, and
- (2) ensuring auditability. To prevent unauthorized access to sensitive data stored in the cloud by external intruders.

This research domain is neglecting contractual and legal matters.

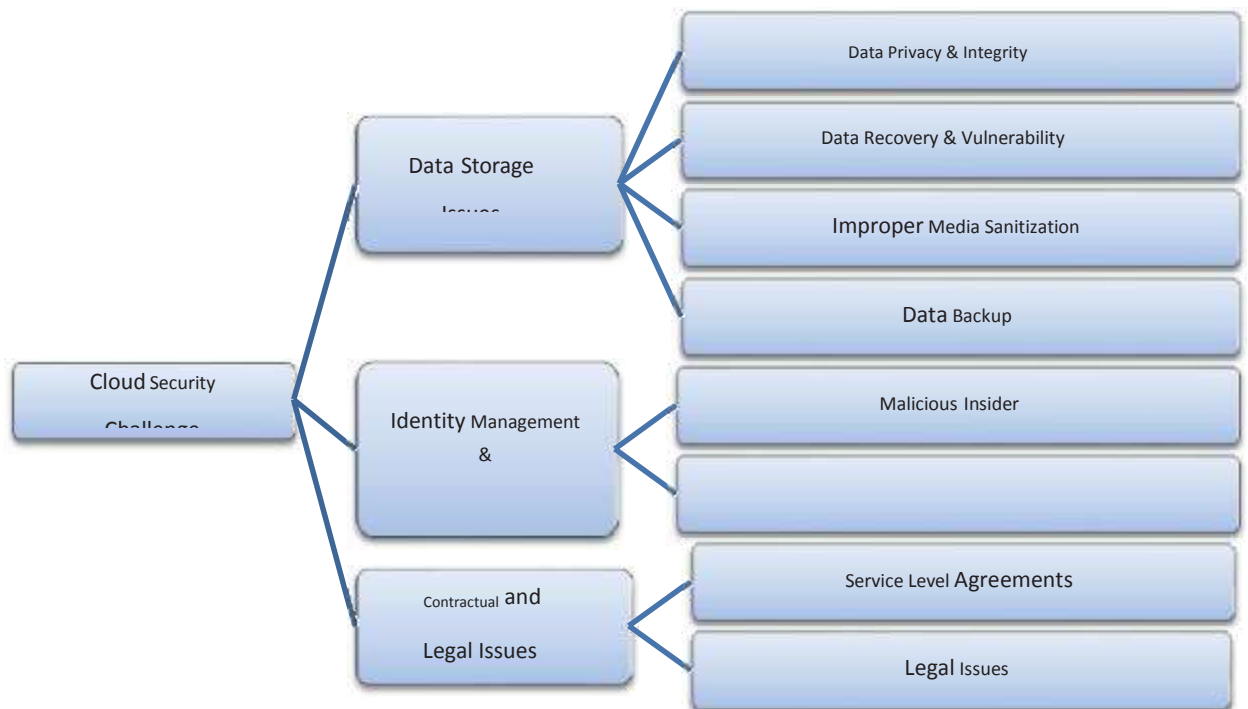


Figure 1.6: Cloud Security Challenges

## 1.5 Blockchain Technology

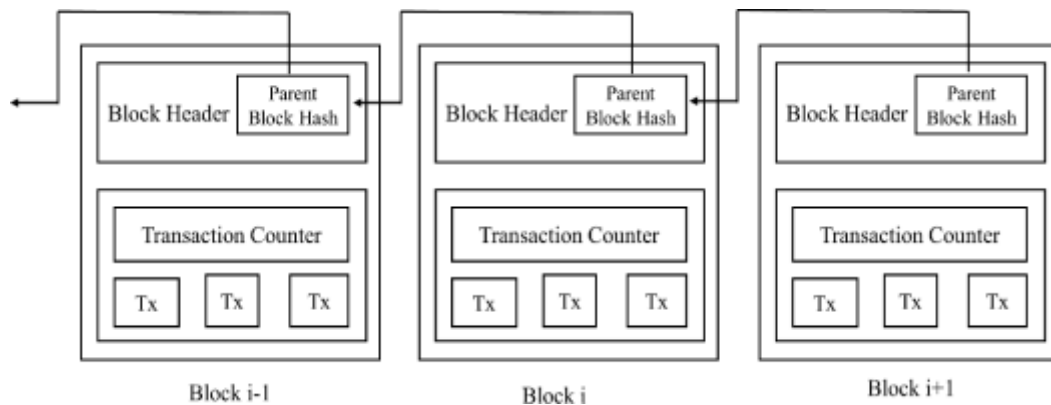


Figure 1.7: Blockchain Architecture

The advancements in the fields of encryption and distributed computing have given rise to a sophisticated technology known as blockchain. Blockchain is a decentralized system that duplicates and shares information over peer-to-peer networks. The concept of blockchain was first proposed by an anonymous individual named Satoshi Nakamoto, who developed Bitcoin as a means to facilitate direct trading of digital currencies without the need for intermediaries. Nakamoto pioneered the concept of a network consisting of interconnected nodes that collaborate to uphold a decentralized and impregnable database. The blockchain serves as the fundamental technology supporting cryptocurrencies, functioning as a communal public database or an ever-evolving record of all transactions. Blockchain might be considered a technological innovation and a significant development in the field of finance. It offers a resolution to every issue by utilizing a reliable ledger in a decentralized environment where perfect faith in actors, people, and computers is unattainable.

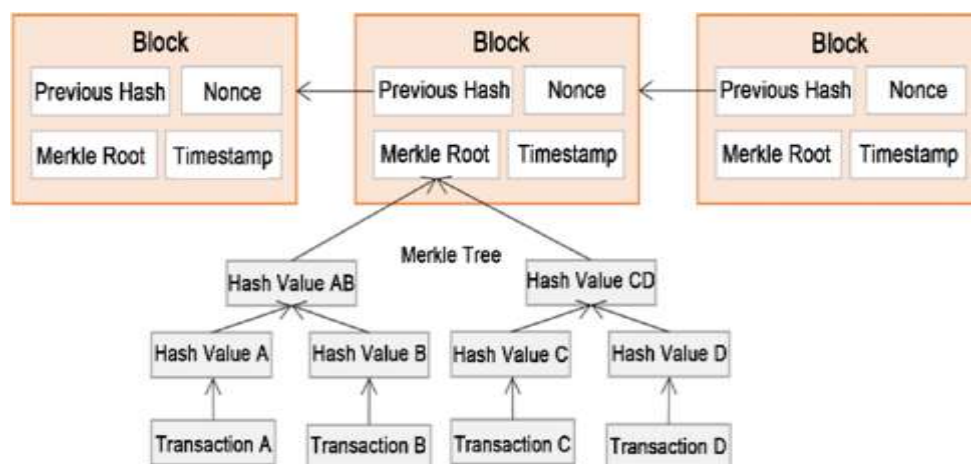


Figure 1.8: Structure of the Blockchain

Moreover, the blockchain employs a sequence of protocols and cryptographic techniques used on a shared network to safeguard the storage of data within a decentralized database consisting of verified blocks that encapsulate the data. The data is stored in the form of a sequential arrangement of blocks. Each block is differentiated by its hash sequence, which is determined by referencing the hash of the previous block, and it is also connected to the block that came before it. The sole deviation is the initial block, referred to as the "Genesis block," which lacks the hash value of the preceding block, also known as the ancestor block. The data is transmitted to the miners, who validate it by solving mathematical riddles and achieving consensus. The system's functionality is ensured by three key principles: 1) blocks and hashing, 2) mining, and 3) consensus. Figure 1.7 illustrates the architecture of the blockchain, while Figure 1.8 exhibits the structure of the blockchain, where each block includes the hash of the preceding block and the Nonce represents the answer to the proof of work puzzle. The timestamp denotes the duration it takes to generate a block, while the Merkle root verifies the authenticity of all the transactions (Austria, 2020).

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

### 1.5.1 Key Characteristics of Blockchain

- **Distributed:** The distributed environment employs a standardized protocol to ensure that every node receives each transaction and follows preset rules to aggregate the transactions into blocks after processing. The blockchain is specifically built to distribute and synchronize data across numerous networks.
- **Decentralization:** Decentralization is a fundamental characteristic of blockchain technology. This is because every node in the network maintains a complete record of all transaction data, eliminating the need for a central authority. This mitigates the susceptibility of a single point of failure. The blockchain network operates without a central authority and does not impose any service fees. Consensus techniques are employed to ensure the integrity of the data.
- **Consensus:** Consensus algorithms ensure data consistency and prevent the inclusion of inaccurate or misleading transactions in the blockchain network. In order to maintain the integrity of transaction data, it is imperative that all nodes reach a consensus by running the conventional consensus procedures. Furthermore, it is necessary to have a

statement or declaration between each member before one can carry out the transaction; the transaction should only be considered legitimate. This process is referred to as a consensus.

- **Anonymity:** Within the blockchain, each user has the ability to engage with a generated address. The system will maintain the confidentiality of the user's personal information, but authorized members will have access to the encrypted transaction details.

- **Traceable:** The blockchain records the time and digital signature of each transaction, allowing the organization to track back to a certain period and identify the corresponding party on the blockchain. As a result, every block is continuously linked to the previous block.

### 1.5.2 Smart Contract

The concept of smart contracts was conceived long ago, but it was only recently implemented. The concept of a smart contract was initially formulated almost two decades ago by Nick Szabo, a renowned cryptographer and academic. The core principle is to incorporate contractual concepts, such as liens and trusts, into computer components. The user offered four essential elements of contract design: evaluation, validation, privacy, and enforceability. According to Szabo's approach, the parties involved in a contract should assess their level of achievement by determining if the contract was executed or breached. Additionally, the smart contract safeguards the privacy of all involved parties and disseminates the necessary information as per the specified requirements. Ultimately, it will operate autonomously.

However, the required framework and specifications were unavailable at that particular moment. Previously, the concept of smart contract deployment was only theoretical, but now it has become achievable due to the progress in blockchain technology. Smart contracts are computer programs that encode and recreate real-world contractual agreements. A fundamental tenet of contracts is to establish a legally binding agreement between two or more parties, wherein each party is obligated to fulfill their contractual duties. The crucial factor to consider is that the contract will be regulated by a legitimate entity (organization). Smart contracts are removing the need for trusted intermediaries, who function as mediators between the parties involved in a transaction. The mediators

are controlled through the implementation of automated program execution in a decentralized blockchain network, with evaluation conducted by the network nodes. The smart contract facilitates transactions among parties who do not trust each other, involving mediator commission payments, reliance on trusted individuals, and the requirement for mutual interaction between the counterparties.

Smart contracts consist of a contract space, a balance, and code. Access to any node in a network can be obtained by posting a transaction to the blockchain. Once included into the blockchain, the smart contract code becomes immutable and cannot be altered. A consortium of miners assumes responsibility for overseeing the blockchain and executing smart contracts. The miners reach a consensus on the implementation outcome of the smart contract and subsequently update the blockchain. After being put into action, every contract is given a 160-bit address and is carried out using this address when a transaction is created. The smart contract leverages multiple platforms for application development. Ethereum is the predominant and pivotal platform for creating decentralized apps, encompassing a wide range of uses such as market forecasting, identity systems, and many economic applications. Bitcoin is a blockchain program that facilitates efficient and secure transactions, particularly in emerging areas, while also enabling the creation of new autonomous organizations. Bitcoin's main objective is to move capital, but its blockchain's immutability and openness have enabled the development of protocols that incorporate smart contracts. Smart contracts often utilize the Bitcoin scripting language to store data on the blockchain. In addition to Bitcoin and Ethereum, there is a growing number of alternative systems that have emerged from or are independent of the original Bitcoin network. These systems offer improvements and novel ways to address various challenges experienced in Bitcoin (Gad et al. 2022, pp. 6720-6738).

## **1.6 Data compression using P-2-P Technique**

P2P, or peer-to-peer, is a method of directly linking users to each other without the need for an intermediary server. It diverges from the conventional client-server architecture, where clients solely receive information transmitted by the server. In the peer-to-peer model, a peer functions as both a client and a server. Peers in the same network can immediately access and utilize a peer's restricted resources, such as network bandwidth

and disk storage. Currently, numerous widely-used applications utilize P2P architectures, which enable the availability of file transfer services. One can offer an additional compression option. That implies that the data will be transmitted in a compressed format from one user and received by the other user. The objective is to present a unified program that combines the functionalities of file sharing and file compression, eliminating the need for multiple applications to do these tasks. The existing algorithms are utilized for compression. It is a method employed to effectively condense the data prior to transmission. The application will be most beneficial in situations when there is a requirement for high-speed transfer of large amounts of data, regardless of its quality. (Nikita Darker, Disha Mishra, Swapnil Sonawane)

### **1.6.1 Tracking Server**

A tracking server functions as an intermediary between peers, facilitating their communication. The primary objective of it is to gather data about the peers and other statistical information. The server stores the data that a user wishes to download (Li, Zhang, Jia, Liu, Jin and Wu, 2019).

### **1.6.2 Client**

Initially, the client retrieves the initial receiving of peer data from the tracking server. Upon obtaining the peer information, the client initiates the process of downloading files. The files transferred through a peer-to-peer (P2P) technology are transformed into fragments or pieces. These fragments are compact files that are more convenient to transport. The purpose of this is to enable the downloading of each fragment from a separate peer, hence boosting the transfer speed and reducing the likelihood of corruption. The veracity of each file segment is verified by a cryptographic hash contained within the P2P file. Throughout the whole transfer, the client often monitors the tracker URL for peer data. Once the download is fully completed, the peer transitions into a seed.

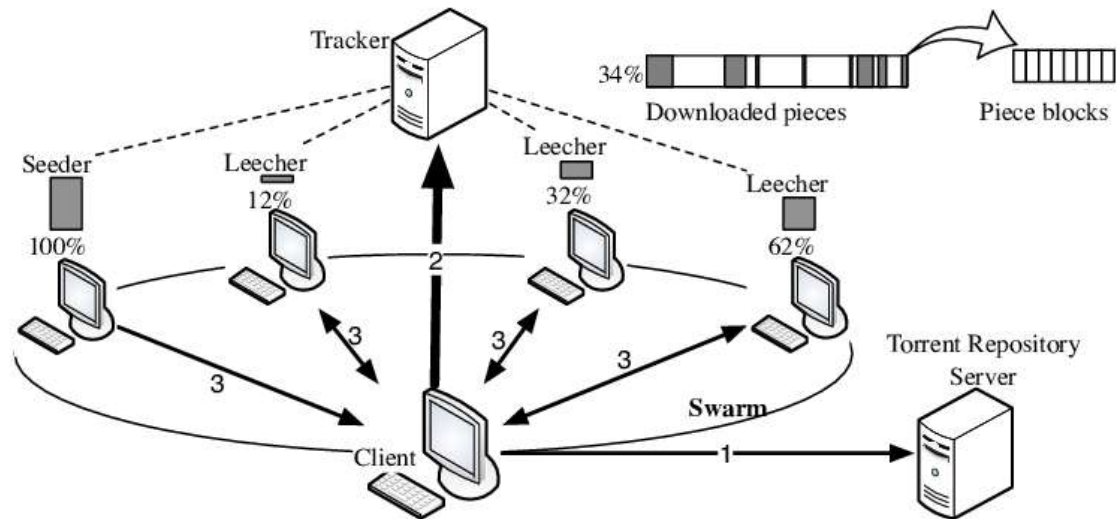


Figure 1.9: Torrent Architecture

The suggested framework utilizes the tracking server notion of P-2-P approach.

## 1.7 What is GFS?

GFS is a file system specifically intended to efficiently manage large volumes of data in batch processing environments. The system is designed to be distributed, meaning that several machines store duplicate copies of each file and numerous machines attempt to access the same file for reading or writing purposes. The original purpose of GFS was to cater to Google's specific needs for searching and indexing the internet. Essentially, GFS focuses on resolving the following issues:

- **Fault Tolerance:** Google employs commodity equipment due to their cost-effectiveness and ease of procurement. However, the software supporting GFS must possess resilience in order to effectively manage any failures of machines, disks, and networks.
- **Large Files:** It is presumed that the majority of files are of considerable size, specifically equal to or greater than 100 MB. While small files are compatible, they are not optimized for.
- **Optimize for Reads + Appends:** The system is designed to be optimized for reading, particularly for massive streaming reads, as well as for appending. This optimization is crucial due to the heavy reliance on these activities for web crawling and indexing.

- **High and Consistent Bandwidth:** Occasional slowdowns are OK, but the total volume of data transmitted through the system should remain constant. Once again, this arises from Google's activities of systematically browsing and categorizing web pages for the aim of organizing and storing information (Austria 2020).

## 1.8 What is HDFS?

Hadoop Distributed File System (HDFS) is a distributed file system specifically designed to operate on inexpensive hardware. It shares numerous resemblances with current distributed file systems. Nevertheless, the distinctions from other distributed file systems are substantial. HDFS exhibits a high degree of fault tolerance and is specifically engineered to be implemented on inexpensive hardware. HDFS offers efficient access to application data with a high data transfer rate, making it ideal for applications that handle extensive data volumes. HDFS modifies certain POSIX specifications in order to facilitate continuous access to data in the file system. HDFS was initially constructed as the foundational framework for the Apache Nutch web search engine initiative. HDFS has been designated as a subproject of Apache Hadoop.

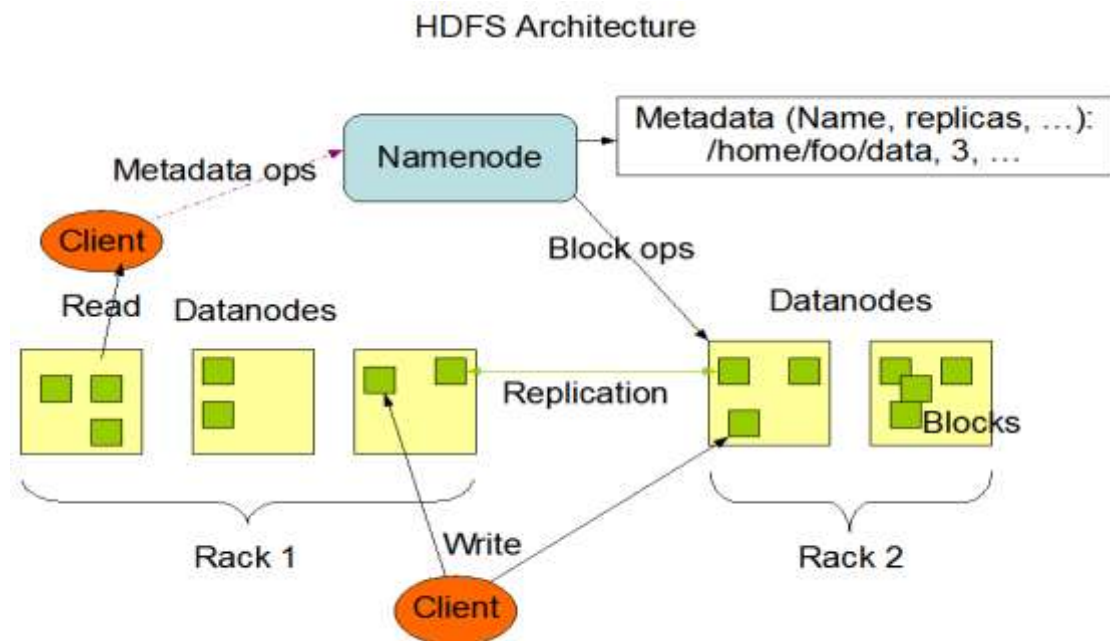


Figure 1.10: HDFS Architecture

HDFS adheres to a master-slave design and consists of the following components.

### **1.8.1 Namenode**

The namenode is a type of commodity hardware that houses both the GNU/Linux operating system and the namenode software. This software is capable of being executed on standard, readily available hardware. The system with the namenode functions as the primary server and does the following tasks:

- Manages the namespace of the file system.
- Controls the client's access to files
- Executes file system actions such renaming, closing, and opening files and directories.

### **1.8.2 Datanode**

The datanode is a commodity hardware having the GNU / Linux operating system and datanode software. For every node (Commodity hardware/System) in a cluster, there will be a datanode. These nodes manage the data storage of their system.

- Datanodes perform read-write operations on the file systems, as per client request.
- They also perform operations such as block creation, deletion, and replication according to the instructions of the namenode.

### **1.8.3 Block**

Typically, the user data is kept within the files of the Hadoop Distributed File System (HDFS). A file in a file system is typically partitioned into many segments and/or stored in separate data nodes. These parts of files are referred to as blocks. Put simply, a Block is the smallest unit of data that HDFS can read or write. The default block size is 64MB, but it can be adjusted according to the requirements by modifying the HDFS configuration.

### **1.8.4 Features of HDFS**

- It is well-suited for distributed storage and processing.
- Hadoop offers a command interface for interacting with HDFS.
- The namenode and datanode include integrated servers that allow users to conveniently monitor the cluster's status.
- Streaming capability to access data in the file system.
- HDFS has functionality for managing file permissions and authentication.

## **1.9 Motivation for the Study**

In today's world there is a significant surge of data, and there is a universal desire to store, safeguard, and transmit this data. Cryptographic techniques are employed to ensure the security of electronic data during network transfers. Therefore, it is crucial to choose an encryption technique that offers both high performance and strong security. In order to mitigate data loss and enhance transmission speed, a variety of compression methods are employed to decrease the data size during transmission. Lossless compression algorithms are typically employed when transferring critical data where any loss of information is unacceptable. When a compressed file is encrypted, it offers enhanced security and a higher transfer rate across the network compared to encrypting and sending an uncompressed file. Therefore, it is necessary to examine several asymmetric key cryptography algorithms across different parameters in order to comprehend the elements that can impact the efficiency of these algorithms. Additionally, determine whether the file has to be compressed before to encryption. To optimize the compression process and enhance the efficiency and security of data transfer, it is necessary to select the most appropriate compression algorithm based on the data type and size. This will help minimize the time required for compression and decrease any associated overhead.

Due to the prevalence of the digital era, communication mostly relies on the transmission of digital information over networks. Communication is commonly conceptualized as a sender who delivers a digital file to a recipient. This file is transmitted from a starting point to a destination, and in order to facilitate fast and instantaneous communication, an encoding technique is required. This strategy should be both efficient and straightforward, while also ensuring security. This communication can be structured using a framework consisting of two distinct and potentially contradictory operations. These operations must be applied to the original file in order to ensure both efficiency and security. The two operations encompassed in this context are data compression and encryption. The concerns of data transmission, including confidentiality, data security, and data loss, continue to be significant reliability issues. In order to address this issue, a comprehensive framework has been developed to ensure the confidentiality, integrity, authenticity, and non-repudiation of data.

Secure storage solutions necessitate substantial CPU capabilities and exceptionally high speeds. The framework will attain equivalent performance by employing a Distributive architecture, ensuring that the workload is evenly distributed across all participating nodes in the storage process.

The file or data can be effortlessly uploaded and downloaded on distributed storage, which is constantly accessible to the user. The framework will accomplish this by employing a variety of techniques like as compression, encryption, file splitting, hashing, and block chain.

Cloud-based data storage is a relatively new trend that provides both free and extensive storage capacity. While the cloud offers numerous benefits, consumers are primarily concerned about security due to the storage of extremely sensitive data in the cloud. Data integrity is a crucial necessity in a cloud storage system that is built on sharing. Furthermore, the access control in the current cloud model necessitates one or more fully trusted attribute or central authority to uphold the access policy. If the central authority is dismantled, the entire system is disrupted. Hence, the task of attaining security and privacy functionalities for compressed and encrypted data is a substantial obstacle. Decentralized systems play a crucial role in ensuring integrity checking and

access control, hence eliminating the potential threat posed by a trusted central authority. Hence, in order to enhance the efficiency of current applications, it is necessary to combine cloud storage and blockchain technology to propose a distributed cloud storage architecture based on blockchain. This architecture would offer secure and dependable cloud storage services for end-users.

### **1.10 Need of the Study**

Cloud computing now faces various issues, namely in the areas of scalability, interoperability, and multi-tenancy. Cloud computing design has presented numerous issues, particularly in regards to safeguarding the security of outsourced data.

Cloud storage is a method of sharing resources, namely for exchanging data between the data owner and authorized clients. However, this sharing can also pose a danger of data breach or leakage. Securing data in the cloud is a challenging task if the client lacks faith in the service provider. The client's willingness to trust the provider's procedures without question can be hindered by the presence of malevolent insiders, including cloud administrators who have easy access to the data.

To solve the aforementioned challenges, the research will develop a framework that offers a novel technique for ensuring data security.

### **1.11 Scope and Contribution of the study**

The primary goal of the thesis is to create a safe architecture using blockchain technology for storing data in the cloud. This architecture will include features such as authentication, integrity checks, and access control to address the drawbacks and obstacles. The suggested architecture utilizes various compression algorithms to reduce the size of data and diverse cryptographic algorithms to establish a key generation mechanism, attain confidentiality, implement a data access control system, and guarantee the integrity of cloud data.

The architecture employs a Honeybee optimization algorithm to maximize resource utilization in the cloud storage system and minimize transaction processing time in

terms of execution and response time. Through the utilization of blockchain technology, it attains decentralization along with robust security, eliminating the requirement for a trusted central authority.

The contributions of the thesis are outlined below:

1. The purpose of the literature review is to investigate and emphasize the methods of utilizing blockchain technology to ensure the security of cloud storage systems. Additionally, it indicates the diverse domains where blockchain can be integrated with the cloud storage system.
2. The suggested cloud storage system utilizes a distributed blockchain technology to ensure security, eliminating the need for any trusted authority. The suggested architecture offers a comprehensive decentralized solution to generate key-related information without the need for any centralized authority, utilizing the blockchain framework.
3. The suggested work utilizes the Huffman Coding and CP-ABE algorithm to guarantee the qualities of confidentiality, integrity, and availability. Confidentiality is ensured by the utilization of encryption techniques to encode user data. The integrity feature is accomplished by the utilization of the blockchain-based Merkle root concept.
4. The cloud storage architecture that is based on blockchain utilizes the Honeybee optimization algorithm in order to optimize resources and decrease transaction response and execution time on a cloud storage system.
5. The performance and security analysis is assessed by implementing the proposed design and demonstrating the system's capabilities through experimental findings.

The framework encompasses the security sector and will offer storage for many data kinds in a highly efficient and cost-effective manner. The number of individuals that benefit from this system is limitless. Large enterprises do not need to spend millions of dollars on costly cloud storage. This framework efficiently does the same task at a low cost by utilizing a distributive framework. Efficient storage can be supplied, ensuring rapid dissemination of information to all participating nodes. The average internet user, who is occasionally deceived by security authorities, will be protected. He will have access to the most up-to-date and advanced storage technology at the most affordable price. This study has the potential to assist governments, schools, colleges, hospitals,

and organizations of all sizes. The study will focus on both ensuring secure storage and achieving efficient and rapid data retrieval.

### **1.12 Research Methodology used in the Study**

A survey in the format of a Google Form is distributed to individuals working in diverse organizations, occupying various roles such as industry professionals, security auditors, ethical hackers, bankers, developers, database administrators, senior managers, HR professionals, university lecturers, students, and others. Their security problems were analyzed. The following are the key findings:

- Many of them have frequently experienced Threats might arise from any source, whether it be from within or outside the shared environment. Choosing between transferring sensitive data or keeping it on premise is a critical issue for individual users and small to medium-sized companies.
- The volume and intricacy of data are expanding at a rapid pace, in tandem with advancements in computing and storage technologies. The administration of data is thus becoming progressively significant. The storage and transmission of digital data pose significant challenges.
- The companies are allocating a significant amount of resources to maintain security, although they do not feel entirely secure.
- They frequently express concern about the potential loss of vital organizational data and resources to less skilled hackers.
- Security concerns in cloud computing encompass data security, backup, network traffic, file system integrity, and host security.

The hypothesis was formulated by selecting multiple compression techniques from various sources and subjecting them to testing using a diverse range of data. Some individuals were able to effectively compress the data, resulting in significant space reduction, whereas others were not successful in doing so. All of these compression algorithms belonged to broad categories. Subsequently, various encryption techniques underwent testing on compressed files/data. The selected compression algorithms

include Huffman coding, Shanon-Fanno, and LZW. For our crypto compression model, we have chosen the asymmetric encryption algorithm from the encryption algorithm.

Thus, it was determined that the current solutions are inadequate compared to the sophisticated tools available to hackers. Therefore, it is necessary to create a comprehensive framework that can successfully address all of these dangers. The requirement is for the solution to be both rapid and cost-effective. The distribution of power should be decentralized, with authority being in the hands of the general public rather than being concentrated in the control of Cloud Service Provider businesses.

The research methodology is a mixed mode, encompassing both qualitative and quantitative approaches. The framework was developed and will be validated by testing it with different file types. The model will undergo initial training by providing it with known samples, allowing it to autonomously acquire the necessary components. Subsequently, it would undergo testing to ascertain its accuracy and validity. Ultimately, the usefulness and performance of the product would be assessed through the analysis of randomly selected samples obtained from the natural environment.

### **1.13 Structure of the Thesis**

#### **Chapter 1. Introduction**

The initial chapter provides an overview of the study. It categorizes the compression techniques according to different sorts of files. One must make a decisive decision among them. Additionally, it categorizes encryption algorithms and chooses one. Subsequently, the file is divided and a hash function is applied to ensure the security and integrity of the files. Comprehend the extent and impact of the study as well as the research approach employed.

#### **Chapter 2. Literature Review**

The second chapter entails a comprehensive literature review of various topics, including compression techniques, encryption techniques, file splitting methods, hashing concepts, file system security vulnerabilities (CVEs), blockchain and peer-to-peer (P2P) techniques, emulator functionality such as VMware Workstation, cloud architecture, blockchain understanding, and the study of digital currencies such as Bitcoins and Ethereum.

### **Chapter 3. Research Methodology**

The third chapter delineates the research methodology employed for the purpose of proposing and constructing the framework. The study's purpose, scope, and contribution are comprehended, assessed, and analyzed. The hypothesis is formulated through interviews and testing with various file kinds obtained from real-world sources. The framework has been meticulously created and all the steps of the suggested framework have been comprehensively understood. Algorithms have been developed for each stage, and formulas have been established to calculate metrics such as compression ratio, encryption rate, time complexity, and efficiency.

### **Chapter 4. Analysis and Interpretation**

The fourth chapter focuses on the analysis and interpretation of data. A thorough examination is conducted, encompassing quantitative analysis, qualitative analysis, and comparative study. I compared GFS and HDFS and comprehended their influence, methodology, and characteristics. Constructed each component of the framework, commencing with the Compression module and extending to encryption, network, and virtualization modules. Illustrates the comprehensive functioning of the framework in a decentralized way and demonstrates how the blockchain idea can be utilized for database tracking, as well as for incentivizing and acknowledging participants.

### **Chapter 5. Findings and Conclusions**

The fifth chapter presents the findings and conclusions of the investigation. Each component of the model was developed using a range of file formats to determine the accuracy of the framework's output. The severity is assessed based on the computed likelihood. The study has been completed and the limitations have been listed. The report also examines the other areas for investigation and identifies further research opportunities.



**Figure 1.11: Structure of Thesis**

## **1.14 Keywords used**

Compression algorithm, encryption algorithms, lossless compression, lossy compression, cloud storage, cryptography, asymmetric-key, symmetric-key, block cipher, stream cipher, cipher text, hashing, blockchain, P-2-P, Shannon-Fano, LZW, tracking server, IPFS, smart contracts, access control, Authentication, CSP, integrity, CP-ABE, RSA, malicious insider, outside intruder, data recovery, data backup, data privacy, Huffman coding. Distributive architecture, decryption, RLE, data vulnerability, public key, private key, SHA, SaaS, confidentiality, authenticity, file sharing, virtualization, VMware Workstation, Digital Currency, Ethereum, GFS, HDFS.

# CHAPTER 2

## LITERATURE REVIEW

### 2.1 Introduction

Cloud computing is a rapidly growing field in the current digital age. Cloud computing refers to the provision of resources or services over the internet by cloud providers, in response to users' requests. It provides all services via the internet according to user demand, including as operating systems, network hardware, storage, resources, and applications. Every firm is storing its data on the cloud, utilizing the storage service offered by the cloud provider. Therefore, it is imperative to safeguard the data from illegal intrusion, alteration, or disruption of services, among other potential threats.

Cryptographic techniques are employed to guarantee the security of data during network transfers. In order to prevent data loss and enhance transmission speed, various compression methods are employed to decrease the data size during transmission. The data files are initially partitioned into several shards of equal size and then spread across multiple servers.

In order to address concerns related to data integrity, the file is partitioned into numerous shards to calculate Merkle root. The calculated Merkle root stored on the blockchain. This function utilizes a *SHA256* hashing technique to produce a hash and root hash. The framework has the capability to engage with the blockchain. This guarantees the integrity of cloud data that is stored. Users have the ability to request an integrity check for the file.

### 2.2 Analysis of Literature Survey:

The objective of doing a literature survey is to gain a comprehensive understanding of the prior research conducted in the selected subject area. A thorough examination of numerous articles was conducted to ascertain the advancements made by other researchers in this particular domain. Upon commencing our investigation on this subject, we encountered numerous domains in which our understanding was limited or

non-existent. It was necessary to examine the principles of compression, encryption, and safe cloud storage. To comprehend the security methodologies employed by researchers in cloud storage thus far. What advantages do these techniques offer and what limitations do they have? In order to gain a comprehensive understanding of various compression and encryption strategies, it is important to analyze their distinct behaviors and evaluate their respective strengths and weaknesses. This knowledge will enable more effective countermeasures against these approaches. Additionally, it is important to understand the concepts of centralized and distributive security techniques in order to assess their respective advantages and disadvantages. The literature review is segmented into the subsequent sections:

1. Data compression techniques,
2. Data encryption techniques.
3. Cloud computing Concept,
4. Understanding cloud architecture and working,
5. File System Security Related CVE,
6. Security aspects for data,
7. Secure cloud storage,
8. Understanding Blockchain technology,
9. Working of P-2-P Torrents,
10. Working of emulator like VMware workstation,
11. Secure Cloud Framework,
12. Distributed file storage,
13. Centralized storage system,
14. Study of digital currencies like Bitcoins and Ethereum.

## **2.3 Data Compression Techniques**

**2.3.1** Jayasankar, Thirumal, and Ponnurangam (2021, pp. 119-140) provide an explanation of the commonly employed data compression techniques, categorized according to quality, coding schemes, data type, and application. A comparative study is conducted to determine the contribution of the researched approaches in terms of their characteristics, underlying concepts, experimental aspects, and constraints. The text provides a detailed explanation of the quality-based, coding scheme-based, data type-based, and application-based methods, along with their respective advantages and

disadvantages. This information is highly valuable for comprehending data compression strategies.

**2.3.2** Thajel and Hashim (2020) devised a method for compressing documents. The lossless approach is composed of two components: the compression part and the decompression part. The compression component consists of several fundamental stages, including pre-processing, block processing, run length encoding (RLE), substitution of maximum values with unused values, level minimization, delta encoding, compression of ones values, and encryption. Once the encryption procedure is finished, the results are saved in two distinct binary files with the ".bmp" suffix. One of these files is the header file, which serves as a key for the second file that holds the compressed data. This methodology was applied to twenty documents and compared with various compression methods such as RLE, JPEG, TIFF, and PNG. The experimental findings demonstrated that the proposed strategy outperforms the other methods in terms of compression ratio.

**2.3.3** In their study, Lal and Rasheed (2020, pp. 590-597) examine lossless data compression algorithms and provide a comparative analysis of these methods. Data compression is a procedure that decreases the size of data by eliminating redundant information. It minimizes the duplication in data representation to decrease the amount of storage needed for that data, therefore also reducing the communication cost by efficiently utilizing the available bandwidth. Data compression is a crucial application in the field of file storage and distributed systems. Various data compression techniques are used for diverse data formats, including text, audio, video, and image files. There are primarily two types of data compression: lossy and lossless. However, in lossless data compression, it is essential to maintain the integrity of the data. This study provides a comprehensive analysis of lossless compression techniques, including an examination of their benefits and drawbacks.

**2.3.4** Sankalp Shukla, Ritu Gupta, Dheeraj Singh Rajpu, Yashwant Goswami, and Vikash Sharma (2022, pages 59-69) examine the effectiveness of different lossless compression methods used on uniformly quantized audio signals. The objective of this work is to elucidate a novel approach to compressing audio signals by employing lossless compression techniques. The audio signal is initially converted into text using uniform quantization with varying step sizes. The text is compressed using lossless

compression algorithms such as Run length encoding (RLE), Huffman coding, Arithmetic coding, and Lempel-Ziv-Welch (LZW) coding. An analysis is conducted to evaluate the performance of different lossless compression algorithms. This evaluation is based on four primary parameters: compression ratio, signal-to-noise ratio (SNR), compression time, and decompression time. The examination of the above parameters was conducted following the uniform quantization of the audio files using various step sizes. The study demonstrates that the LZW coding has the potential to serve as an effective alternative to the MP3 lossy audio compression algorithm for efficiently compressing audio signals.

## **2.4 Data Encryption Techniques**

**2.4.1** In their work, Krishna and Manikandan (2020, pp. 321-327) examined the topic of cryptography, focusing on symmetric key algorithms like DES, Triple DES, AES, Blowfish, as well as asymmetric key algorithms such as RSA, Diffie Hellman, and ECC. Symmetric-key cryptography involves encryption methods when both the sender and receiver possess the same key. Symmetric key ciphers can be classified into two categories: block ciphers and stream ciphers. A block cipher encrypts the input data of a stream cipher in the form of plaintext blocks, rather than individual characters. Asymmetric cryptography is significantly slower compared to symmetric cryptography. In asymmetric cryptography, one key is used for encryption and another key is utilized for decryption. The asymmetrical systems provide enhanced stability compared to symmetrical systems. Asymmetric encryption is a recently developed and intricate method of encryption. The system is complex as it employs two cryptographic keys to ensure the security of the data. These types of keys are commonly known as a public key and a private key. The Public Key, as implied by its name, is accessible to all individuals who desire to transmit a message. Conversely, the individual who possesses the public key ensures that the private key is stored in a secure location.

**2.4.2** In this study, Wang (2022) examines, evaluates, and contrasts different encryption techniques. A proposed hybrid cryptosystem utilizes the MD5 hashing algorithm, the AES encryption algorithm, and elliptic curve cryptography for key generation. This paper presents an overview of commonly employed encryption algorithms, including the fundamental concepts, techniques, and features of symmetric

and asymmetric encryption algorithms. Additionally, it presented the attributes of the AES algorithm and examined its encryption, decryption algorithm, and security. Consequently, this text presents an introduction to the essential functions of the elliptical curve cryptosystem, followed by a thorough analysis of its performance. Notably, this system offers several advantages over RSA and DSA, including superior speed, enhanced convenience, and increased resistance to decryption. This study utilizes state-of-the-art encryption technology to provide a network file encryption system. The system offers a fundamental mode and serves as a realistic foundation for secure transmission of network files.

**2.4.3** Sharma and Mishra (2021, pp. 274-277) provided a concise overview of cryptographic approaches and introduced a novel cryptographic algorithm for enhancing data security. This algorithm may effectively safeguard data in cloud computing applications. This study presents a novel and efficient cryptographic method that is characterized by its simplicity and minimal execution time for both encryption and decryption processes. Furthermore, the suggested algorithm is readily applicable in practical projects involving the Encryption and Decryption Process.

**2.4.4** Tushar, Aniket Sharma, and Ankit Mishra (2021, pp 274-277) provide a concise overview of cryptographic approaches and introduce a novel cryptographic method for enhancing data security. This method can effectively safeguard applications on cloud computing platforms. Additionally, a novel and streamlined Cryptography algorithm has been developed. This algorithm is characterized by its simplicity and minimal execution time for both encryption and decryption processes. Moreover, the suggested technique can be readily implemented in a practical Encryption and Decryption Process for real-world projects.

**2.4.5** The numerical value is 2.4.5. Yang Ba, Xuexian Hu, Yue Chen, Zenghang Hao, Xuewei Li, and Xincheng Yan (2021) suggest a blockchain-based approach called BCP-ABE-PHAS that incorporates partially hidden access structures. This scheme aims to provide precise access control while safeguarding user privacy. Initially, it presents a proficient CP-ABE technique incorporating partially concealed access structures, wherein the size of the ciphertext remains constant. In order to aid in data decryption, a garbled Bloom filter is designed to facilitate the rapid identification of wildcard positions within the access structure. It proposes a data storage architecture

that combines blockchain technology and the interplanetary file system to enhance storage efficiency and system scalability while preserving data integrity. Ultimately, it utilizes smart contracts to facilitate a transparent method of storing and sharing data, eliminating the need for involvement from third parties. The suggested BCP-ABE-PHAS method has been analyzed for security and performance, and the results indicate that it may effectively maintain policy privacy while requiring little storage and computing resources.

## **2.5 Cloud Computing Concepts**

**2.5.1** Sheth, Bhosale, and Kadam (2021, pp. 87-92) examine various Cloud Computing products, such as AWS Elastic Compute, Google Cloud Engine, and AWS Lambda. Notable cloud computing services include Amazon Web Services, Google Cloud Platform, and Microsoft Azure. Cloud Computing exhibits a high degree of flexibility. The text also covers the three primary service models of cloud computing: Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS). This text explores both the advantages and disadvantages of Cloud computing.

**2.5.2** In publication, Xiao (2022, pp.93294-93314) suggests the implementation of a cloud computing-based educational information management platform. The text presents the concept of layered architecture and highlights the benefits of cloud computing technology. The design of the educational information management platform in colleges and universities incorporates the benefits of cloud computing technology. The architecture model of this platform is based on cloud computing technology and consists of four layers: user access layer, SaaS layer, PaaS layer, and IaaS layer. The functional modules of the platform design are then analyzed. The JSP system page was constructed on a Linux operating system server terminal, using Apache as the Web application server. It established a SQL Server database that comprises information tables for users, departments, teachers, courses, teaching resources, and statistical analysis. Lastly, an analysis is conducted on the practical impact of the school information management platform that utilizes cloud computing technologies. The experimental results indicate that the platform underwent testing and achieved a system data operation accuracy of 90%. Furthermore, the system

demonstrated stable and seamless performance for a duration of over 2 hours. Additionally, the system was capable of accommodating up to 200 concurrent users, with a response delay of less than 3 seconds. The educational information management platform is designed to cater to the specific needs of instructors and students in schools, providing a user-centric experience and offering robust technical support for innovative teaching practices.

**2.5.3** Bangar, Pallavi, and Shetty (2022, pp. 14-22) present a concise overview of cloud computing assessments derived from an analysis of over 30 articles on the subject. The findings of this study depict the condition of the IT sector prior to and following the advent of cloud computing. Cloud computing refers to the delivery of network, storage, and server resources over the internet, either on demand or through a pay-per-use model. While cloud computing has been beneficial to the Information Technology industry, further research and development in this field is still necessary. This work provides a comprehensive overview of the concept of cloud computing and addresses the most cutting-edge research topics in the field.

**2.5.4** In Bola Abimbola's (2021) work, the origins of cloud computing are examined, followed by an exploration of its fundamental notion. The discussion will also cover the future of Cloud computing and the latest trends in its growth. Hence, it will offer a thorough synopsis of cloud computing, including its definition, the methods by which its services are delivered, and the various service models that exist for tailoring cloud computing services to meet customer needs.

## **2.6 Understanding cloud architecture and working**

**2.6.1** Qureshi (2021, pp. 16-23) examined the cloud computing architecture and its impact on reducing operating expenses. Resources can be efficiently distributed and reallocated in a cloud environment. Thus, there is no requirement for a service provider to deliver additional capacity in response to high levels of demand. During periods of low demand, it is possible to allocate resources in a way that reduces operational expenses, resulting in significant cost savings. The writers have emphasized that public cloud is a captivating framework for organizing and delivering services that has recently emerged. The public cloud is revolutionizing the digital technology landscape by finally fulfilling the long-awaited promise of utilitarian computing. Although cloud

technology offers numerous advantages, the existing techniques are not advanced enough to fully achieve their potential. Several crucial topics in this domain, including automated service selection, energy efficiency, and data privacy, are currently under investigation in academic circles.

**2.6.2** Hu, Fang, Yang, and Zhang (2020) examine the present state of development of enterprise distributed business information systems and identify their existing issues. They integrate cutting-edge theory and technology of Cloud Computing and SOA to devise a Cloud Computing architecture model based on SOA. This model is then implemented in enterprise distributed business systems. This solution effectively addresses the issues of the current system and significantly enhances the quality and efficiency of system development. It seamlessly integrates the existing legacy systems and cloud services offered by the cloud platform, hence optimizing resource use. It has the ability to provide an open information environment that allows for the system to grow and adapt to changing business needs. It holds significant importance in directing the development of enterprise information systems.

**2.6.3** Bajaj, Jatain, Chaudhary, and Nagpal (2021, pp. 70-73) acknowledge the forthcoming concerns and challenges that the cloud storage business will confront. They also aim to define and evaluate the new paradigm for academics in the field of cloud storage. This article primarily addresses the obstacles and potential solutions in cloud storage, specifically regarding data security and data management. Encryption techniques and blockchain technology can be utilized to bolster the security of cloud storage, hence increasing its resilience. Authentication methods for access control might introduce an extra security measure to detect and apprehend unauthorized individuals attempting to access the data stored on the cloud. The discourse over the future prospects of cloud storage and the diverse range of potential has concluded. It can be inferred that the cloud computing environment is a rapidly expanding technology that is supplanting the traditional computing environment. Advanced approaches and secure access features are being used to preserve the client's sensitive data.

## 2.7 Decentralized Data Storage

**2.7.1** Ganesh J., Priyadharshini R., Swetha K., and Vaishnavi G. (2022, pp 330-335) introduced a system that encrypts the user's file and distributes it among various peers in the network using the IPFS (InterPlanetary File System) protocol. IPFS generates a hash value. The hash value signifies the file's location and is recorded within the blockchain. Additionally, Aslo prioritizes decentralized and secure storage of data, ensuring high availability and effective utilization of storage resources. Here is an analysis on utilizing Blockchain technology as a decentralized data storage system. This study aims to demonstrate the advantages of decentralized data storage using blockchain technology by employing resilience attacks to conduct essential operations on the proposed decentralized structure.

**2.7.2** In their 2021 study, Kai He, Chunxiao Huang, Jiaoli Shi, Xinrong Hu, and Xiying Fan presented a technique for verifying the integrity of dynamic data in cloud servers using a blockchain and T-Merkle hash tree. A decentralized system is suggested as a means to remove the limitations of prior centralized schemes. The data tags are created using the ZSS short signature mechanism and then stored on the blockchain. A refined verification technique has been developed to ensure the integrity of cloud data by offloading computation from a verifier to a cloud server and blockchain. Additionally, a storage architecture known as T-Merkle hash tree, which is constructed using a combination of T-tree and Merkle hash tree, has been developed to enhance storage efficiency in blockchain and enable binary search within the chain. Furthermore, it accomplishes efficient and secure dynamic update operations on the blockchain through a method of only adding new information without modifying existing data. In addition, we may expand our scheme to include batch verification, which allows for the simultaneous processing of large jobs. This results in enhanced efficiency and lower communication costs. Ultimately, a prototype solution was developed using Hyperledger Fabric to authenticate the scheme. The suggested scheme has been evaluated by security analysis and performance studies, which demonstrate its strong security and efficiency.

**2.7.3.** In their study, Khan et al. (2022) aimed to assess the security of cloud systems utilizing the Ethereum blockchain. The authors also provided a brief overview of cloud

computing, highlighting its benefits and drawbacks. Blockchain technology briefly showcased the concept of a decentralized cloud. Moreover, the preceding papers were evaluated and displayed in a tabular format. It is evident that there are still areas of research that need to be addressed in the realm of blockchain-based cloud computing platforms. This paper presents a methodology for storing data securely on a decentralized cloud using the Ethereum blockchain.

**2.7.4.** In their publication, Racin Nygaard, Hein Meling, and John Ingve Olsen (2023, pp 165-173) introduce a new protocol called the Storage Upkeep Protocol (SUP). This protocol employs proof-of-storage queries to identify any missing pieces. The evaluation of SUP was conducted on a substantial cluster consisting of 1000 peers that were running an up-to-date version of Ethereum Swarm. The contributions encompass the development and execution of SUP, as well as an examination of Swarm's redundancy characteristics. Our assessment demonstrates that SUP greatly enhances the efficiency of bandwidth use and reduces the time required for data maintenance when compared to the current solution. Under typical circumstances, the use of SUP can result in a bandwidth savings of up to 94% and a reduction in the time required for re-uploading by as much as 82%. Although the cost of employing SUP depends on the pricing policy of the storage network's bandwidth, it can potentially lower the overall expenses associated with data maintenance.

## **2.8 File System security related CVE's**

**2.8.1.** In their publication, Sun, Li, Xu, and Huang (2022) conduct a comprehensive analysis of the security of contemporary file systems, focusing on vulnerabilities. More precisely, it gathered 377 vulnerabilities related to file systems that were recorded in the CVE database during the course of the last two decades. Classify them based on four aspects, encompassing the causes of vulnerability emergence, the methods by which vulnerabilities can be exploited, the potential consequences that may arise, and the approaches employed to rectify the vulnerabilities. By doing so, one can develop a comprehensive comprehension of the attack surfaces encountered by file systems, the threats posed by these attack surfaces, and the effective and ineffective measures for preventing attacks on file systems. Our study aims to provide valuable insights for the

future development of file systems, the improvement of file system security, and the implementation of effective vulnerability mitigation measures.

**2.8.2** In their study, Gousiya Begum, S. Zahoor Ul Huq, and A. P. Siva Kumar (2019, pp. 24-28) examined the Hadoop Framework to identify potential security flaws. They also presented a method to mitigate these vulnerabilities. The results encompass the weaknesses in the logging process, vulnerabilities in the file system, and the inclusion of external jar files into the framework. Resolved these vulnerabilities by employing customized Map Reduce jobs. It is crucial to familiarize oneself with the API as it plays a vital role in developing customized applications for Map Reduce and HDFS. The Hadoop framework is also vulnerable to insider attacks, a concern that is often overlooked and requires significant attention. The Cloudera Distribution for Hadoop (CDH) is a popular Hadoop distribution that incorporates TLS encryption and Kerberos authentication for security purposes. The security measures implemented by CDH have certain limitations and require more definitive solutions to effectively tackle security and privacy concerns within the Hadoop framework.

## **2.9 Security aspects for data**

**2.9.1** Sharma and Sinha (2021, pp. 1372-1382) analyze the topic of data security in cloud computing. This is an examination of data stored in the cloud and the various security-related factors associated with it. The article will provide an in-depth analysis of data protection strategies and procedures employed worldwide to achieve optimal data security by mitigating risks and threats. Additionally, it included an analysis of the potential dangers and vulnerabilities that data stored in the cloud may face, along with a summary of three specific types of security issues. The primary focus of this study was on the issue of data security in cloud computing, including the associated hazards and consequences. The discussion has focused on data in different states and the efficient approaches for encrypting data in cloud computing. The study included an overview of block ciphers, stream ciphers, and hash functions, which are utilized to encrypt data in the cloud, whether it is at rest, in transit, or in use.

**2.9.2** In their study, Kumar, Athavale, and Kartikey (2021, pp. 18-29) provide a detailed overview of cloud computing security. They also examine and evaluate the security concerns and threats that are most susceptible in cloud computing. The purpose of their

research is to ensure that both end users and service providers are well-informed about the significant security concerns and threats associated with cloud computing. It also addresses the security goals and aspects of cloud security. It also emphasizes the primary concerns of customers regarding data integrity. In addition, we will examine instances of the most severe hacks, cyber-attacks, and data breaches.

**2.9.3** Koteswari and Kopparthi (2022, pp. 1928-1932) provide a detailed analysis of the security issues and concerns associated with cloud computing. Additionally, it addresses privacy concerns related to cloud computing, the absence of user autonomy, and the obstacles faced by cloud computing. The study evaluated the primary privacy and security concerns associated with migrating to cloud environments and outlined some strategies to mitigate these challenges. Several of these topics are developed and investigated. Cloud service models and their ability to process and analyze large amounts of data enable the development of innovative tactics that use the increased value of personal information. Simultaneously, the widespread utilization of personal records in the extended company can be highly controversial. Therefore, it is imperative to provide procedures that enable individuals to maintain control over their personal information. Currently, due to the abundance of information that is known, documented, and easily accessible, it has become challenging for individuals to avoid being evaluated based on their past activities. Cloud computing is a novel development that has the potential to greatly transform our internet usage. However, there are numerous factors that require careful consideration and caution. Numerous novel applied sciences are rapidly emerging, each accompanied by technology advancements and the potential to enhance the lives of humans.

**2.9.4.** The authors, Muhammad Bilal Qureshi, Muhammad Shuaib Qureshi, Saqib Tahir, Aamir Anwar, Saddam Hussain, Mueen Uddin, and Chin-Ling Chen (2022), conducted a thorough analysis and visually depicted the workflows of several data security and encryption systems. These techniques were evaluated based on shared parameters. This survey is to compile a comprehensive collection of data encryption approaches that have been extensively discussed in the literature. The focus is on securing data in smart devices that are offloaded to cloud computing services. After examining different encryption algorithms for the data transmitted by appliances in the smart architecture, it is clear that a careful trade-off between complexity and security must be achieved when choosing an algorithm based on the organization's business

requirements. An intricate technique typically requires a longer duration for encryption or decryption, whereas a less complex algorithm may not be suitable for extremely sensitive material. The primary obstacle preventing smart systems management from transferring their data to the cloud is data security. Enhancing the security of encryption and decryption keys is vital to prevent unauthorized access to authentication information by malicious entities.

## **2.10 Secure Cloud Storage**

**2.10.1** In their study, Bhuyan, Sontakke, Bisht, Singh, and Adtani (2021, pp. 1490-1496) conducted a survey on the concepts of distributed computing and the security challenges that arise in the context of cloud infrastructure. This article comprehensively examines the primary research and challenges associated with cloud computing. It also provides best practices for both professionals and organizations seeking to enhance their current environment and maximize the utilization of cloud services. The primary focus of our study is to analyze existing literature and appreciate the concept of multi-occupancy security concern.

**2.10.2** Chittumothu Srividhya and Deepthi B (2022, pp.875-881) examine the advancement of cloud computing technology in response to the rapid expansion of unstructured data. As a result, cloud storage technology has garnered increased attention and undergone significant improvement. The cloud provider does not offer recommendations regarding the management and storage of global cloud data. Privacy protection systems typically rely on encryption technology. There are numerous techniques available to protect data stored in the cloud and maintain privacy. Our proposition entails a fog computing-based storage structure consisting of three layers. The proposed system may effectively leverage cloud storage while ensuring data privacy. The HashSolomon coding algorithm is utilized to partition data into distinct segments. If a single data component is missing, we will lose the data information. Our framework employs algorithms based on the bucket concept to ensure the security and efficiency of data information. Furthermore, utilizing computational intelligence, this technique has the capability to calculate the distribution fraction that is saved in the cloud, fog, and local machine, respectively. Software as a Service (SaaS) refers to a software distribution model where applications are hosted by a service provider and

made available to customers over the internet. The client deploys their application on a hosting environment that can be accessed by application users from different clients via a network. The client has no authority or oversight over the underlying cloud infrastructure, except for a few particular application configuration options that are exclusive to the user. Google Apps and Microsoft Office 365 are both examples of Software as a Service (SaaS).

**2.10.3** In their study, Kumar and Prakash (2020, pp. 334-340) developed a system with the objective of securely storing and retrieving data on the cloud, ensuring that only the data owner has power over it. Data security concerns in cloud storage are addressed by the implementation of cryptography and steganography techniques. Data security is ensured through the utilization of the RC6, 3DES, and AES algorithms. The LSB technique (Steganography) is employed to securely store important information. The multithreading technology reduces the time required for the encryption and decryption process. Through the implementation of the suggested security mechanism, we have successfully achieved enhanced data integrity, heightened security measures, reduced latency, robust authentication, and strict confidentiality.

## **2.11 Understanding Blockchain technology**

**2.11.1** In Mihus's (2022, pp. 42-50) study, the author analyzes the key developmental phases of blockchain technology in different organizations' operations from 1991 to the present. The correlation between the evolutionary stages and Tiers of Blockchain has been established as follows: 2008-2013 (Blockchain 1.0); 2013-2015 (Blockchain 2.0); 2015-2018 (Blockchain 3.0); from 2018 to the present (Generation X). The various categories of blockchain, including public blockchains, private blockchains, semi-private blockchains, sidechains, permissioned blockchains, distributed ledgers, shared ledgers, totally private or proprietary blockchains, tokenized blockchains, and tokenless blockchains, have been organized systematically. The study focuses on the specificities of implementing blockchain technologies in the operations of firms across several sectors of the economy. A SWOT analysis was performed, uncovering that blockchain technologies will surely progress, impacting several industries such as public administration, retail, information technology, travel, health, education, agriculture, and entertainment. One potential approach to enhancing the utilization of blockchain

technologies involves: enhancing the confidentiality of transactions; expanding the scalability of blockchain networks; establishing interoperability between diverse blockchain systems; fortifying the security of blockchain operations; and adopting a personalized approach to the application of blockchain technology.

**2.11.2.** Palacios, Gordón, and Aranda (2020) analyze blockchain evaluation projects from a perspective that considers the development and progression of technology. In addition, a mapping exercise was carried out to determine the elements that influence blockchain efforts, consolidate existing evidence, and identify any gaps in the relevant techniques. It provides a concise overview of the several generations of blockchain and highlights the distinctions between them. Authors emphasize the need to include several governance approaches in the factor, including both "on-chain" and "off-chain" governance. They suggest evaluating the relevance of these approaches to the current situation and considering how the platform evolves. This study identifies and examines many limitations related to the common challenges to validity in mapping investigations.

**2.11.3** In Taherdoost's (2023) study, an analysis is conducted on the present condition of smart contracts in the blockchain. The study explores their various uses and examines the potentially transformative impacts resulting from their distinct attributes. The presence of constraints and their subsequent impact on other areas has been acknowledged, along with the challenges and omissions found in the existing literature on this matter. The deficiencies and obstacles in the pertinent literature have also been examined, with specific emphasis on the constraints. Smart contract technology shows promise, but it is still in its early stages of development. There are several issues that need to be addressed, including the limited data processing capacity and inefficient management of smart contracts by mainstream blockchain systems, the absence of a sophisticated programming language for contract development, and the lack of effective techniques for scanning and addressing vulnerabilities. Additionally, smart contracts have low maintainability and are susceptible to various security vulnerabilities. Exploration of forthcoming advancements in fields like data science, artificial intelligence, and game theory, which can be seamlessly incorporated into smart contracts in blockchain systems, has been contemplated and showcased.

**2.11.4.** In their comprehensive study, Dilbag Singh, Suhasini Monga, Sudeep Tanwar, Wei-Chiang Hong, Ravi Sharma, and Yi-Lin He (2023) thoroughly examine the technology of blockchain and its use in the healthcare sector from several angles. A comprehensive mapping study was undertaken to explore several scientific databases in order to identify the prevailing issues in healthcare management systems and examine the present applications of blockchain technology in healthcare. While blockchain possesses intrinsic features such as distributed ledger, encryption, consensus, and immutability, the use of blockchain in healthcare faces hurdles. This paper also offers insights into the research problems in blockchain technology and presents a solution taxonomy through a comparative analysis.

## **2.12 Working of P-2-P Technique**

**2.12.1** Sarvesh Wadi, Mrunal Shidore, Malhar Surangalikar, Devarshi Talewar, and Vedant Savalajkar (2022, pp. 78-82) want to create a ride-sharing application that is easy to use, minimizes errors, and utilizes blockchain technology. The primary objective was to explore Blockchain, a groundbreaking technology. The proposal suggests an architecture for a ridesharing service that is built on blockchain technology, allowing for decentralization and direct interaction between peers. A decentralized application is created to facilitate and enhance this ride-sharing system. This DApp utilizes Ethereum, an open and decentralized blockchain, along with cryptographic protocols, to automate network transactions and facilitate data exchange. In summary, blockchain technology may be utilized to construct a system that securely records digitally encoded smart contracts in databases that are both visible and decentralized. All of these databases are designed to store data that may be modified. The P2P Android Application offers ride sharing services in a manner similar to existing company services, with a superficially similar interface and user experience. However, the underlying system is decentralized.

**2.12.2** Ying Xing, Hui Shu, Fei Kang, and Hao Zhao (2022) present a botnet detection framework that utilizes random walks. They also develop a rule using big data platforms to differentiate between P2P and non-P2P traffic. In addition, the Jaccard coefficient is employed to compute the P2P cluster confidence for the creation of a shared neighbor graph, whereas SAW is utilized to differentiate between benign and

malignant communities. A dataset evaluation experiment was conducted, and the experimental results were compared to those of the existing approach on an unstructured dataset. The findings demonstrate that the identification of P2P bots can be achieved with a notable level of precision even when normal P2P network activity is present.

**2.12.3** Wang, Gu and Ma (2020), in this paper the BitTorrent file-sharing service of P2P is selected as the platform for concealing information, and a multimode steganographic technique utilizing Bitfield messages is introduced. By utilizing BitTorrent's cooperative transmission and the non-content-authentication mechanism of the Bitfield message, our scheme delivers secret information during the exchange process of BitMapInfo between two peers. The steganographic mode is dynamically selected based on the size of the secret, allowing for adaptive bandwidth. Experimental results demonstrate that our scheme effectively resists statistical-based detection and outperforms existing methods by achieving a lower degree of detection rate in machine learning-based steganalysis.

## **2.13 Working of emulator like VMware Workstation**

**2.13.1** In Ahmed's (2020, pp. 346-355.) paper, an extensive examination of the most promising methods and technologies of virtualization is presented. The well-known virtualization mechanisms, including Full Virtualization, Para Virtualization, Desktop Virtualization, and Network Virtualization, have been examined. In addition, a brief explanation was provided for various commonly used virtualization tools such as VMware, OpenVZ, Xen, Qemu, Docker, and Kernel-based Virtual Machine (KVM). This study conducted a comprehensive evaluation of recent research conducted by other scholars, which utilized a variety of visualization processes and tools. Consequently, a comprehensive analysis has been conducted on different virtualization solutions, focusing on key characteristics such as virtualization approach, usability, and accessibility. This study will be valuable for aspiring researchers who are interested in working in this field.

**2.13.2** In their study, Kapil, Mishra, and Gupta (2022, pp. 48-56) evaluate the performance of several hypervisors by analyzing their migration capabilities and comparing these qualities. The hypervisors include Xen, VMWare, and KVM, each

with their individual migration features: XenMotion, vMotion, and KVM migration. The study reveals that several factors significantly impact the performance of live virtual machine migration. These factors include extended downtime, the iterative transmission of a substantial amount of data, and an increased dirtying rate, all of which contribute to an extended total migration time. This study demonstrates that VMWare exhibits the lowest amount of system unavailability. This study examines the diverse migration services offered by numerous hypervisors. This report presents a comprehensive examination of the performance of these migration services. Multiple performance parameters can be influenced by several factors, such as the speed of the connection and the number of applications executed on virtual machines, as well as the migration of these machines over a wide area network (WAN). This paper explores a range of research obstacles that can be valuable in guiding future research endeavors. Furthermore, describe the notion of live virtual machine migration.

**2.13.3** Muhammad Garzali Qabasiyu and Umar Danjuma Maiwada (2022, pp. 59-66) conducted a research study. Cloud computing is an innovative concept that allows for the efficient and reliable processing of massive amounts of data on a scalable platform. Cloud computing provides access to extensive self-managed server pools, resulting in reduced expense and elimination of administrative challenges. Cloud computing services can dynamically scale up or down in response to changes in demand. Cloud computing is particularly advantageous for small and medium firms (SMEs) as it provides them with cost-effective IT tools that enhance productivity without requiring significant investments in in-house resources and technical equipment. Due to the expanding understanding of cloud computing, there has been a rise in its use by organizations. A virtual computer was established as a component of an Infrastructure as a Service (IaaS) platform. The architecture employed Vsphere ESXi, with Vsphere vclient being employed for the administration of virtual machines in ESXi. The research elucidates the significance of utilizing this approach and the manner in which it enhances understanding. Furthermore, it is a cutting-edge SSnew architecture that will facilitate the expansion of the Internet into the future's dominant computer platform. Considering the possibility of significant cost reductions and enhanced productivity in a challenging economic environment.

**2.13.4.** In their study, Kapil, Mishra, and Gupta (2022, pp. 48-56) examined the performance of several hypervisors by analyzing their migration aspects and comparing

them. The hypervisors include Xen, VMWare, and KVM, each with their individual migration features: XenMotion, vMotion, and KVM migration. The study reveals that several factors significantly impact the performance of live virtual machine migration. These factors include extended downtime, the iterative transmission of a substantial amount of data, and an increased dirtying rate, which results in longer migration times. Upon comparison, it is evident that VMWare experiences the lowest amount of downtime. Furthermore, it thoroughly examines and addresses the several research obstacles in order to encourage academics to pursue this line of inquiry.

## **2.14 Secure Cloud Framework**

**2.14.1** Reddy and Sein (2021, pp. 681-685) propose a robust testing method that ensures data integrity, confidentiality, and privacy protection. The suggested method exclusively utilizes the cloud server for the purpose of storing encrypted data blocks. It does not require any more computer verification. The responsibility for all the functions of the scheme is with TPA and the data owner. Various resources are employed to assess the planned audit procedure. The proposed method effectively fulfills all the required operations while decreasing the workload on cloud servers. In the future, data dynamics operations such as data evaluation, deletion, and installation will be executed.

**2.14.2** Abdullah and Kabir (2021, pp. 31-42) conducted a literature review on the security challenges in Cloud computing. They proposed a security model and a multilayer framework to create a secure Cloud computing environment. This framework identifies the security requirements, attacks, threats, and concerns that are associated with the deployment of clouds. The approach we present is modular, meaning that we address each threat individually and find a solution for each one. This facilitates more efficient management of the cloud system and empowers the administrator to implement targeted solutions to mitigate the threat. It also presents the primary security concerns of cloud computing systems, examines the potential risks, and suggests methods to address them. We employ a quantitative security risk assessment model to provide a multilayer security framework for addressing the security vulnerabilities of cloud computing systems. Furthermore, asymmetric encryption using RSA is employed to mitigate the potential vulnerabilities associated

with authentication and authorization. This system effectively mitigates targeted attacks.

**2.14.3** Awad (2020, pp. 264-280) proposed a cloud-based framework to address security concerns in contemporary data storage systems. Existing storage systems are vulnerable to a multitude of cyber security vulnerabilities due to their traditional architecture and the procedures used to manage data on them. The migration of local storage services into a cloud environment is implemented to address several interconnected cyber security concerns. However, this shift has led to the emergence of several cyber security vulnerabilities, including the exposure of public networks and the sharing of infrastructure among multiple tenants. The vulnerabilities resulting from this conversion expose the security breaches and pose a threat to the underlying confidential information. An in-depth analysis is conducted on the significance of migrating storage services to the cloud environment and the emerging cybersecurity challenges associated with this transformation. The intended framework is designed to bolster the current information systems security measures. Additionally, it provides cyber security professionals with the opportunity to create and tailor cloud services to better meet the requirements of enterprises. Additionally, provide a concise overview of the cybersecurity weaknesses present in cloud-based frameworks.

## **2.15 Distributed File Storage**

**2.15.1** Macko and Hennessy (2022) provide an overview of significant design choices and the often employed alternatives within those choices. Additionally, it provides a qualitative investigation of the compromises they make. We use a range of relatively new designs and their different versions to demonstrate various tradeoff decisions in the design field, even though they have not been thoroughly researched. By doing this, we offer an introduction to distributed file systems and identify both extensively studied and neglected regions, with the aim of stimulating further research.

**2.15.2** Chang and Cui (2021) offer a distributed storage technique that includes temporal and geographical semantic constraints. Considering the extensive features of multisource economic big data in terms of time, space, and semantics, this analysis examines the existing mainstream methods for dividing data. The advantages and disadvantages of each method are compared, and a new approach is suggested: a

multilevel data division that combines Geohash and Hilbert. Using the time stamp and semantic coding table, we establish a unified coding method that incorporates temporal and spatial semantic constraints. This method enables efficient distributed storage of large-scale economic data from multiple sources and facilitates cross-modal management, analysis, and application of economic big data. In this study, we developed a series of automated deployment frameworks for NoSQL data clusters on cloud platforms. The main objective is to utilize distributed storage workloads to pre-deploy the resource configuration of the cluster. This allows the data cluster to optimize the utilization of storage resources and minimize customer expenses, all while adhering to SLA limits. The objective can be effectively accomplished through the utilization of distributed storage workloads. Nevertheless, distributed storage systems invariably encounter faults, and these errors can potentially become significant, presenting a formidable challenge to our study. In the current economic big data landscape, resource allocation is more stringent due to the presence of numerous sensitive aspects. Given the varying workload, the conditions of distributed storage will likewise alter. Consequently, it is imperative to adapt our distributed storage approach in accordance with the specific attributes of the workload. The multilevel data division approach, which combines Geohash and Hilbert, exclusively takes into account the temporal, spatial, and semantic connections among the data. This work focuses solely on basic semantics when it comes to data partitioning, due to the intricate and extensive nature of data semantics. Additional investigation can be conducted to enhance the abundance and accuracy of semantic constraints in data partitioning. This will provide comprehensive examination and extraction of vast economic big data.

**2.15.3** Venkatakrishna and Reddy (2020, pp. 1055-1061) have developed a method that allows users to efficiently transfer large amounts of data across the network by compressing the files. The system should also authenticate the user and provide communication according to the specified level of security, while transferring the requested file and executing the required server-side process if necessary. Currently, data will be transmitted through the system. The customer who obtained the record will do duties such as deimplanting, decoding, and decompressing in a systematic and gradual manner. Additionally, it provides insight into constructing a framework that encompasses all the essential requirements of the client.

**2.15.4** Aditi Sharma and Parmeet Kaur's (2023, pp. 184–192) survey analyzes the deployment of multitenancy in distributed, cloud-based storage services. The text examines the current strategies and frameworks utilized by commercial cloud service providers. The presentation included an examination of different types and architectural patterns of multitenant data storage, as well as an analysis of the current solutions offered by cloud storage providers for software as a service applications. A comprehensive exposition of the research conducted in the field has been presented, including a comparative study. The utilization of blockchain for reliable storage has been deliberated. According to the research, some unresolved issues have been identified for future investigation.

## **2.16 Centralized file storage**

**2.16.1** Kaur, Saxena, and Mahajan (2021, pp. 401-406) presented a comprehensive study of decentralized cloud storage technologies, namely IPFS, FileCoin, Sia, Swarm, and Storj. The storage of vast amounts of data on these platforms is a key advantage of the internet. From the era of Web 1.0, which had a centralized system, to Web 2.0, which adopted a decentralized approach, we have now entered the era of Web 3.0. Web 3.0 is characterized by a distributed decentralized web that utilizes storage platforms such as IPFS, Sia, Swarm, and Storj. Interplanetary File System (IPFS) and Filecoin have replaced the old HTTP protocol for file storage. They focused on addressing the essential needs of the user through online backup services. These cloud systems cater to the user's requirements for decentralized peer-to-peer storage, but they do not focus on the actual implementation. This study specifically examines the applications, benefits, drawbacks, and challenges related to decentralized file systems.

**2.16.2** Doan, Psaras, Ott, and Bajpai (2022, pp. 7-15) offered a comprehensive explanation of IPFS and its fundamental characteristics. They demonstrated how IPFS utilizes a combination of various networking protocols and peer-to-peer ideas to establish a basis for decentralized cloud storage. The fundamental components of this system facilitate the distribution and transfer of files through peer-to-peer networks, allowing for a departure from centralized cloud storage. These components include the ability to maintain consistent names for files, eliminate duplicate files, verify file integrity using CIDs, resist censorship, tolerate network partitions, and ultimately

achieve decentralization. These are just a few of the notable features of this system. However, IPFS still faces hurdles in addressing issues such as access control and built-in privacy, participation incentives, and ensuring persistent availability and replication of content. Researching the performance of decentralized solutions is currently a relevant and urgent topic that we strongly encourage the community to pursue.

**2.16.3** Liu et al. (2021, pp. 4-10) conduct a comparison between two design options for Hydra: one involves developing it over TCP/IP with a centralized controller, while the other involves constructing it over Named Data Network (NDN) to enable distributed control. This study demonstrates that constructing Hydra using TCP/IP and a central controller results in a straightforward and uncomplicated design. However, to ensure scalability and reliability, the controller must be replicated, and cloud CDN is required to expand data delivery, which introduces additional complexity to the overall design. On the other hand, implementing Hydra over NDN automatically provides scalable and efficient data dissemination and enables distributed control with strong resilience.

**2.16.4** Kritikaa, Himani, Abhay Kumar Singh, Ashish Upadhyay, Aman Agrawale, and Himanshu Gupta (2022) proposed a secure centralized cloud storage solution that utilizes hybrid cryptography and access control using blockchain. The purpose of this paper is to compare the centralized storage system with the decentralized storage system. The centralized cloud storage systems employ hybrid algorithms such as AES, Blowfish, and SHA-256. Furthermore, Scheme offers a centralized storage solution with the ability for users to set their own key. The purpose of the model is to enhance the security of file encryption on the cloud. This study additionally demonstrates the analysis of execution time for the suggested centralized storage model. The findings of the examination of the suggested hybrid model were also compared with the results of the analysis of each individual algorithm. The outcome of execution analysis can change as a result of the fluctuation in Internet speed. In a decentralized storage system, the data requester uploads a file and submits it. The file is then published on IPFS and a unique hash is returned. Additionally, the file provided by the data requester must be confirmed by a data verifier. The two implementations demonstrate the superiority of decentralized storage, as it is exceedingly difficult to manipulate data on a decentralized server. Additionally, it is cost-effective when there is a need for significant storage capacity. Ongoing research is focused on enhancing data security in response to the

growing volume of data resulting from increased internet usage. In the future, consider implementing a dual access control system utilizing Blockchain and IPFS.

## **2.17 Study of digital currencies like Bitcoins and Ethereum**

**2.17.1** Huang, Wang, and Wang (2020) examined the architecture of Ethereum and the operational principles of smart contracts. They made enhancements to address security concerns associated with smart contracts. Thus, the concept and philosophy of Ethereum are introduced in the context of blockchain technology, utilizing the data format exclusive to Ethereum. Next, the technical level of smart contract is explained by providing pseudo code, which serves as the fundamental component of Ethereum. In practical smart contract trials, the auction is executed using the Solidity programming language. The issues with the auction application have now been resolved and enhanced to enable the utilization of smart contracts on the Ethereum network.

**2.17.2** Sidra Anwar, Sadia Anayat, Sheeza Butt, Saher Butt, and Muhammad Saad (2020, pp.30-39) demonstrated that Ethereum's blockchain represents a more sophisticated iteration of Bitcoin's blockchain. This study has provided a comprehensive understanding of the superior cryptocurrency and the impact of blockchain technology on established enterprises. Additionally, a comparison was made between ethereum and bitcoin based on their shared attributes, leading to the conclusion that ethereum is more expedient than bitcoin. Ethereum, founded 5 years after Bitcoin in 2009, is comparatively younger. This research aimed to elucidate the distinctions between bitcoin and ethereum in order to determine their relative superiority and provide reasoning for such a conclusion. The utilization of blockchain technology in the government sector and its applications has been demonstrated to have the potential to bring about a significant and transformative impact on the world.

**2.17.3** Umair Khan, Zhang Yong An, and Azhar Imran (2020) provide a thorough examination of the security and privacy aspects of Blockchain. Blockchain technology offers transparent, tamper-proof, and secure platforms that have the potential to facilitate innovative solutions, particularly when integrated with smart contracts. The proposal suggests utilizing Blockchain Ethereum Technology to implement a system for content protection and transactions. The suggested system incorporates an encryption method to ensure transparent transactions and avoid smart forgery and

hacking by implementing it directly on the material. The experimental results demonstrate that the suggested strategy has significant potential to improve transparency in transactions by reducing security risks in digital content transfers.

**2.17.4** Dániel Kondor, Nikola Bulatovic, József Stéger, István Csabai, and Gábor Vattay (2023) examine the transactions of Bitcoin and Ethereum, two of the largest publicly available real-world complex networks. The researchers provide a comprehensive analysis of the networks' time evolution, offering a full understanding of their dynamics. They garnered significant interest from the network science community, as well as assessments from economic and cryptography viewpoints. An research of the early stages of the Bitcoin network has revealed the prominent existence of preferential attachment, sometimes known as the "rich-get-richer" phenomena. Furthermore, please reconsider this subject by utilizing a current iteration of the Bitcoin network that has expanded nearly 100 times in accordance with the initial research. Moreover, it also conducts a comparison with Ethereum, the second most significant cryptocurrency. The results indicate that preferential attachment remains a crucial feature in the development of both the Bitcoin and Ethereum transaction networks. In order to simplify additional analysis, we have released a recent version of both transaction networks, as well as an efficient software implementation. This software is capable of evaluating linking statistics, which are important for studying preferential attachment on networks that have several hundred million edges.

## **2.18 Hadoop Distributed File System**

**2.18.1** Wahid Rajeh (2022, pp. 23-42) examines the strengths and weaknesses related to illegal access. The first section of the article provides a description of Hadoop, including an overview of its fundamental issues and the security challenges specifically related to the Hadoop Distributed File System (HDFS). The final segment of the initial section delves into topics such as authentication and authorization, which are direct outcomes of the parallelism and distributed data processing capabilities of Hadoop technology. Section two of the article focuses on illegal access, categorizing users, various levels of security, and malevolent behaviors. The final portion of section two delineates the security solution, elucidates the distinction between direct and indirect access users, and provides specific advice for rectifying unauthorized access

difficulties. The paper concludes by introducing honeypot nodes, which serve as security devices for gathering important data on dangerous entities. The presentation introduces the concept of creating a novel Intrusion Detector that utilizes an artificial neural network. The detector will serve as an essential component of a novel virtual honeypot method and serves as the foundation for future research endeavors.

**2.18.2** Gousiya Begum, S. Zahoor Ul Huq, and A. P. Siva Kumar (2020, pp. 1-10) Any authorized user has the ability to execute a map reduce job on the Hadoop cluster to process any file. However, there is no system in place to inspect the jar files for the potential presence of harmful code that could damage the files stored within HDFS. Map reduce jobs can expose the file system to security attacks, such as data tampering, data theft, and metadata gathering. This paper presents a system in which sandbox security is used to test every map reduce jar. The purpose is to verify that the jar is not malicious, and to prevent suspicious jar files from processing the data in the HDFS. This functionality is currently absent in the current Apache Hadoop framework, and this work has been made accessible on GitHub for evaluation and potential inclusion in future versions of Apache Hadoop.

**2.18.3** Pooja Pandit (2021) investigates the Hadoop Ecosystem. Initially provides a detailed overview of the Hadoop architecture. Examine the different elements of the Hadoop Ecosystem, including the Hadoop Distributed File System, Map Reduce, YARN, and others. Additionally, examine the characteristics of the HDFS and YARN MapReduce framework.

## **2.19 Gap Analysis**

- After extensively reviewing numerous study articles and publications pertaining to the subject, it has been identified that there are deficiencies in the existing research.
- When information is communicated over the internet, it can be sent individually or as part of an archive file, and it may be compressed using either ZIP or GZIP. A zip file is a method of compressing files. The degree of file compression is contingent upon the specific content of the file. Plain text files exhibit significant compression, but formatted text files demonstrate less compression, and image files experience less compression.

- Let's consider an example where we have a text file with a size of 0 bytes and we compress it into a zip file. The zip file can have a size of approximately 112 bytes.



Figure 2.1 Original File and its Zip file

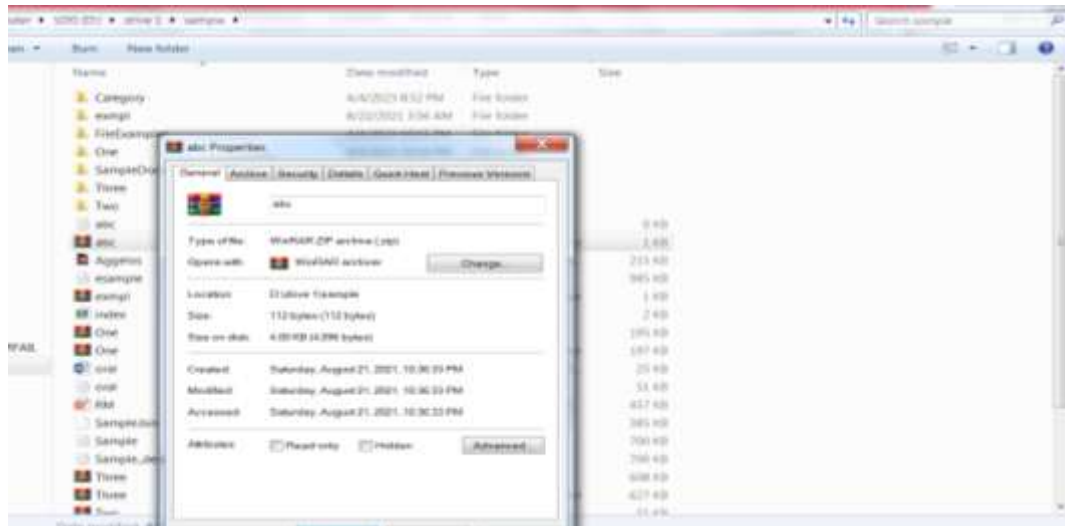


Figure 2.2 Size of Zip file

- Let's examine another example involving a folder that includes three files. There is one image file in .bmp format, one text file in .txt format, and one document file in .docx format. Each of these files has a size of 0 bytes. When these files are compressed into a zip file, the resulting size is 356 bytes. The zip file contains both an exact copy of the original folder and a compressed version of it.

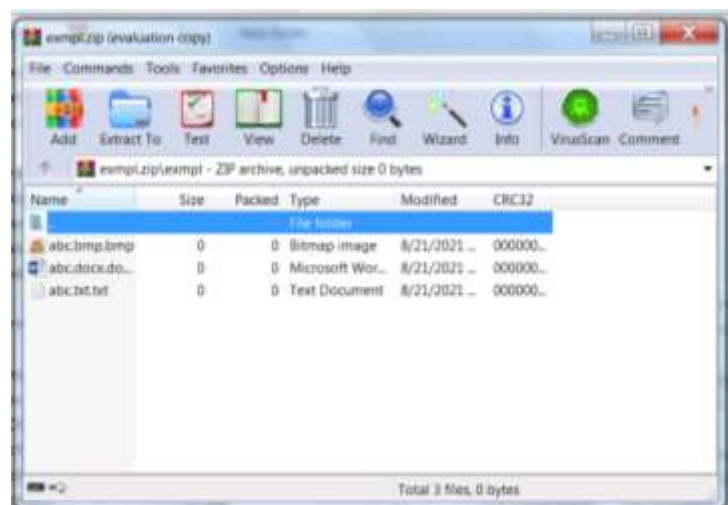


Figure 2.3 Original files

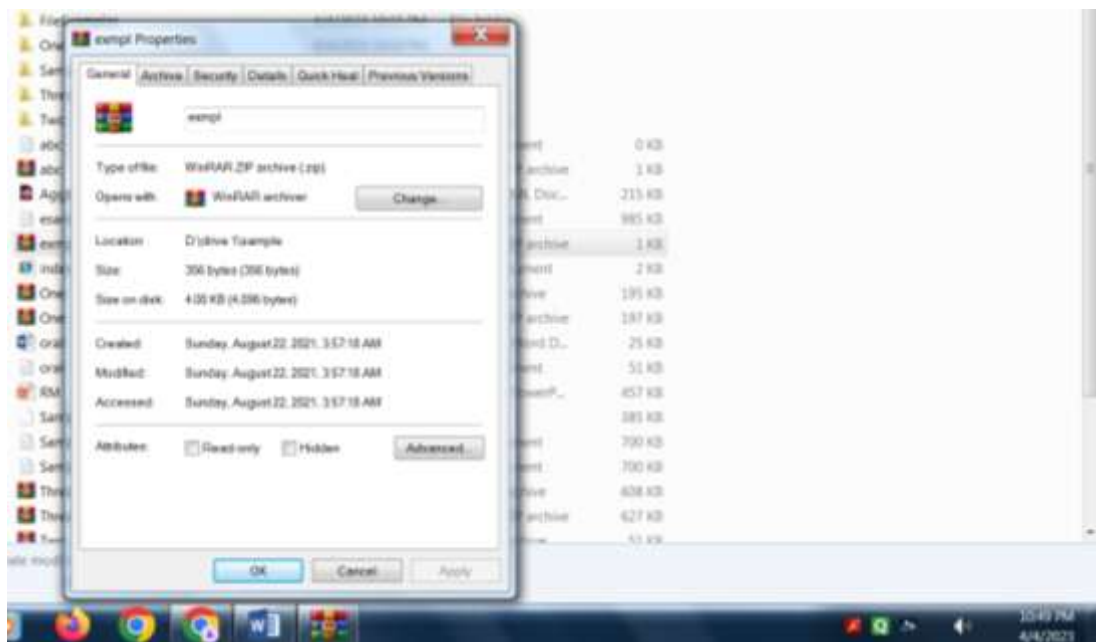


Figure 2.4 Size of zip file

- 
- Based on the aforementioned points, it can be inferred that zip files provide a higher compression ratio for larger files compared to smaller files.
- When encrypted files are added to a zipped folder, they will be automatically decrypted upon unzipping, potentially leading to inadvertent exposure of personal or sensitive data. Therefore, it is advisable to refrain from compressing encrypted files.
- The predominant compression technique utilized in zip is deflate, which is essentially the implementation of the Huffman algorithm as the underlying approach. One of the data compression strategies
- Data compression techniques have the ability to reduce the size of a text file to 50% or even a substantially greater proportion of its original size.
- Additionally, it is worth noting that if you apply a data compression algorithm such as Huffman coding to a .txt file that has a size of 0 bytes, the resulting compressed file will likewise have a size of 0 bytes.
- Thus, regardless of the size of the file, the data compression technique is effective.
- Certain publications focus solely on the security of user data, but they fall short when it comes to addressing the privacy of user data.

- Several papers discuss the Confidentiality of user data, yet they neglect to verify the integrity of the data.
- The field of data protection and privacy raises significant concerns regarding the storage of sensitive data in cloud environments. The perception and trust of users are crucial in guaranteeing the adoption of cloud-based services.
- Several articles discuss how manufacturers and cloud providers are utilizing cloud-based storage methods to supply their security solutions, which are commonly referred to as Security as a Service (SecaaS).
- Some publications solely discuss the implementation of compression algorithms, encryption approaches, and the operation of cloud storage. Nevertheless, there is limited discourse among researchers regarding the shared characteristics of compression and cloud security. Their implementation of cryptographic compression fails to provide a security solution for cloud storage. Flooding attacks, namely denial of service assaults, have been identified as being associated with cloud storage. The significant limitations and research deficiencies discovered indicate that, even after several decades, many users still lack confidence in cloud computing. Many people are uncertain about whether or not to store sensitive data on the cloud. The user's lack of confidence in authentication, authorization, and access control has been identified.
- Several papers discuss the various challenges that must be addressed in relation to security and privacy in cloud storage. These challenges include ensuring authentication, authorization, and access control mechanisms, preventing data loss or leakage, securing cloud credentials, ensuring control and management of cryptographic keys, and securing data during transit.
- Several papers focus only on cloud security, analyzing specific security needs and proposing effective solutions to mitigate potential dangers through the utilization of a trusted Third Party. However, only a few number of academics have suggested a successful approach to establish safe and reliable cloud storage. This approach involves employing a distributed storage integrity auditing system that combines holomorphic tokens with distributed erasure-coded data to store data in a dynamic manner. They do not successfully implement cryptographic compression to provide a security solution for cloud storage.
- Several publications analyze the functions of security and trust in cloud computing environments from the perspective of organizations that save their confidential data

with cloud computing providers. The user proposes that security solutions for cloud computing should prioritize problem abstraction by employing model-based approaches to capture various security perspectives and integrate them into a comprehensive cloud security model.

- Several publications discuss challenges and issues related to cloud computing, with security being the primary focus. Primarily, cryptography approaches are used for ensuring the security of data, however, a significant number of these techniques are now considered obsolete. The suggested framework aims to establish a system that utilizes both cryptography and hash algorithms.
- Therefore, a holistic solution is necessary that considers all the aforementioned factors in order to establish a robust storage system. Thus, a comprehensive and secure framework for cloud storage is proposed, which addresses all these factors and ensures perfect security.

# CHAPTER 3

## RESEARCH METHODOLOGY

This section will provide a comprehensive overview of several research approaches. Subsequently, the justification for choosing a certain approach for this thesis will be presented.

### **3.1 Introduction: Research Approach**

This chapter is dedicated to a particular research methodology and presents a unique approach that can be used in research. A research proposal adheres to a comprehensive methodology for its overarching objective, and it may encompass experiments, surveys, models, and case studies. Various data collection techniques can be used in a certain research methodology, and both quantitative and qualitative methods can be utilized to analyse the entire inquiry.

The chosen research methodology for this study is a Pragmatic approach to research, specifically utilizing Mixed Methods, in order to effectively achieve the research objectives. Pragmatic research encompasses the utilization of several methods, strategies, and procedures commonly associated with both quantitative and qualitative research. Furthermore, it is widely recognized that each method has inherent limitations and that several approaches can be mutually reinforcing. The selected analytical methodology was utilized to gather research data and offer a resolution to the problem statement. Throughout the thesis, we shall strictly adhere to a research philosophy grounded in pragmatism and place a strong emphasis on practical realities. When analysing the utilization of cloud services in a business, both subjective and objective perspectives will be considered.

### **3.2 Types of Research Methods**

The types of methodologies encompass a wide range of study strategies and procedures, varying from overarching assumptions to intricate methods for gathering, analyzing,

and interpreting data. (Source: Research, 2020) There are three research methodologies that will be discussed:

1. Qualitative
2. Quantitative, and
3. Pragmatic (Mixed methods)

1. Qualitative research is a method used to investigate and comprehend the social or human issues that individuals articulate. The objective is not to provide a superficial description of a huge population sample, but to get a deep comprehension of a particular organization or event. This form of research mostly centres on the examination of individuals' emotions, cognitive processes, and decision-making. This study mostly involves the exploration of concepts through extensive discussions and the examination of specific unresolved concerns. Representatives are requested to provide an explanation for their response. This can illustrate the fundamental motivations, connections, and stimuli of behaviour (Qualitative research, 2020).

Typical methods of data collecting include focus groups, in-depth interviews, continuous observation, newsrooms, and the ethnographic participation/observation employed in this study.

2. Quantitative research is the examination of the relationship between variables in order to test target theories. This type of research employs a rational methodology, offering a statistical assessment of the attitudes and beliefs of individuals. To illustrate, conduct quantitative research to determine the number of firms who utilize and endorse Cloud usage and its services.

This research mostly relies on methodologies such as surveys and interviews, which prompt respondents to select one or more options. Possible response options may consist of an acceptance scale (ranging from strongly opposed to strongly in favor), a Likert scale, and priority ranking, and so on.

These questionnaires can be completed via telephone, online, or in paper format. This research is feasible. The sole constraint is that the number of participants must be sufficiently substantial to yield conclusive outcomes.

| <b>Criteria</b>                   | <b>Qualitative</b>                                                                                       | <b>Quantitative</b>                                                                                         |
|-----------------------------------|----------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| <b>Data</b>                       | Data is in the form of words, Pictures or objects.                                                       | Data is in the form of numbers and statistics.                                                              |
| <b>Method used</b>                | Methods include focus groups, in-depth interviews, and reviews of documents for non-numeric information. | Surveys, structured interviews & observations, and reviews of records or documents for numeric Information. |
| <b>Process of Research</b>        | Inductive approach used to Formulate theory or hypotheses.                                               | Deductive approach used to test Pre-specified concepts, constructs, and hypotheses.                         |
| <b>Response options</b>           | Unstructured or semi-structured Response options.                                                        | Fixed response options.                                                                                     |
| <b>Statistical test</b>           | No statistical tests are performed.                                                                      | Statistical tests are performed for analysis.                                                               |
| <b>Time required</b>              | Time spend at the time of planning is lower than that spent during the analysis phase.                   | Time spend at the time of planning is higher than that spent during the analysis phase.                     |
| <b>Objectivity / subjectivity</b> | Highly subjective.                                                                                       | Highly objective.                                                                                           |
| <b>Result</b>                     | Results depend on skill and accuracy of the researcher.                                                  | Results depend on the measuring device / instrument.                                                        |
| <b>Final Report</b>               | Report contains textual details & verbatim from research participants.                                   | Report contains statistical analysis of data.                                                               |

Table 3.1: Qualitative versus Quantitative Research

### 3.3 Method of Data Collection

The data collection approach relied on characteristics such as the size of the sample, the level of confidence, and the margin of error percentage. The association between increased or decreased values is provided below:

| <b>Parameter</b> | <b>Value Increased</b> | <b>Value Decreased</b> |
|------------------|------------------------|------------------------|
| Margin of Error  | Accuracy decrease      | Accuracy increase      |
| Confidence Level | Accuracy increase      | Accuracy decrease      |
| Sample Size      | Accuracy increase      | Accuracy decrease      |
| Population Size  | Accuracy decrease      | Accuracy increase      |

Table 3.2: List of Parameters

**Calculating Sample Size:** The following formula (Stokes, 2017) is used to calculate The sample size required.

**Formula:**

**Necessary Sample Size = (Z-score)<sup>2</sup> \* StdDev\*(1-StdDev)/(margin of error)<sup>2</sup>**

| Desired Confidence Level | Z-Score |
|--------------------------|---------|
| 80%                      | 1.24    |
| 85%                      | 1.44    |
| 90%                      | 1.65    |
| 95%                      | 1.96    |
| 99%                      | 2.58    |

Table 3.3: Desired Confidence Level

**Above are the z-scores for most common confidence levels:**

Necessary Sample Size =  $1.96 * 2 * 0.5 * (1-0.5) / 0.05 * 0.05 = 384.16$

**Our sample size = 32 (interview respondents)  
+ 363 (survey respondents)  
= 395**

The list of procedures used for Data Collection are as follows:

1. **The semi-structured interview technique** was employed to gather data. The selected participants were from S4DS. I am affiliated with a group called S4DS. S4DS is an acronym that stands for Society for Data Science. S4DS is an organization comprised of Data Scientists. Its objective is to enhance the understanding of students, researchers, corporations, and professionals in order to help them build successful careers in the field of Data Science. It was established in 2017. Data science, sometimes referred to as data-driven science, is an interdisciplinary domain that focuses on the application of scientific methods, procedures, and systems to extract valuable knowledge or insights from data in diverse formats. Currently, it operates in 20 different nations. The members of S4DS encompass a range of specialists, such as information security auditors, information security professionals, educators, consultants, regulators, computer scientists, and internal auditors. These chapters offer education, facilitate resource sharing, networking, advocacy, and other related activities. S4DS hosts online Datathons, monthly challenges, digital meetings, webinars, workshops, and summer schools. I participated in webinars and workshops. During several conferences, I had

the opportunity to engage with security experts, auditors, chief technology officers (CTOs), and regulators. The sample size consisted of 32 participants.

There were few structured inquiries. Utilizing more open-ended questions allowed me to assess the current security situation and determine my ultimate study topic. The discussions provided me with a comprehensive grasp of the security dangers that these professionals encounter on a daily basis, their strategies for addressing these risks, the financial implications of utilizing cloud storage services, the issues that current cloud storage systems are capable of resolving, and, most significantly, their limitations. In addition, they were queried about their primary concerns regarding security and the most vulnerable assets. To access a sample question sequence, please refer to Annexure I.

- 1) **Surveys** were administered with both Google online forms and physical paper forms. A total of 363 participants completed these forms. Some individuals have the necessary qualifications to be considered security professionals, while others did not possess the same level of qualifications. Some individuals had direct responsibility for security within their enterprises, while others did not have such responsibility. A diverse range of organizations was chosen. The time period specified is from August 2019 to December 2020.
- 2) **The In-Depth interview** technique was employed either with individual participants or a targeted group of respondents. This methodology was employed due to its efficacy in delving into the respondent's nuanced viewpoint on a specific concept, initiative, or remedy. It is quite advantageous when we desire to thoroughly investigate novel matters and concepts. The laddering strategy of question asking was employed, whereby each question served as a stepping stone to the next. This methodology is especially advantageous during the initial phases of investigation as it offers a clear path for the researcher. These interviews and questionnaires were crucial in helping me finalize my topic and gather data. Each interview had a duration of approximately 15-20 minutes. The interview format would consist of either an individual interview or a group interview with a specific focus. A total of approximately 32 individuals were included in the sample, and the snowball sampling technique was employed for participant selection. This involved asking interviewees to

identify other potential participants, creating a chain of referrals until the desired sample size was achieved. Snowball sampling is advantageous in situations where the researcher cannot locate enough participants and when potential participants may be hesitant to reveal their names. Certain security personnel may be unwilling to acknowledge that they have indeed suffered a data loss and have made payment to retrieve their encrypted files and data. The company's reputation is in jeopardy.

### **3.4 Research Methodology Applied**

The research is exploratory until it identifies the essence of the problem. It is both descriptive, as it explains the reason for the problem, and experimental, since it provides a solution to the problem. Hence, the study encompasses all facets of inquiry. Therefore, it employs a combination of research methodologies. It is attempting to gather information about the current challenges encountered by the security industry. The data is gathered using comprehensive interview methodologies.

Additionally, it establishes the cryptographic architecture and conducts tests using standard security solutions to validate the hypothesis that typical cloud storage solutions are incapable of delivering. Next, analyze the characteristics of widely used cloud storage security measures and security frameworks to examine their construction, authentication, authorization, and data integrity procedures. It is necessary to research various strategies employed in creating a security framework in order to understand the problem thoroughly before proposing a solution.

Additionally, it examines the reasons behind the lack of robust security in conventional cloud storage solutions and the factors contributing to their shortcomings. The research is classified as qualitative due to its proposition of a comprehensive framework that can offer robust security solutions. The process involves constructing the framework and evaluating its performance using both familiar and unfamiliar samples. The process begins by constructing the framework, followed by testing its functionality. Furthermore, it operates the framework in a distributed environment and discovers that the framework is very efficient and cost-effective compared to the existing frameworks in the sector.

Prior to addressing the problem, it is necessary to examine some cloud storage systems. Table 3.4 provides a comparison of several cloud storage systems.

| Category                   | Dropbox                                                                        | Google Drive                                                                   | Microsoft OneDrive                                                                                   |
|----------------------------|--------------------------------------------------------------------------------|--------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|
| Compression Technique      | NO                                                                             | YES                                                                            | NO                                                                                                   |
| Multifactor Authentication | YES                                                                            | YES                                                                            | YES                                                                                                  |
| Server Data Encryption     | 256-bit AES                                                                    | 128-bit AES                                                                    | 256-bit AES                                                                                          |
| Client Side Encryption     | No                                                                             | No                                                                             | No                                                                                                   |
| Speed                      | Due to its LAN Sync feature, transfer of large data happens at fast pace.      | Transfer of larger files takes longer time.                                    | Similar speed, faster transfer of Microsoft native files due to block level copying.                 |
| Storage                    | 2 GB                                                                           | 15 GB                                                                          | 5 GB                                                                                                 |
| File Sync                  | Makes use of Block syncing to save file changes.                               | Slower syncing as compared to Dropbox.                                         | Limited file types can be backed up but more customization for backup of media files.                |
| File Sharing               | Allows file sharing across devices with the option of revoking access anytime. | Allows file sharing across devices with the option of revoking access anytime. | Sharing of file possible across devices with the option of revoking access.                          |
| Data Security              | Securely backs data to the cloud.                                              | Securely backs data to the cloud.                                              | Data is backed up securely to cloud; you can expect high data security as it is a Microsoft product. |

Table 3.4: Comparison of some cloud storage system

The research model that has been suggested is outlined as follows:

### 1. Data Collection:

There are two types of data collection:

**a. Primary Data:** This involves directly gathering information from organizations or individuals through surveys or interviews. Collect data regarding their existing cloud configurations, protective measures, encountered difficulties, and other relevant details.

**b. Secondary Data:** Incorporate pre-existing datasets, studies, and scholarly articles to complement the primary data.

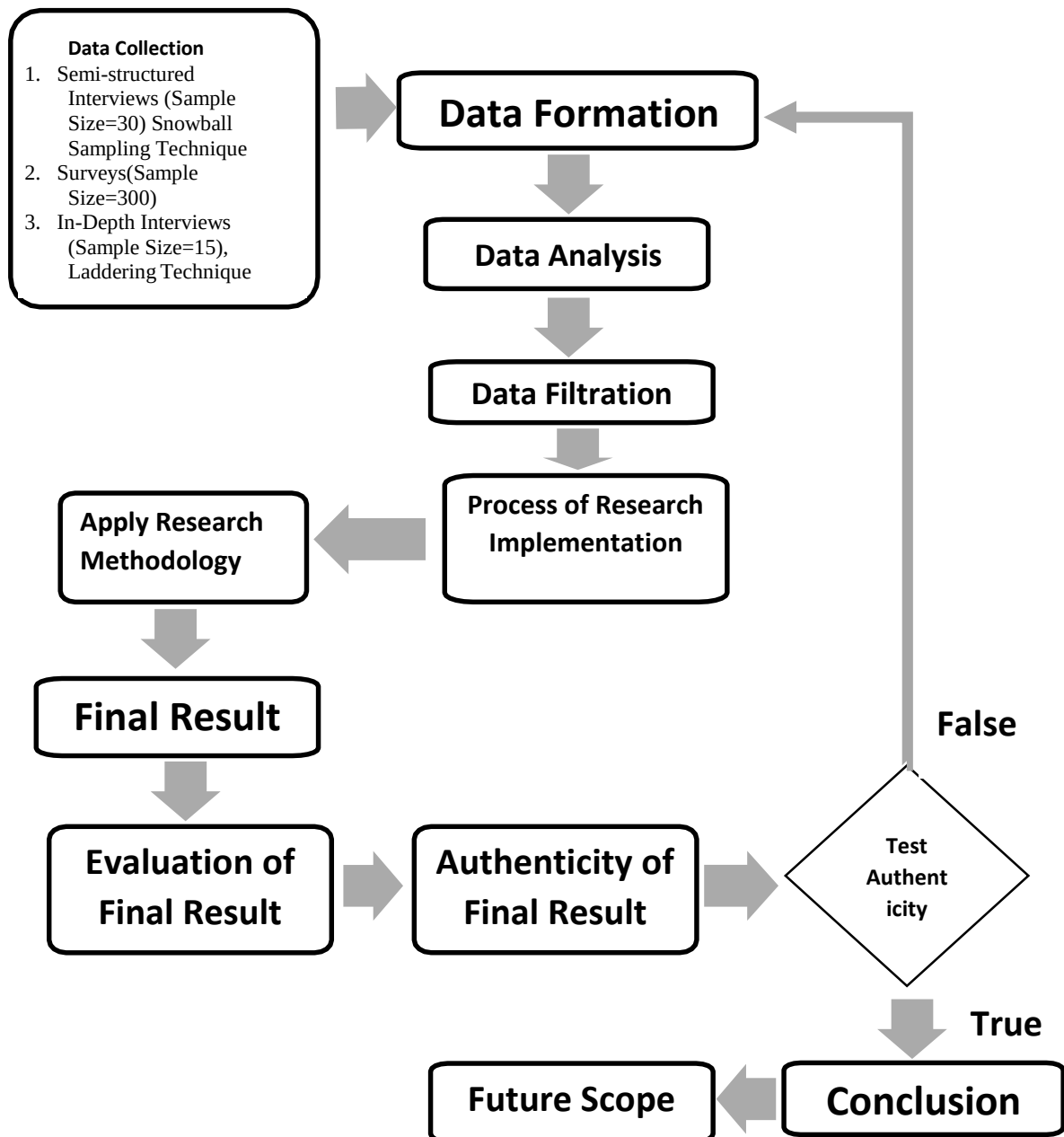


Figure 3.1: Research Methodology

## 2. Data analysis:

It involves the examination and interpretation of acquired data through the use of regression analysis.

**Regression analysis** is a statistical method used to quantify the relationship between variables and determine its influence. It provides solutions to the questions regarding the significant variables. Which item can be disregarded? What is the nature of their

interaction? Furthermore, the crucial aspect is in the level of certainty regarding these variables.

**The dependent variables** refer to the primary factors that we are attempting to comprehend or forecast.

### **Enumeration of Dependent variables**

**1. Security Level Attained:** This is a crucial characteristic that quantifies the level of security in the cloud environment. The level of security provided by a system depends on factors such as the strength of encryption protocols, the effectiveness of access control measures, the reliability of authentication mechanisms, and other related features.

**2. Cloud Service Budget:** The financial resources allotted or expended on cloud services are of utmost importance as they dictate the availability of funds for installing security measures and other features related to the cloud.

**3. Cloud Dependency Level:** The extent to which the company depends on the cloud for its operational and data management needs.

**4. Data Upload Size on Cloud:** The quantity of data that is consistently transferred to and stored in the cloud.

**5. Data Upload Frequency on Cloud:** The regularity with which data is transferred to the cloud.

**6. Bandwidth Requirement for Data Upload:** The necessary bandwidth for efficient data uploading to the cloud.

**7. Cloud User Count in Organization:** The total number of people in the organization who access or utilize cloud services.

**8. Security Level Attained:** The entire level of security attained through the implementation of security measures, encompassing integrity, privacy, confidentiality, and authenticity.

**9. Security Threats to Organization:** The various types and levels of security threats that the organization's cloud environment is exposed to.

### **Independent variables**

The independent variables are the elements that are believed to have an influence on the dependent variable.

### **Enumeration of independent variables**

**1. Organizational Type:** Various institutions, such as government, healthcare, and finance, may have distinct security and legal prerequisites.

**2. Employee Responsibility:** The duties and obligations of employees who engage with the cloud infrastructure and data.

**3. Cloud Service Provider:** The designated provider(s) responsible for delivering cloud services for deployment.

**4. Common reason for Utilizing Cloud Service:** The main business or operational factors that lead to the implementation of cloud services.

**5. Technical reason for Adopting Cloud:** The technical factors (such as scalability and flexibility) that drive the decision to choose cloud solutions.

**6. Cloud Deployment Model:** Specifies the type of cloud deployment, such as public, private, hybrid, or multi-cloud.

**7. Utilized Backup Drive:** The methods and strategies implemented to ensure data redundancy through backup solutions.

**8. Data Transmission Mode** refers to the specific method or protocol employed for the purpose of transmitting data to and from the cloud.

**9. Encryption Standard:** The cryptographic algorithms and protocols used to safeguard data.

**10. Upload of Sensitive Data:** The act of uploading sensitive or regulated data to the cloud.

**11. Authentication Method:** The techniques employed to confirm and validate the identities of individuals who are accessing cloud resources.

**12. Access Control Policy:** Policies that dictate the authorized individuals or entities who are allowed to access specific resources within the cloud environment.

**Types of Regression:**

1. There are different types of regression analysis, and one of them is simple or linear regression.

Simple linear regression is a statistical model that calculates the association between a response variable and a predictor variable by fitting a straight line.

**The equation is:**

$$y = m1.x + m2.z + c$$

Where y represents the dependent variable that requires prediction.

x represents the initial independent variable. This is the initial input. The value of m1 represents the slope of x1. It provides the measurement of the angle formed by the line (x). The variable z is the second independent variable. This is the second input. The slope of z is represented by m2. Having knowledge of the line's angle (z) is beneficial. The intercept of the line is represented by the variable c. A constant that determines the value of y when both x and z are equal to 0.

**2. Multiple Linear Regression:** Multiple linear regression is a statistical method used to determine the association between two or more independent variables and a single dependent variable. The distinction between simple and multiple linear regression models is in the quantity of independent variables.

Occasionally, the aforementioned regression models may be ineffective.

Regression analysis is mostly employed to comprehend the correlation between a dependent variable and an independent variable. In reality, there are several instances where multiple independent variables are influenced by other variables, necessitating the consideration of alternative options instead of relying solely on a single regression model that can only accommodate one independent variable.

Given these setbacks, the superior model that addresses the limitations of Simple and Multiple Linear Regression is Multivariate Regression.

**3. Multivariate Regression:** It is a supervised machine learning approach that analyzes many data variables. Multivariate regression is a statistical technique that expands upon multiple regression by incorporating one dependent variable and numerous independent variables. The prediction of the outcome is attempted based on the quantity of independent factors.

Multivariate regression aims to construct a formula that can elucidate how variables respond concurrently to changes in other variables.

$$y = \beta_0 + \beta_1.x_1 + \beta_2.x_2 + \dots + \beta_n.x_n$$

- The equation is represented as y equals the sum of  $\beta_0$ ,  $\beta_1$  times  $x_1$ ,  $\beta_2$  times  $x_2$ , and so on up to  $\beta_n$  times  $x_n$ .
- The variables in the equation are represented as follows: n denotes the number of independent variables,  $\beta_0 \sim \beta_n$  represents the coefficients, and  $x_1 \sim x_n$  represents the independent variables.
- The obtained data includes a variety of dependent and independent factors, which are used to analyze the relationships between these variables and the resulting outcomes. For example, when a certain level of security has been reached, Multivariate regression can be employed.

### 3.5 Survey Questionnaires

The survey questions are categorized into several keywords to promptly detect issues.

The following items are:

1. Demographic details
2. Technical Details
3. Cloud service-related details
4. Security details of cloud services.

# Questionnaire

## ORGANIZATIONS AVAILING CLOUD SERVICES

This survey aims to gain insights about the utilization of cloud services in various enterprises, considering different aspects. Please respond to all the questions provided in the survey.

### Section I. Demographic Question

1. Email ID

.....

2. Name

.....

3. Company Name

.....

4. Location (Mark Only One)

- ☐ New Delhi
- ☐ Bangalore
- ☐ Pune
- ☐ Navi Mumbai
- ☐ Mumbai
- ☐ Indore
- ☐ Kolkatta
- ☐ Channai
- ☐ Other

5. Type of Organization (Check all that apply)

- ☐ Telecommunication
- ☐ Technology Services
- ☐ Financial / Banking
- ☐ Education
- ☐ Healthcare
- ☐ Government
- ☐ Retail
- ☐ Transportation
- ☐ Other

6. Which level do you hold in your organization (Make only one oval?)
- ☐ Executive Level
  - ☐ Senior Level
  - ☐ Middle Level
  - ☐ Lower Level
7. What is your degree of responsibility for your organization's cloud services? (Mark only one oval)
- ☐ Primary Responsibility
  - ☐ Secondary Responsibility
  - ☐ Sometime Responsible, whenever needed
  - ☐ Not Responsible

## **Section II: Technical Questions**

8. What are the technical reasons for availing Cloud Services (Check all that apply)
- ☐ Data Backup
  - ☐ Data Compression
  - ☐ Load balancing
  - ☐ On Demand Service
  - ☐ Shared Infrastructure
  - ☐ Fault Tolerance
  - ☐ Clustering
9. What is the required bandwidth for data upload on cloud? (Mark only one oval)
- ☐ 25 Mbps
  - ☐ 50 Mbps
  - ☐ 100 Mbps
  - ☐ 200 Mbps
10. Which data transmission mode is used by your organization? (Mark only one oval)
- ☐ Simplex
  - ☐ Half Duplex
  - ☐ Full Duplex
11. Which authentication method is used while availing cloud service?
- ☐ Multifactor Authentication
  - ☐ Two Factor Authentication
  - ☐ Three Factor Authentication
  - ☐ OTP
  - ☐ Password

12. Which access control policy is used while availing cloud service?

- ☐ Mandatory Access Control (MAC)
- ☐ Discretionary Access Control (DAC)
- ☐ Role Base Access Control (RBAC)
- ☐ Rule Based Access Control (RAC)
- ☐ Attribute Based Access Control (ABAC)

### **Section III: Cloud Service-Related Questions**

13. Which type of cloud service were most common in your organization over past 3 years (Check all that apply)

- ☐ Storage as Service
- ☐ Platform as Service
- ☐ Security as Service
- ☐ Software as Service
- ☐ Infrastructure as Service
- ☐ Database as Service
- ☐ Testing as Service
- ☐ None

14. What percentage of IT budget is allocated to cloud service in your organization (Mark only one oval)

- ☐ >10%
- ☐ 10 – 20%
- ☐ 20 - 30%
- ☐ 30 – 40%
- ☐ 40 - 50%
- ☐ > 50%

15. What is the level of dependency of cloud services? (Mark only one oval)

- ☐ Fully?
- ☐ Partial
- ☐ Occasional
- ☐ No concern

16. What is the size of data uploaded on cloud? (Mark only one oval)

- ☐ 1 TB
- ☐ 2 TB
- ☐ 3 TB
- ☐ 4 TB
- ☐ 5 TB

17. What is the frequency of data upload on cloud? (Mark only one oval)
- ☐ Daily
  - ☐ Alternate Day
  - ☐ Twice a Week
  - ☐ Weekly
  - ☐ Fortnightly
18. How many users in your organization use cloud service. (Mark only one oval)
- ☐ 10
  - ☐ 20
  - ☐ 50
  - ☐ 100
  - ☐ More than 100
19. Which Cloud deployment model is used? (Mark only one oval)
- ☐ Public Cloud?
  - ☐ Private Cloud
  - ☐ Community Cloud
  - ☐ Hybrid Cloud
  - ☐ Multi Cloud
20. What is the average response by CSP after threat detection (Mark only one oval)
- ☐ 1 Hour
  - ☐ 1 Day
  - ☐ > 1 Day
  - ☐ 1 Week
  - ☐ > 1 Week

#### **Section IV: Security Related Questions**

21. What are the most common reason for availing cloud service (Check all that apply)
- ☐ Security Issue
  - ☐ Privacy Issue
  - ☐ Availability
  - ☐ Accessibility
  - ☐ Cost Saving
  - ☐ Other
22. What level of security gain with the use of cloud services (Mark only one oval)
- ☐ High
  - ☐ Critical
  - ☐ Personal
  - ☐ Low

23. Which backup drive is used? (Mark only one oval)

- ☐ Acronise
- ☐ BackBlaze
- ☐ CrashPlan
- ☐ DropBox
- ☐ IDrive
- ☐ OneDrive

24. Frequency of data backup on cloud

- ☐ Daily
- ☐ Alternate Day
- ☐ Twice a Week
- ☐ Weekly
- ☐ Fortnightly

25. Which data encryption standard is used by CSP for your organization (Mark only one oval)

- ☐ AES?
- ☐ DES
- ☐ RSA
- ☐ ECC
- ☐ TSL

26. Did the employee in your organization store sensitive data on cloud?

- ☐ YES
- ☐ No

27. What level of security gain by the use of cloud services (Mark only one oval)

- ☐ Confidentiality
- ☐ Integrity
- ☐ Non-Repudiation
- ☐ Availability
- ☐ Authenticity
- ☐ Reliability

28. What are the threats need to face during the use of cloud service (Check all that apply)

- ☐ Data Breach
- ☐ Data Leakage and Loss
- ☐ Insecure API's
- ☐ Account Hijacking
- ☐ Daniel of Service
- ☐ Abuse of Cloud Planning
- ☐ Inadequate Cloud Planning
- ☐ Cloud Related Malware

- ☐ Closer of Cloud Service
- ☐ Natural Disaster
- ☐ Hardware Failure
- ☐ Shared Technology Vulnerabilities

29. What is the most important factor hindering use of cloud services? (Check all that apply)

- ☐ Resource Crunch
- ☐ Insufficient Training
- ☐ Lack of Information
- ☐ New Automated Services
- ☐ Other

### 3.5.1 Survey on demographic details

#### a. Location

Cloud services are utilized throughout India, with Pune, Bangalore, and New Delhi being the primary adopters of cloud technology. While Chennai and Mumbai are significant metropolitan centers, the majority of the responders, accounting for 62%, are from Pune, Bengaluru, and New Delhi. Certain participants were provided with online surveys to complete, while others completed them in physical, printed formats. Data points were collected through interviews.

| Location     | Count      |
|--------------|------------|
| Pune         | 97         |
| Bengluru     | 81         |
| New Delhi    | 72         |
| Chennai      | 51         |
| Mumbai       | 26         |
| Navi Mumbai  | 23         |
| Kolkotta     | 18         |
| Indore       | 15         |
| Other        | 12         |
| <b>Total</b> | <b>395</b> |

Table 3.5: Count of locations of respondents

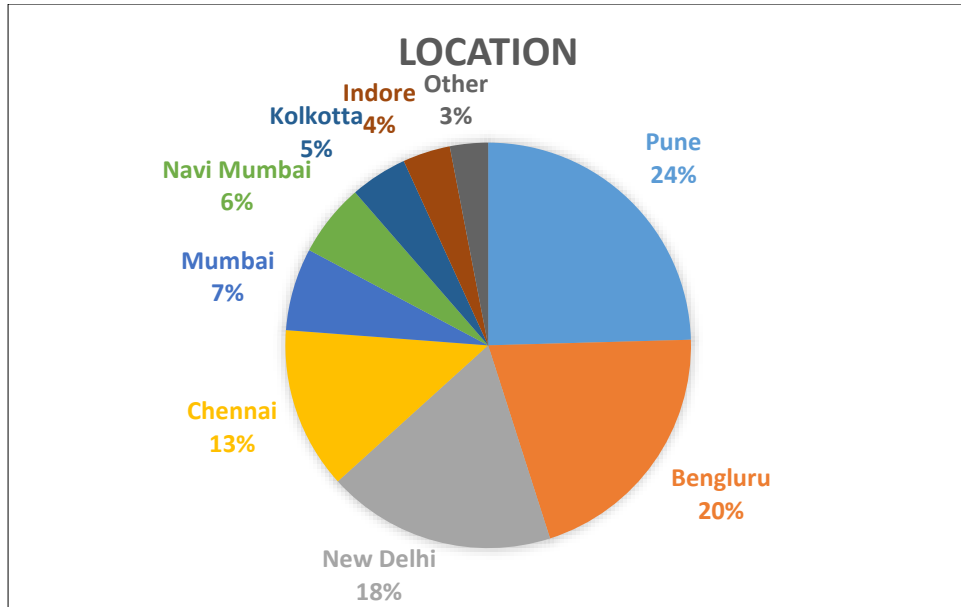


Figure 3.2: Location of organizations

#### b. Type of Organization

The primary industries targeted among the responders are Telecommunications, Technology, and Financial/Banking services. While cloud services are available to all types of enterprises, the aforementioned organizations provide attractive resources to service providers. Therefore, these organizations constitute 65% of the organizations who responded to our survey.

| Type of Organization | Count      |
|----------------------|------------|
| Telecommunication    | 99         |
| Technology Services  | 86         |
| Financial / Banking  | 73         |
| Education            | 63         |
| Healthcare           | 22         |
| Government           | 16         |
| Retail               | 13         |
| Other                | 12         |
| <b>Total</b>         | <b>395</b> |

Table 3.6: Type of organizations of respondents

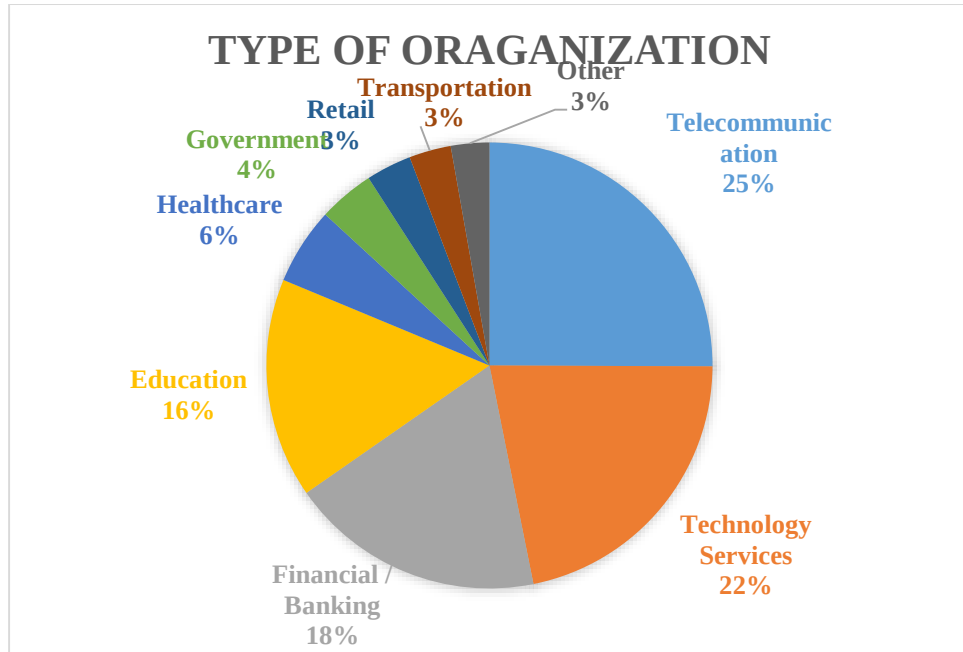


Figure 3.3: Companies Covered, location and role of respondents during Data Collection and Analysis

**c. Level of designation**

Among all the participants, 26% were senior-level personnel. 23% of individuals were classified as medium level, while 38% were classified as executive level. These individuals held influential positions inside the firm and were the primary consumers of cloud services. Additionally, they held the primary responsibility for formulating security policies and ensuring compliance throughout the firm.

| Title level     | Count |
|-----------------|-------|
| Executive Level | 149   |
| Senior Level    | 103   |
| Middle Level    | 91    |
| Lower Level     | 52    |
| Total           | 395   |

Table 3.7: Count of title level of respondents

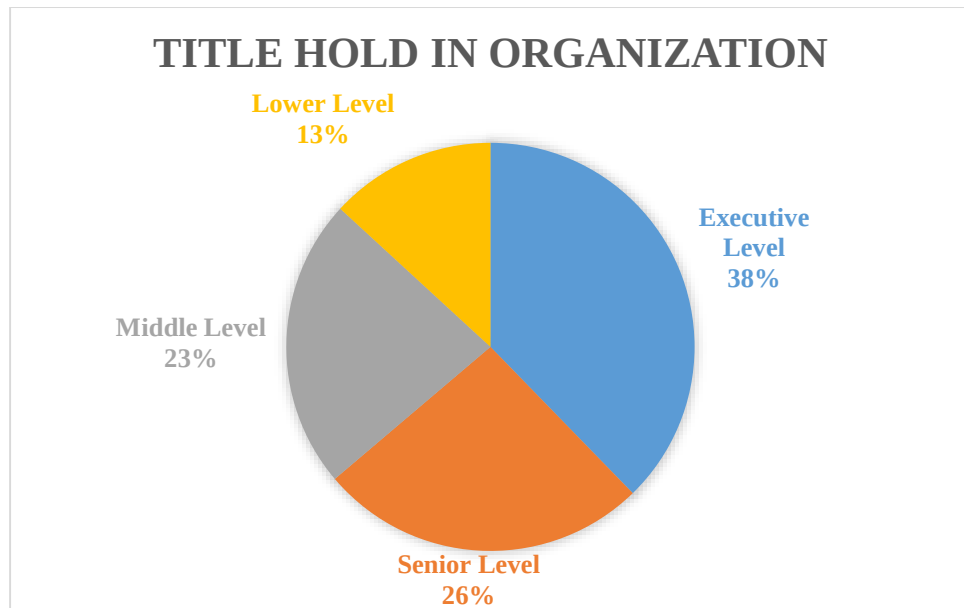


Figure 3.4: Title level hold in the organization

**d. Degree of security responsibility**

Among the respondents, 39% had primary responsibility for security in their firms. 28% had a secondary responsibility for security, while just 12% were occasionally responsible for security, but not usually.

| Degree of responsibility              | Count |
|---------------------------------------|-------|
| Primary Responsibility                | 153   |
| Secondary Responsibility              | 112   |
| Sometime Responsible, Whenever needed | 84    |
| Not Responsible                       | 46    |
| Total                                 | 395   |

Table 3.8: Count of degree of responsibilities of respondents

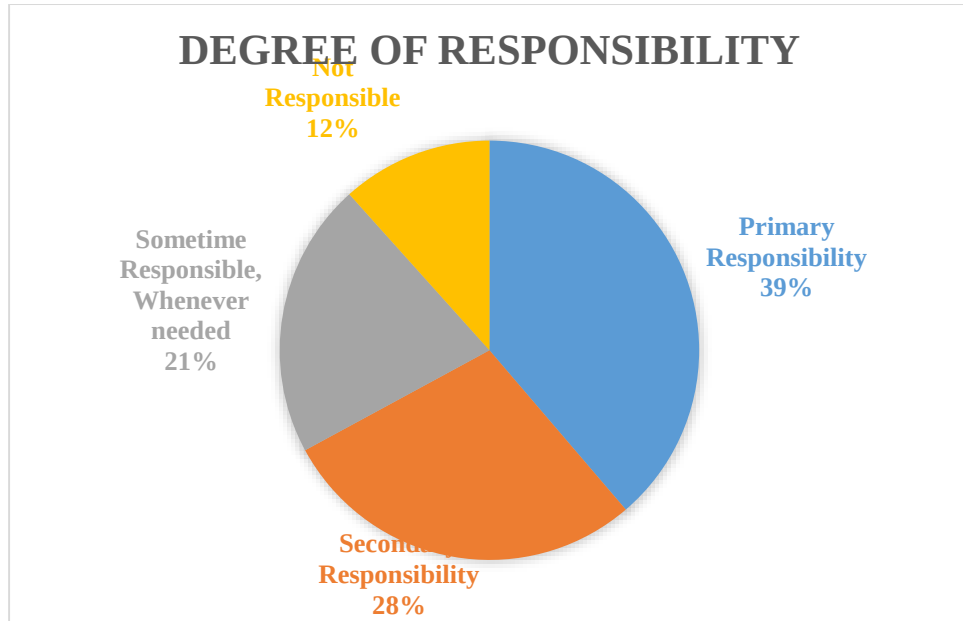


Figure 3.5: Degree of responsibility of respondent

### 3.5.2 Survey on Technical details

#### a. Technical Reason for Availing Cloud Service

The majority of organizations utilize cloud services primarily for data backup (25%) or data compression (23%). The On Demand service represents 20% of Cloud Services, and cost savings are typically not a significant factor.

| Technical Reason      | Count |
|-----------------------|-------|
| Data Backup           | 99    |
| Data Compression      | 92    |
| Load Balancing        | 28    |
| On Demand Service     | 81    |
| Shared Infrastructure | 71    |
| Fault Tolerance       | 13    |
| Clustering            | 11    |
| Total                 | 395   |

Table 3.9: Count of Technical Reason for Availing Cloud Service

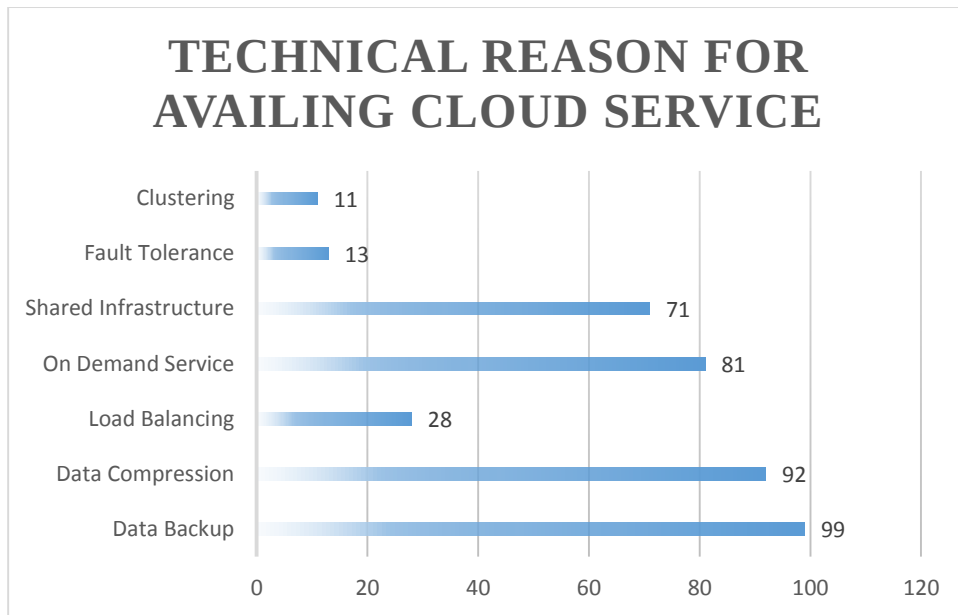


Figure 3.6: Technical Reason for Availing Cloud Service

#### b. Bandwidth required for data upload on cloud

Among the entire pool of respondents, 17% indicate that their firms necessitate a bandwidth of up to 200Mbps for uploading data to the cloud. 31% of respondents believe that the speed should be 100Mbps. 25% of respondents indicate that their firms have a requirement of 50Mbps bandwidth for data upload on the cloud, while 24% of respondents' state that their organizations require 25Mbps bandwidth for data upload on the cloud.

| Bandwidth Required | Count |
|--------------------|-------|
| 25 Mbps            | 96    |
| 50 Mbps            | 102   |
| 100 Mbps           | 126   |
| 200 Mbps           | 71    |
| Total              | 395   |

Table 3.10: Bandwidth required for data upload on cloud

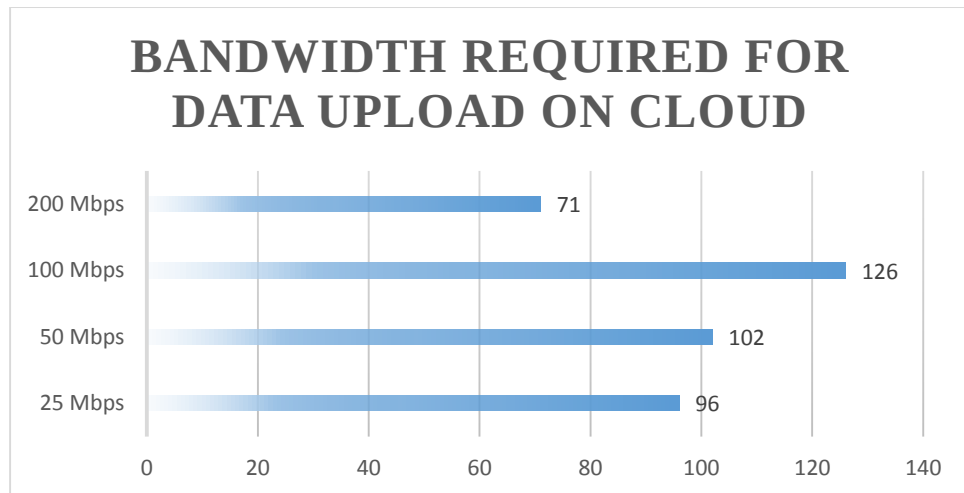


Figure 3.7: Bandwidth require for data upload on cloud.

#### c. Data Transmission mode used by Organization

Among the entire pool of responders, 36% utilized the Full Duplex data transmission option. 12% of data transmission utilizes simplex communication. Only 50% of users utilize the Half Duplex data transfer technique.

| Transmission Mode | Count      |
|-------------------|------------|
| Simplex           | 51         |
| Half Duplex       | 201        |
| Full Duplex       | 143        |
| <b>Total</b>      | <b>395</b> |

Table 3.11: Data Transmission mode used by organization

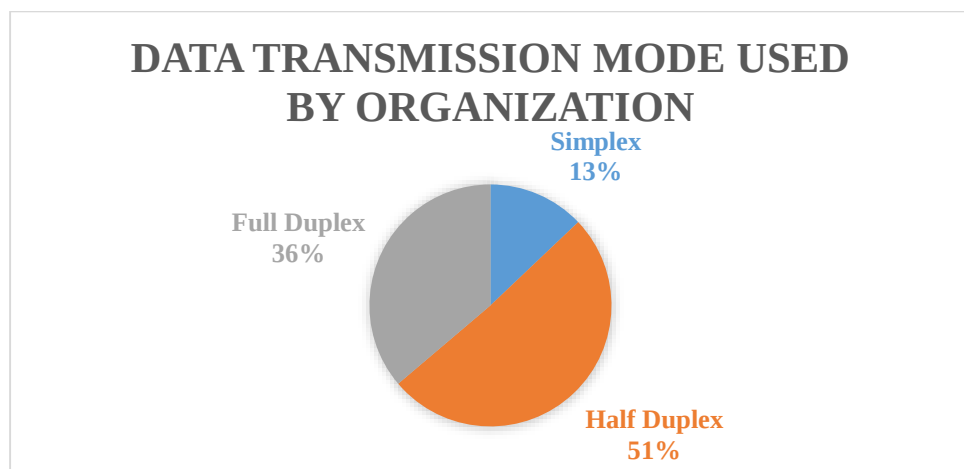


Figure 3.8: Data Transmission mode used by organization

#### d. Authentication Method used by CSP

27% of cloud service providers utilize two-factor authentication, while 23% rely on password authentication and 20% employ multifactor authentication.

| Authentication Method       | Count |
|-----------------------------|-------|
| Multifactor Authentication  | 81    |
| Two Factor Authentication   | 108   |
| Three Factor Authentication | 59    |
| OTP                         | 54    |
| Password                    | 93    |
| Total                       | 395   |

Table 3.12: Authentication Method used by CSP

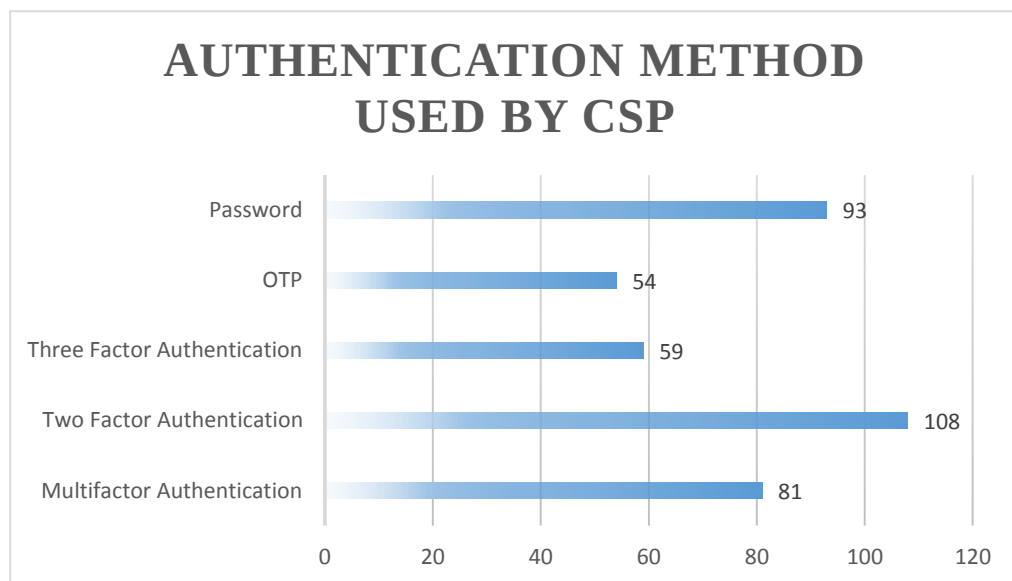


Figure 3.9: Authentication method used by CSP

#### e. Access Control Policies used by CSP

The access control policy of MAC is used by 27% of the CSP 24% of users choose DAC (Discretionary Access Control) and 20% utilize RAC (Role-Based Access Control) as access control policies to prevent unauthorized individuals from accessing the data stored on the cloud.

| Access Control Policy                 | Count      |
|---------------------------------------|------------|
| Mandatory Access Control (MAC)        | 107        |
| Discretionary Access Control (DAC)    | 97         |
| Role Base Access Control (RBAC)       | 57         |
| Rule Based Access Control (RAC)       | 82         |
| Attribute Based Access Control (ABAC) | 52         |
| <b>Total</b>                          | <b>395</b> |

Table 3.13: Access Control Policies used by CSP

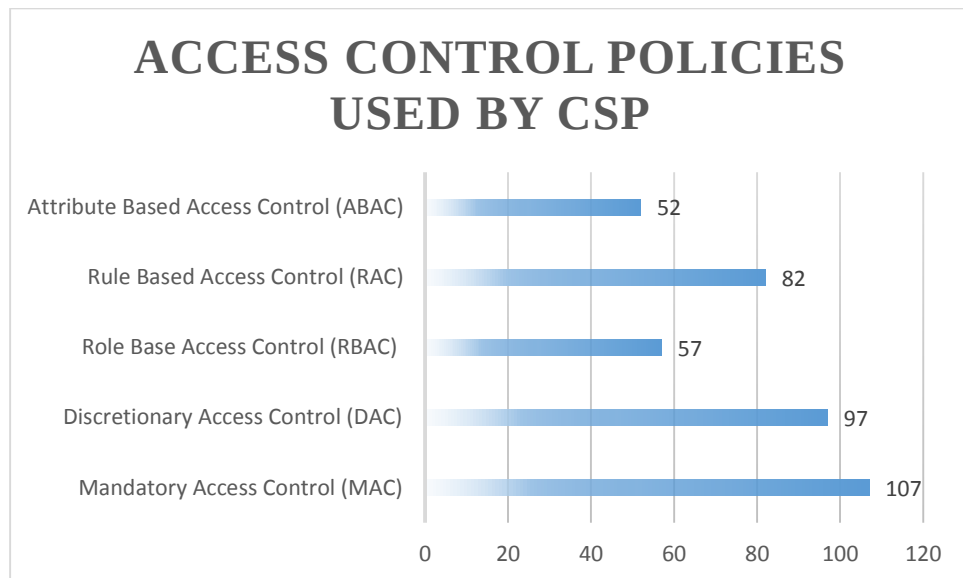


Figure 3.10: Access Control Policies used by CSP

### 3.5.3 Survey on cloud service details

#### a. Type of cloud service

A company can access many types of cloud services. Of them, 25% are classified as Storage service. Organizations can avail database as a service 5% of the time. Just 13% of users utilize software as a service, while only 9% utilize infrastructure as a service. The term "three years" refers to the time span encompassing the years 2020, 2021, and 2022.

| Type of Cloud Service     | Count |
|---------------------------|-------|
| Storage as Service        | 99    |
| Platform as Service       | 79    |
| Security as Service       | 78    |
| Software as Service       | 53    |
| Infrastructure as Service | 38    |
| Database as Service       | 23    |
| Testing as Service        | 14    |
| None                      | 11    |
| Total                     | 395   |

Table 3.14: Type of cloud Service used by respondents

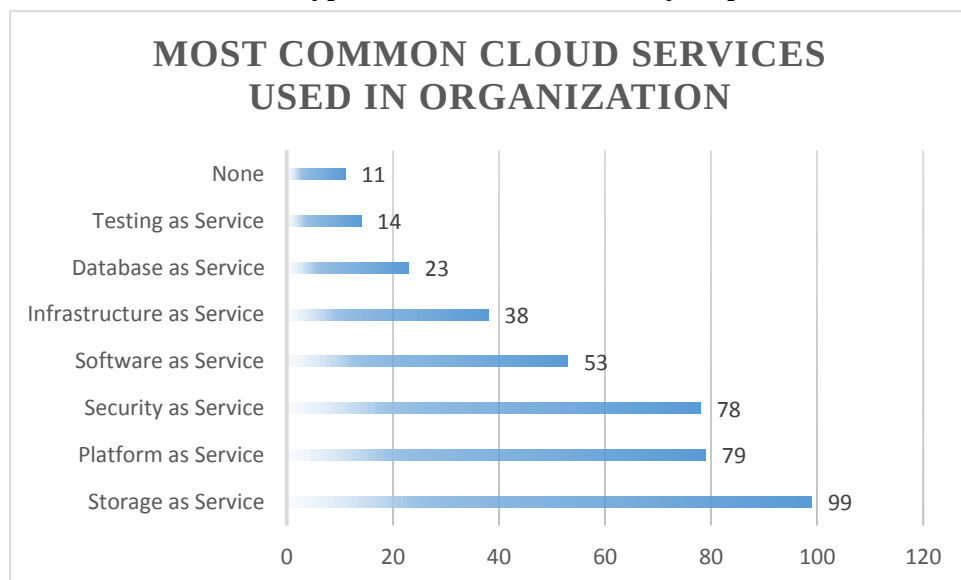


Figure 3.11: Type of Cloud Services

#### b. Percentage of IT budget allocated to Cloud Service

A majority of organizations (19%) invest less than 10% of their IT expenditure to Cloud Services. Only 22% of organizations invest 10-20% of their budget to Cloud Service. Among all the respondents, there were companies that allocated 27% of their budget to Cloud Service.

| <b>% of IT Budget</b> | <b>Count</b> |
|-----------------------|--------------|
| >10%                  | 76           |
| 10 – 20%              | 87           |
| 20 - 30%              | 109          |
| 30 – 40%              | 98           |
| 40 - 50%              | 13           |
| > 50%                 | 12           |

Table 3.15: Percentage of IT budget allocated

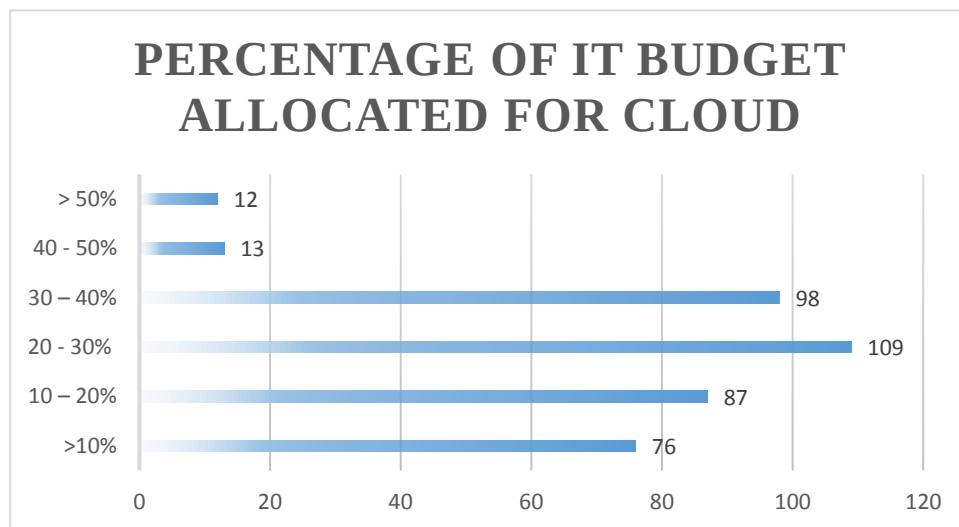


Figure 3.12: Percentage of budget allocated for Cloud Service

### c. Level of dependency of Cloud Services

18% of the participants hold the belief that organizations do not prioritize Cloud Services. Only 31% of firms have a moderate level of concern, while 24% have an intermittent level of concern over Cloud Services.

| <b>Level of dependency</b> | <b>Count</b> |
|----------------------------|--------------|
| Fully                      | 99           |
| Partial                    | 123          |
| Occasional                 | 98           |
| No concern                 | 75           |
| Total                      | 395          |

Table 3.16: Level of dependency for Cloud Services

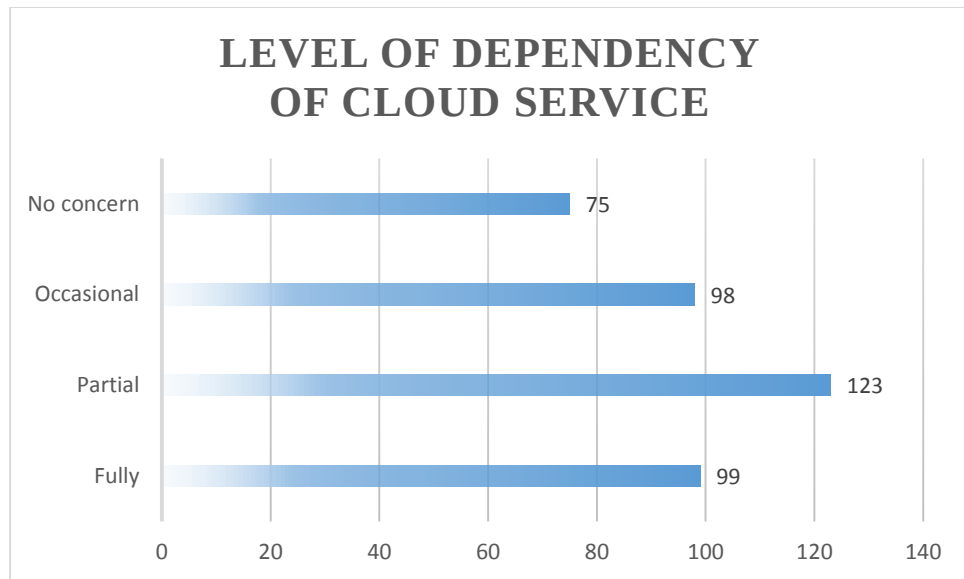


Figure 3.13: Level of dependency of Cloud Services

#### d. Size of Data Uploaded on Cloud

Among the entire range of enterprises surveyed, 13% of them upload a total of 1 terabyte of data. 25% of the total data, which amounts to 5 terabytes, will be uploaded to the cloud on both servers and all platforms. Just 26% of users transfer 4 terabytes of data onto the cloud.

| Size of Data Upload | Count |
|---------------------|-------|
| 1 TB                | 53    |
| 2 TB                | 58    |
| 3 TB                | 82    |
| 4 TB                | 103   |
| 5 TB                | 99    |

Table 3.17: Size of Data uploaded on cloud

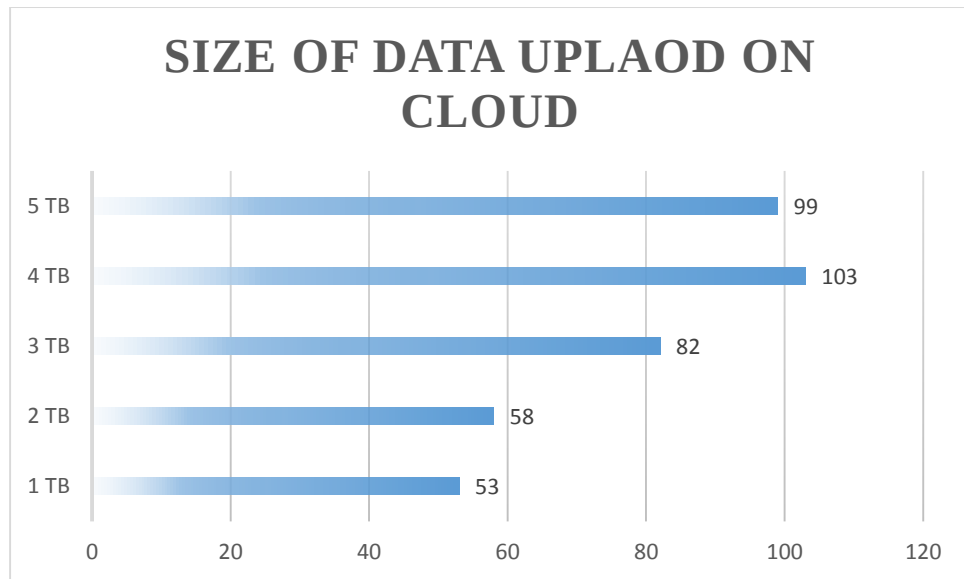


Figure 3.14: Size of Data uploaded on Cloud

#### e. Frequency of Data upload on Cloud

Among all the respondents, 26% regularly upload data to the cloud on a daily basis. Approximately the same proportion of uploads, specifically 24% and 20%, occur every other day and twice a week, respectively.

| Frequency     | Count |
|---------------|-------|
| Daily         | 104   |
| Alternate Day | 96    |
| Twice a Week  | 79    |
| Weekly        | 59    |
| Fortnightly   | 57    |
| Total         | 395   |

Table 3.18: Frequency of data upload on cloud

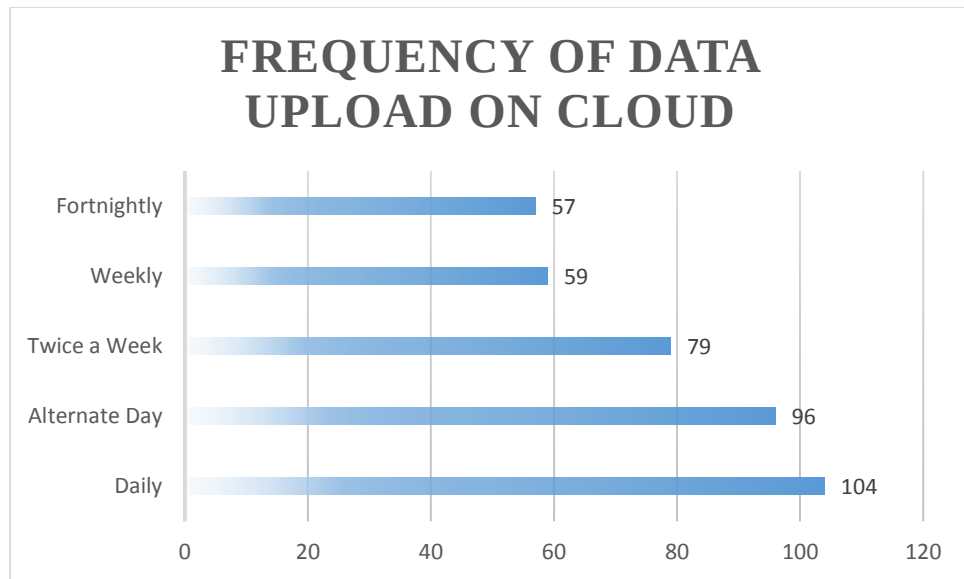


Figure 3.15: Frequency of data upload on cloud

**f. Users in Organization use cloud service**

Among the respondents, 25% reported that in their firm, there are 50 customers utilizing cloud services. According to the data, 25% of the total number of users, which is 100, utilize cloud services. According to the data, 21% of users, which is more than 100 individuals, utilize cloud services. Only 13% of respondents reported that they have 10 people utilizing cloud services.

| No. of Users | Count |
|--------------|-------|
| 10           | 54    |
| 20           | 58    |
| 50           | 99    |
| 100          | 99    |
| >100         | 85    |
| Total        | 395   |

Table 3.19: Users in Organization use cloud Service

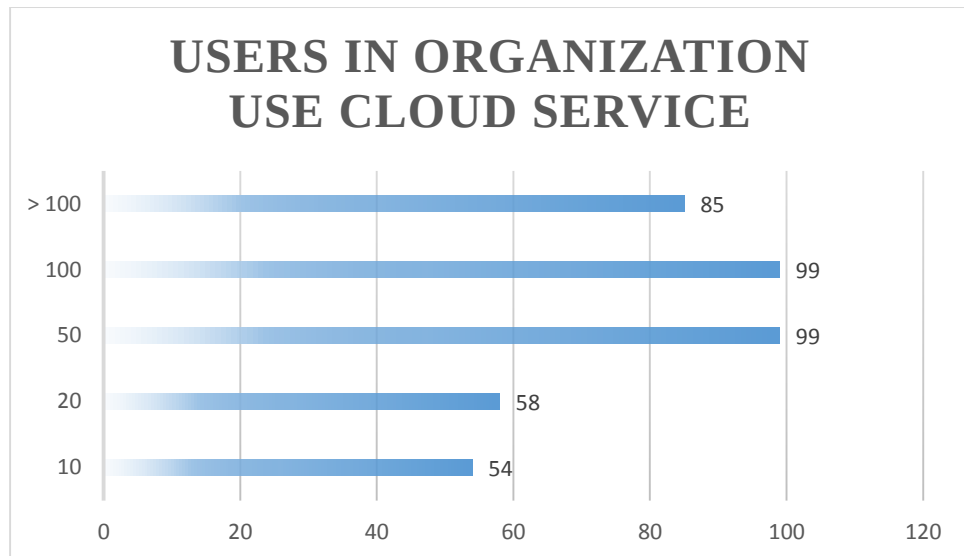


Figure 3.16: Users in Organization use cloud services

**g. Cloud Deployment Model used**

Among all the participants, 23% employ the Hybrid Cloud method. The usage of Public Cloud and Community Cloud is nearly equal, with 26% and 14% respectively.

| Model           | Count      |
|-----------------|------------|
| Public Cloud    | 106        |
| Private Cloud   | 84         |
| Community Cloud | 59         |
| Hybrid Cloud    | 93         |
| Multi Cloud     | 53         |
| <b>Total</b>    | <b>395</b> |

Table 3.20: Cloud Deployment Model used

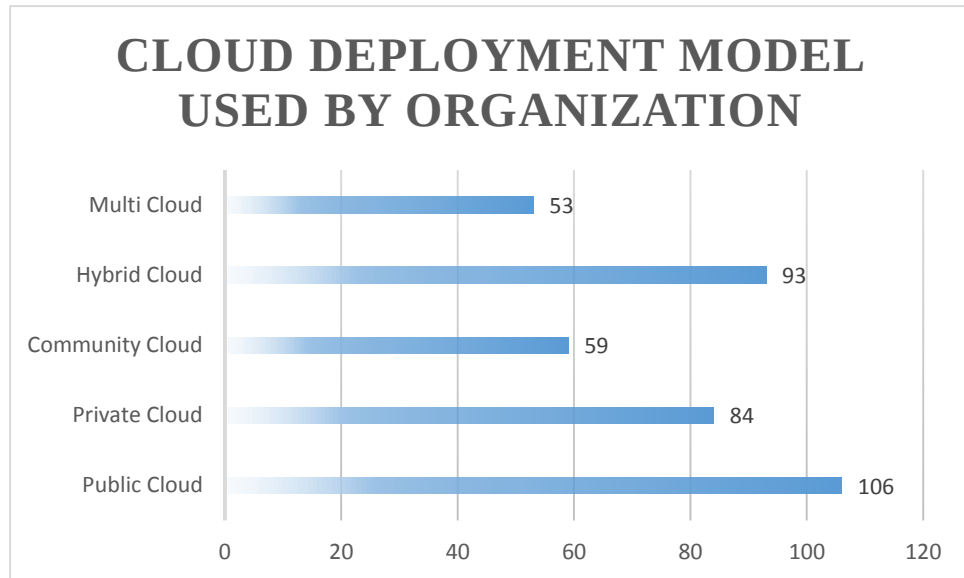


Figure 3.17: Cloud Deployment Model used

#### h. Response by CSP after threat detection

26% of firms report receiving a response from their Cloud Service Provider (CSP) within one hour of threat detection. 22% state that they receive a response within one day, while 21% indicate that they receive a response after one day. 16% of individuals receive a response within one week, while 13% receive a response after one week.

| Response By CSP | Count      |
|-----------------|------------|
| 1 Hour          | 103        |
| 1 Day           | 89         |
| > 1 Day         | 83         |
| 1 Week          | 66         |
| > 1 Week        | 54         |
| <b>Total</b>    | <b>395</b> |

Table 3.21: Response given by CSP after threat detection

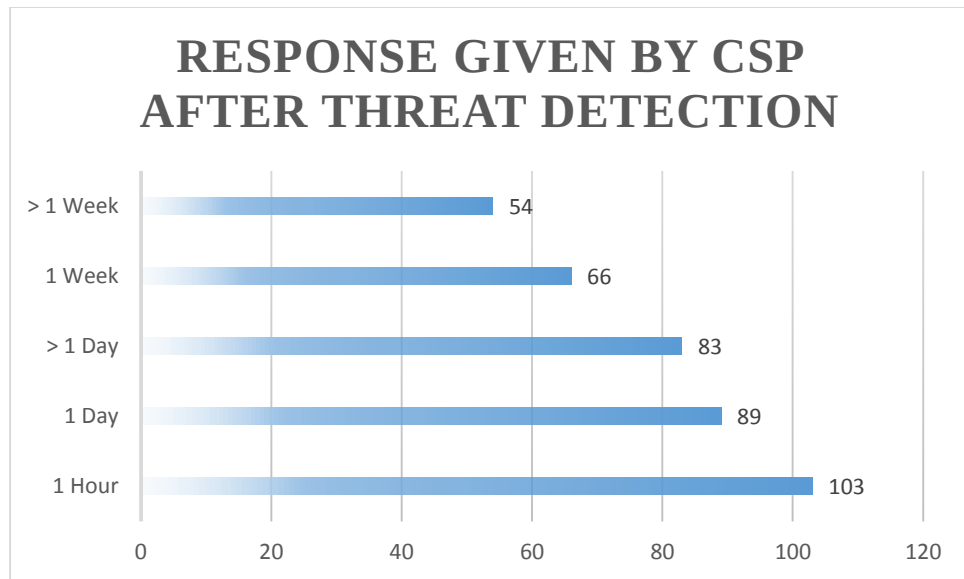


Figure 3.18: Response given by CSP after threat detection

### 3.5.4 Survey on security details of cloud service

#### a. Common Reason for Availing Cloud Services

The majority of organizations choose cloud services primarily for security purposes (25%) or for privacy concerns (20%). Accessibility represents 23% of the factors considered for Cloud Services, whereas cost savings often have minimal significance.

| Common Reason  | Count |
|----------------|-------|
| Security Issue | 102   |
| Privacy Issue  | 81    |
| Availability   | 79    |
| Accessibility  | 92    |
| Cost Saving    | 27    |
| Other          | 14    |
| Total          | 395   |

Table 3.22: Count of common reason for availing cloud service

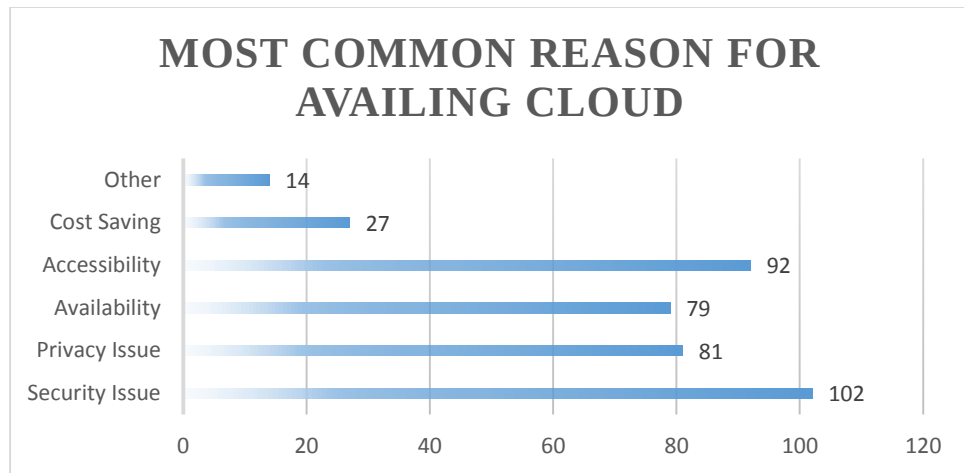


Figure 3.19: Common Reason for availing Cloud service

#### b. Level of Security gain by organization

When working with cloud technology, it is limited to high security measures 27% of the time. Approximately 23% of the time, the reason is Critical security, most likely because of network segmentation. In a mere 9% of instances, the level of security is low.

| Level    | Count |
|----------|-------|
| High     | 109   |
| Critical | 91    |
| Personal | 122   |
| Low      | 73    |
| Total    | 395   |

Table 3.23: Count of level till which devices get affected

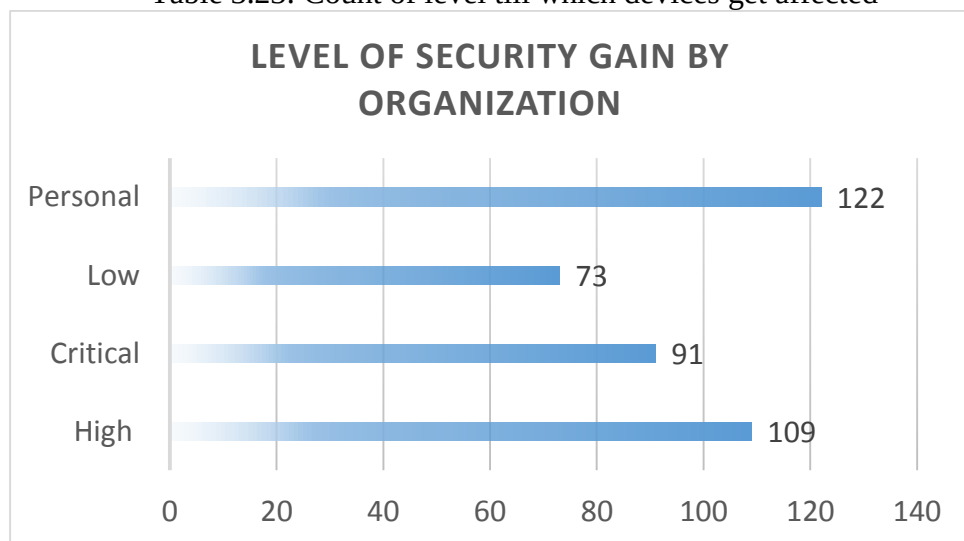


Figure 3.20: Level of security gain by organization

### c. Backup Drive used by Organization

Among the entire pool of respondents, 27% utilized DropBox as a means of backing up their data. 21% of users using Idrive. 25% of users utilize OneDrive as their backup disk, whereas 19% of users rely on BackBlaze for this purpose. Just 3% of users utilize CrashPlan as their backup solution.

| Backup Drive | Count      |
|--------------|------------|
| Acronise     | 13         |
| BackBlaze    | 76         |
| CrashPlan    | 14         |
| DropBox      | 107        |
| IDrive       | 86         |
| OneDrive     | 99         |
| <b>Total</b> | <b>395</b> |

Table 3.24: Backup Drive used by Organization

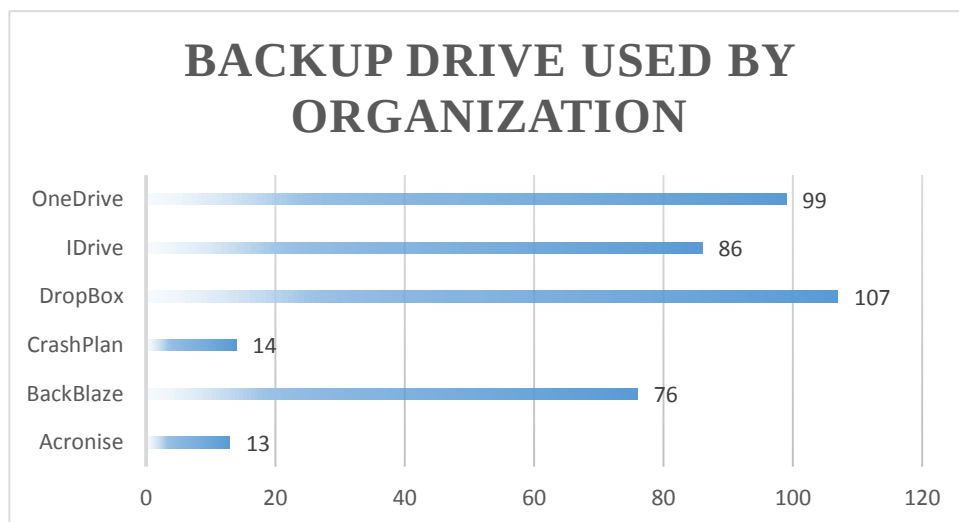


Figure 3.21: Backup Drive used by Organization

### d. Data Backup Frequency in organization on cloud

Among the entire pool of respondents, 29% engage in weekly cloud backup. 25% is the amount of tales that occur on a daily basis. 15% of individuals take medication every other day, and 21% use medication twice a week as a backup. Just 8% of individuals take backups every two weeks.

| Frequency     | Count      |
|---------------|------------|
| Daily         | 102        |
| Alternate Day | 61         |
| Twice a Week  | 83         |
| Weekly        | 117        |
| Fortnightly   | 32         |
| <b>Total</b>  | <b>395</b> |

Table 3.25: Data Backup Frequency in organization on cloud

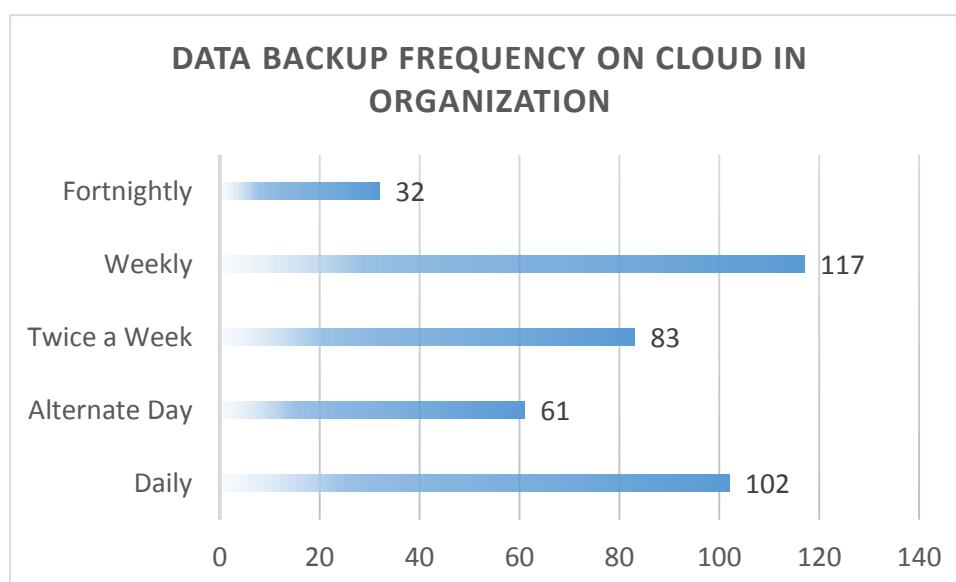


Figure 3.22: Data backup frequency on cloud in organization

#### e. Data Encryption Standard Provided by CSP

27% of the respondents feel that enterprises use AES as the Data Encryption Standard. Only 23% of firms prioritize the RSA Data Encryption Standard, whereas 13% prioritize the TSL Data Encryption Standard.

| Encryption Standard | Count      |
|---------------------|------------|
| AES                 | 108        |
| DES                 | 76         |
| RSA                 | 93         |
| ECC                 | 65         |
| TSL                 | 53         |
| <b>Total</b>        | <b>395</b> |

Table 3.26: Data Encryption Standard provided by CSP

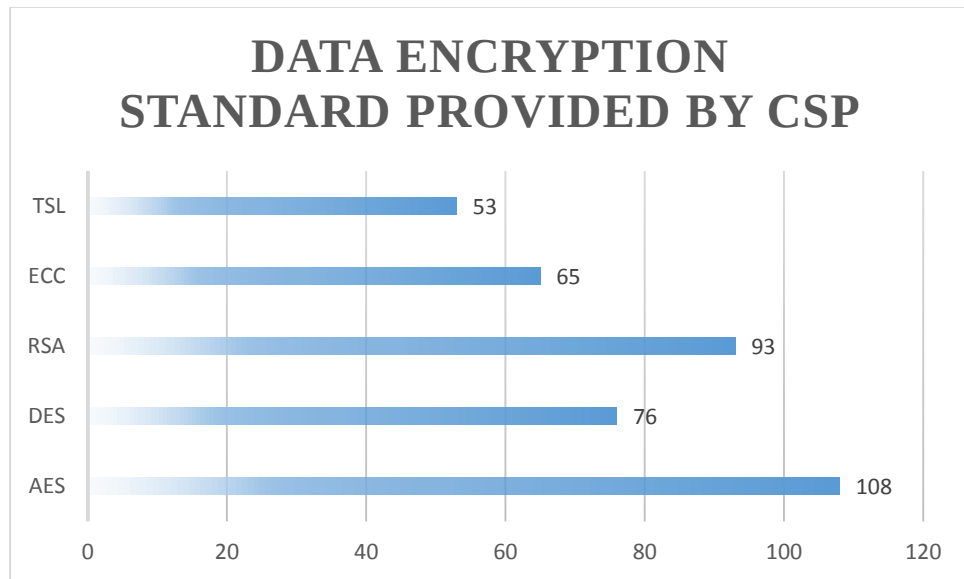


Figure 3.23: Data Encryption Standard provided by CSP

**f. Does the user uploads sensitive data on cloud**

Approximately 58% of users often transfer their sensitive data to the cloud, making the security and privacy of data a crucial aspect of cloud storage systems.

| <b>Sensitive Data</b> | <b>Count</b> |
|-----------------------|--------------|
| YES                   | 231          |
| No                    | 164          |
| Total                 | 395          |

Table 3.27: Sensitive data uploaded on cloud by user

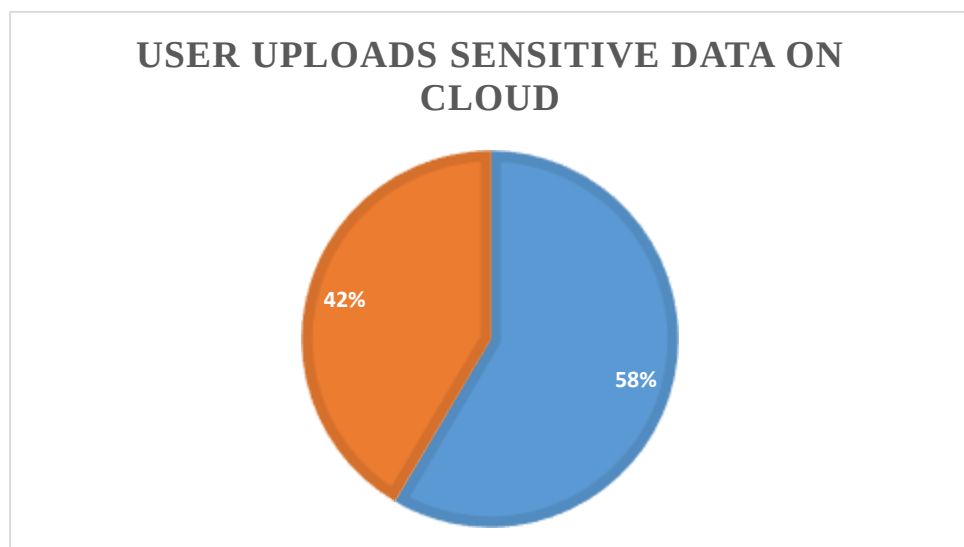


Figure3.24: User uploads sensitive data on cloud

#### g. Level of Security Gain by the use of Cloud Services

By utilizing cloud services 24% of organizations report that they achieve secrecy. Integrity experiences a 23% increase, while Authenticity experiences an 18% increase. Additionally, both aspects acquire availability as security measures.

| Level of Security | Count      |
|-------------------|------------|
| Confidentiality   | 98         |
| Integrity         | 91         |
| Non-Repudiation   | 13         |
| Availability      | 69         |
| Authenticity      | 72         |
| Reliability       | 52         |
| <b>Total</b>      | <b>395</b> |

Table 3.28: Level of Security Gain by Organization

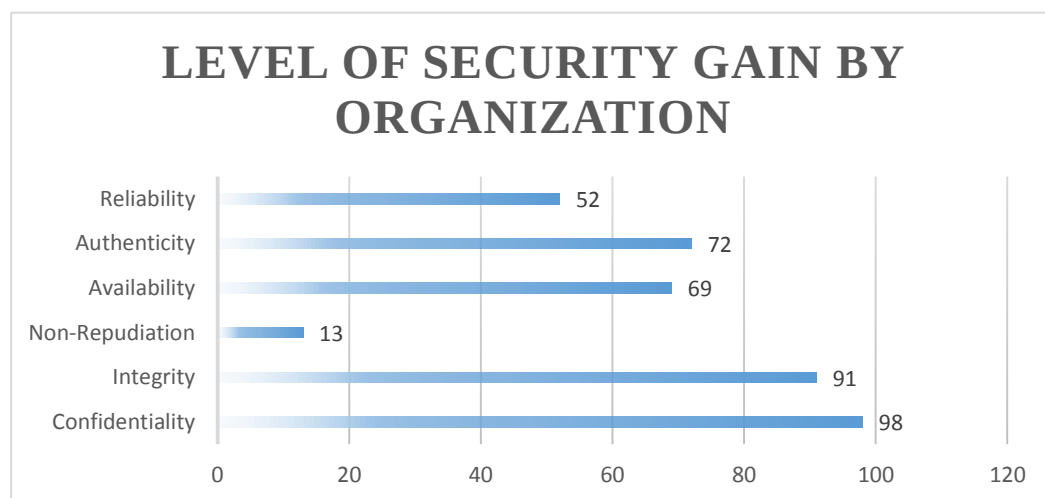


Figure 3.25: Level of Security gain by organization

#### h. Threats Faced during the use of cloud service

The 16% organizations are in the fear of threats like data breach, 16% in Data Leakage and Loss, 15% in Insecure API's and 15% in Account Hijacking the during the cloud usage.

| Threat Faced                      | Count      |
|-----------------------------------|------------|
| Data Breach                       | 66         |
| Data Leakage and Loss             | 64         |
| Insecure API's                    | 63         |
| Account Hijacking                 | 61         |
| Daniel of Service                 | 27         |
| Abuse of Cloud Planning           | 23         |
| Inadequate Cloud Planning         | 19         |
| Cloud Related Malware             | 18         |
| Closer of Cloud Service           | 16         |
| Natural Disaster                  | 14         |
| Hardware Failure                  | 13         |
| Shared Technology Vulnerabilities | 11         |
| <b>Total</b>                      | <b>395</b> |

Table 3.29: Threat Faced during the use of cloud service

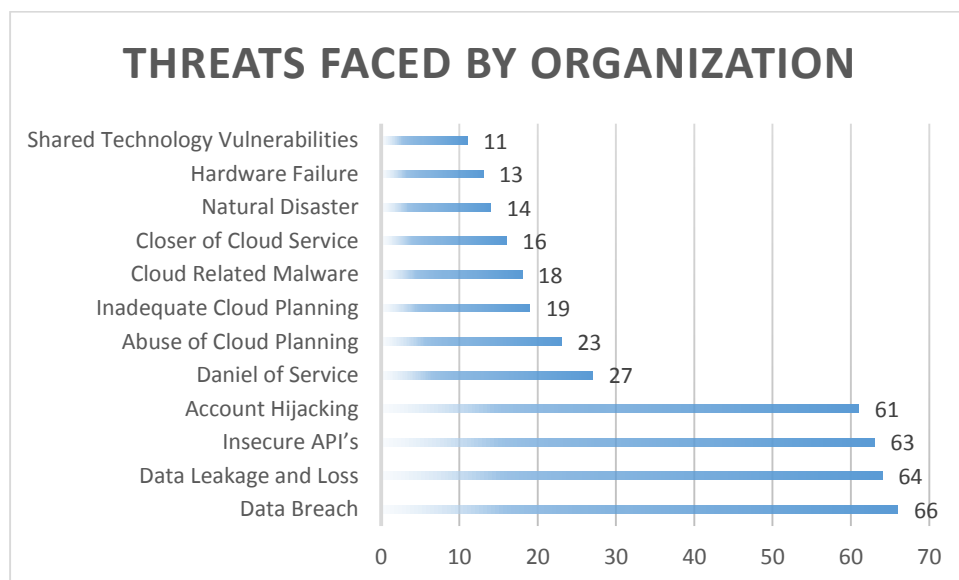


Figure 3.26: Threat Faced by Organization during cloud usage

#### i. Factors hindering during use of cloud services

29% of organizations struggle to utilize cloud services due to limited resources, while 26% face

difficulties due to inadequate training. Additionally, 22% believe that a lack of information contributes to the challenges of using cloud services.

| Factor Hindering       | Count      |
|------------------------|------------|
| Resource Crunch        | 113        |
| Insufficient Training  | 101        |
| Lack of Information    | 89         |
| New Automated Services | 74         |
| Other                  | 18         |
| <b>Total</b>           | <b>395</b> |

Table 3.30: Factors hindering during use of cloud services

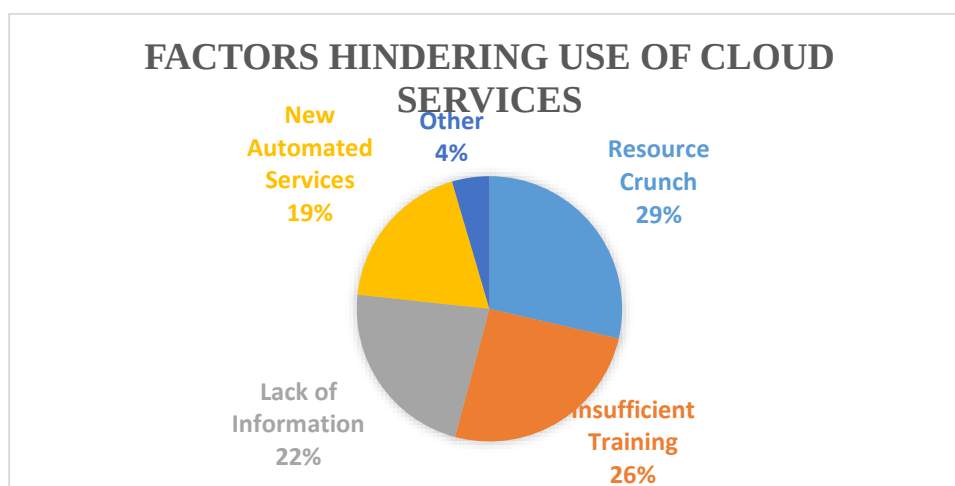


Figure 3.27: Factors hindering use of cloud services

### 3.6 Key Understandings

The key understandings that we have got after analysis of the collected data are as follows:

- 1) Organizations mostly utilize cloud technology for both storage and security purposes.
- 2) One technical purpose for using cloud services is to ensure data backup.
- 3) Organizations may be ignorant of the multitude of hazards they face while using cloud services, such as *account hijacking and the potential closure of the cloud service*.
- 4) Insufficient and infrequent training and information provided to employees results in their vulnerability to the utilization of modern cloud services.

- 5) Cloud service providers utilize sophisticated automated processes to manage cloud services, whereas companies struggle to handle them.
- 6) The threat detection team has a significantly high average response time, resulting in *the rapid spread of infections throughout the entire firm*.
- 7) Organizations have a relatively low to moderate level of worry regarding cloud services.
- 8) Most likely, organizations are utilizing a Public Cloud to access cloud services.
- 9) Centralized cloud security systems result in decreased performance.
- 10) The majority of Cloud Service Providers utilize the Advanced Encryption Standard (AES) for data encryption.

### **3.7 Experimental Identification of the Problem**

#### **3.7.1 Hypothesis for qualitative analysis**

Many experiments that are categorized in various forms to implement the proposed Framework are required.

1. Preliminary Settings
2. Implementation of existing Portfolio to initiate Experiments.
3. Implement Research Gaps
4. Implementation of the System Training
5. Testing the system
6. Results and Analysis
7. Facts and Findings.

#### **3.7.2 Hypothesis of Study**

The efficacy of cloud services in attaining utmost security, integrity, privacy, confidentiality, and authenticity within enterprises differs depending on aspects such as the organization's nature, geographical location, employee roles, technical prerequisites, and budget allocation.

The cloud storage system enables customers to conveniently access their data from any location and at any time, therefore facilitating a flexible and cost-effective on-demand and pay-per-use framework. Users can utilize the cloud storage model to rent and pay for storage and computing services according to their own needs. The cloud offers numerous advantages and capabilities to its customers, including cost-effective storage, adaptability, automatic updates, and disaster resilience. Nevertheless, safeguarding user privacy and guaranteeing data protection is crucial due to the potential risk of data leakage when users save their data in the cloud. Furthermore, when users outsource their data to the cloud, they relinquish control over it, and cloud data is susceptible to potential security breaches and numerous forms of assaults. The existing cloud storage systems possess additional drawbacks, such as the presence of centralized data storage, which undermines server security, and the reliance on trusted third parties impacts user privacy.

Moreover, the current distributed cloud storage system stores data in several data centers or servers in a distributed fashion, albeit it is not entirely distributed. Multiple data centers house data with high density. Consequently, a substantial quantity of data will be vulnerable if any of the servers or data centers are breached. Global public media outlets have extensively reported on security issues associated with cloud storage, including instances where users' confidential files have been unintentionally disclosed on iCloud (R. Lyengar et. al., 2020). Regrettably, there are now no feasible remedies available for ensuring the security of cloud systems.

Several hypothetical experiments, including the Complex Hypothesis, the Directional Hypothesis, and the Null Hypothesis, were conducted to detect and comprehend the underlying ideology driving the execution of different options and processes in real-time. The emphasis is on producing empirical outcomes. These studies aid in identifying deficiencies and constraints in research and in analyzing the exhibited outcomes.

### **3.7.3 Testing of Hypothesis**

#### **3.7.3.1 H01 – Case Study -1**

**Title:** Manage increasing volume of data in cloud storage system.

**Introduction:** In the present time, we are experiencing a significant growth in data, and there is a widespread need to store, protect, and transport this data. Due to the substantial volume of data, a significant storage capacity is necessary. Compressing data before storing might potentially decrease the required storage capacity.

**Objective:** The file with a smaller size can be efficiently transferred over the network in a shorter amount of time.

Implementation steps to mitigate the issue of increase volume of data.

Data compression aims to identify and remove repetitive patterns in data using effective approaches. This allows compressed data to conserve space for storage, reduce transfer time, and lower maintenance costs. The original file is condensed using coded notation, resulting in a compressed form that eliminates repetition. Due to its substantial size, the file necessitated a significant amount of storage capacity. Compressing a file before storing it might potentially lower its volume, therefore using less storage capacity. There are primarily two categories of data compression: Lossy and Lossless. The first form of data loss occurs at the output, which is considered acceptable. In the second type, no data is lost, and the original file can be fully recovered by decoding the encoded file. Lossless compression can be categorized into two coding techniques: entropy-based and dictionary-based. The algorithms that fall within the category of entropy-based coding are Huffman coding, Shannon-Fanon encoding, and Arithmetic coding. The entire LZ family falls under dictionary-based encoding.

Within our framework, we are implementing two algorithms that rely on entropy: Huffman coding and Shannon Fanon. Additionally, we are utilizing a dictionary-based technique called LZW.

The following information provides the details for Huffman code:

Huffman coding utilizes the frequency of each character in a file and the count of characters in a data structure with a frequency of 0. The Huffman encoding for a typical text file reduces the size of the original material by around 50%.

Huffman encoding trees yield the shortest possible character encodings employed in data compression. The nodes inside the tree symbolize the frequency at which a character appears. The root node corresponds to the string's length, and navigating the tree provides us with the encodings specific to each character. After the tree is built,

navigating through the tree provides us with the corresponding codes for each symbol. Lal and Rasheed (2020, pp. 590-597) conducted a study.

The image below displays the sample of the ideal tree once it has been completed.

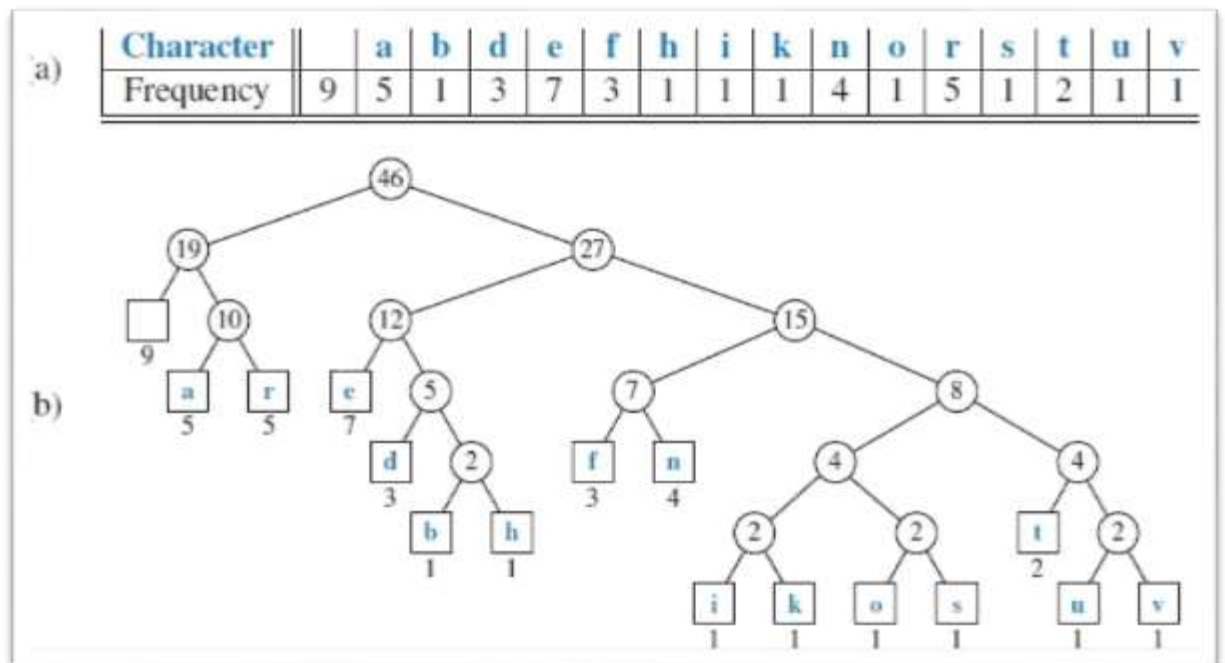


Figure 3.28: Representation of Huffman tree creation

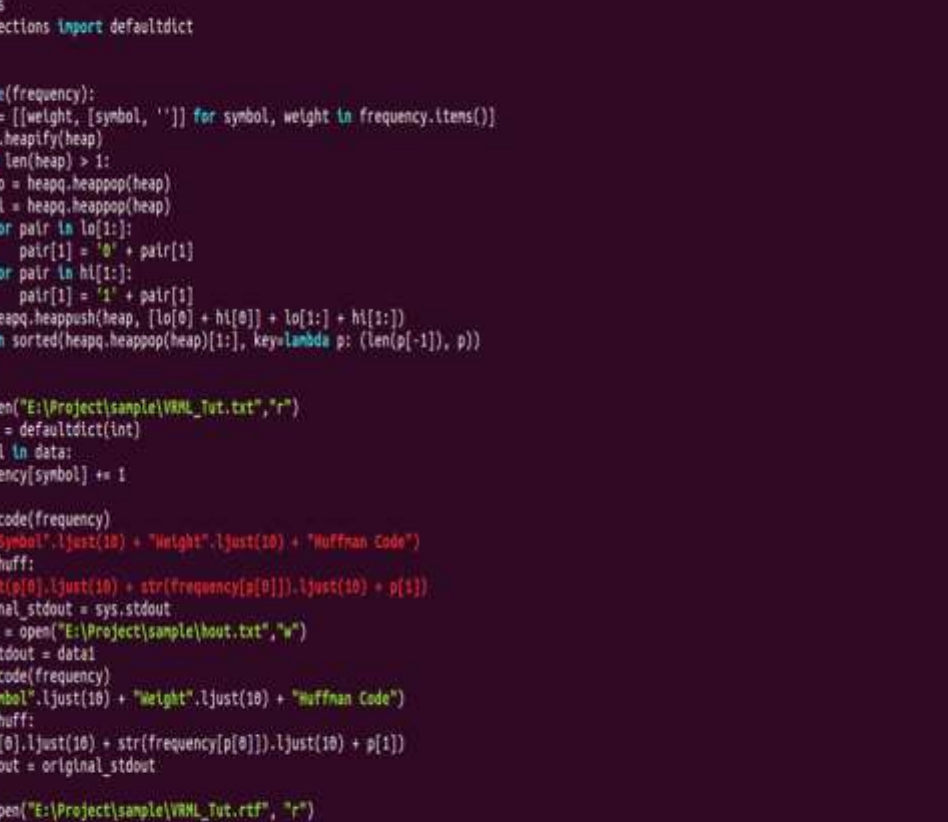
Developing the data compression program. The crucial elements in the execution were:

- Compiling a frequency dictionary.
- Choose two symbols with the lowest frequency and combine them in a repetitive manner: Utilized a minimum heap data structure.
- Construct a hierarchical structure representing the aforementioned process: Implemented a HeapNode class and utilized objects to uphold the hierarchical structure of the tree.
- Allocate code values to individual characters: Traversed the tree in a recursive manner and assigned the codes that corresponded to each element.
- Transform the supplied text into a coded form. Implemented padding for the encoded text if its length is not a multiple of 8. Store this padding information, consisting of 8 bits, at the start of the resulting code.

Save the outcome to an output binary file, which will serve as our compressed file.

- After processing many sample text files, the Compression Ratio was determined to be 2.1:1 on average.

Procedure: The initial stage in experimenting with compression techniques is applying a Huffman coding method to .txt and .doc files. Let's develop one using Python.



```

abc@ubuntu: ~/Project/Ultra
GNU nano 2.5.3                                File: Huffman.py

import heapq
import sys
from collections import defaultdict

def encode(frequency):
    heap = [[weight, [symbol, '']] for symbol, weight in frequency.items()]
    heapq.heapify(heap)
    while len(heap) > 1:
        lo = heapq.heappop(heap)
        hi = heapq.heappop(heap)
        for pair in lo[1:]:
            pair[1] = '0' + pair[1]
        for pair in hi[1:]:
            pair[1] = '1' + pair[1]
        heapq.heappush(heap, [lo[0] + hi[0]] + lo[1:] + hi[1:])
    return sorted(heapq.heappop(heap)[1:], key=lambda p: (len(p[-1]), p))

data = open("E:\\Project\\sample\\VRML_Tut.txt", "r")
frequency = defaultdict(int)
for symbol in data:
    frequency[symbol] += 1

huff = encode(frequency)
#print ("Symbol".ljust(10) + "Weight".ljust(10) + "Huffman Code")
for p in huff:
    # print(p[0].ljust(10) + str(frequency[p[0]]).ljust(10) + p[1])
    original_stdout = sys.stdout
    data1 = open("E:\\Project\\sample\\hout.txt", "w")
    sys.stdout = data1
    huff = encode(frequency)
    print("Symbol".ljust(10) + "Weight".ljust(10) + "Huffman Code")
    for p in huff:
        print(p[0].ljust(10) + str(frequency[p[0]]).ljust(10) + p[1])
    sys.stdout = original_stdout

data2 = open("E:\\Project\\sample\\VRML_Tut.tif", "r")
frequency = defaultdict(int)
for symbol in data2:

```

Figure 3.29: Applying Huffman coding compression algorithm

Let us run the software in VMware using the Ubuntu operating system and the Python programming language. The output of the application will be provided below.

This is the initial file with a size of 699 kilobytes.

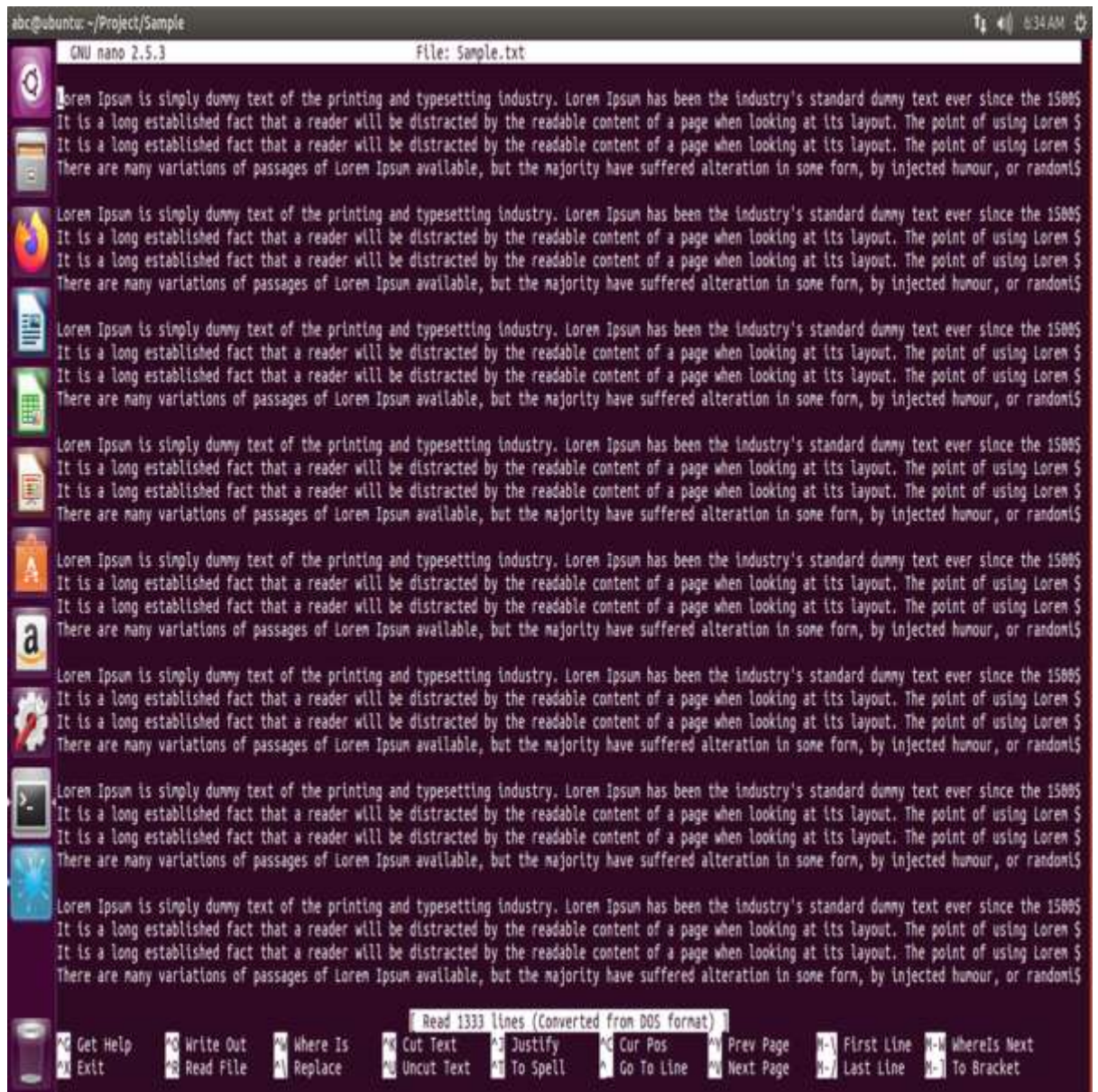


Figure 3.30: Original file before compression

It is readily apparent that there is a recurring word in the file. Upon compression, the resulting file will be encoded. We are now implementing the Huffman coding technique. The original file is reduced to a size of 384 KB.

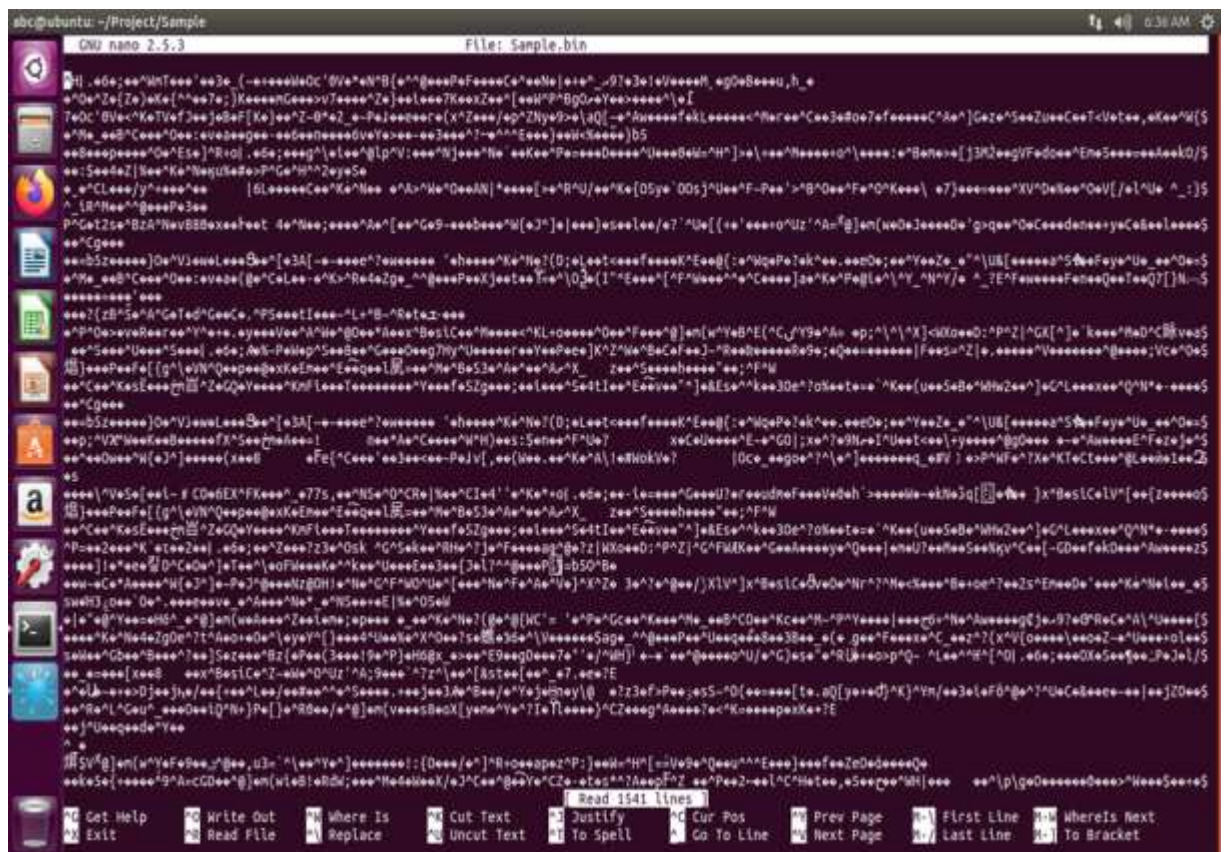


Figure 3.31: Compressed file after applying Huffman coding Compression algorithm

By utilizing the decompression process, we can restore the original file, which has a size of 699 KB.

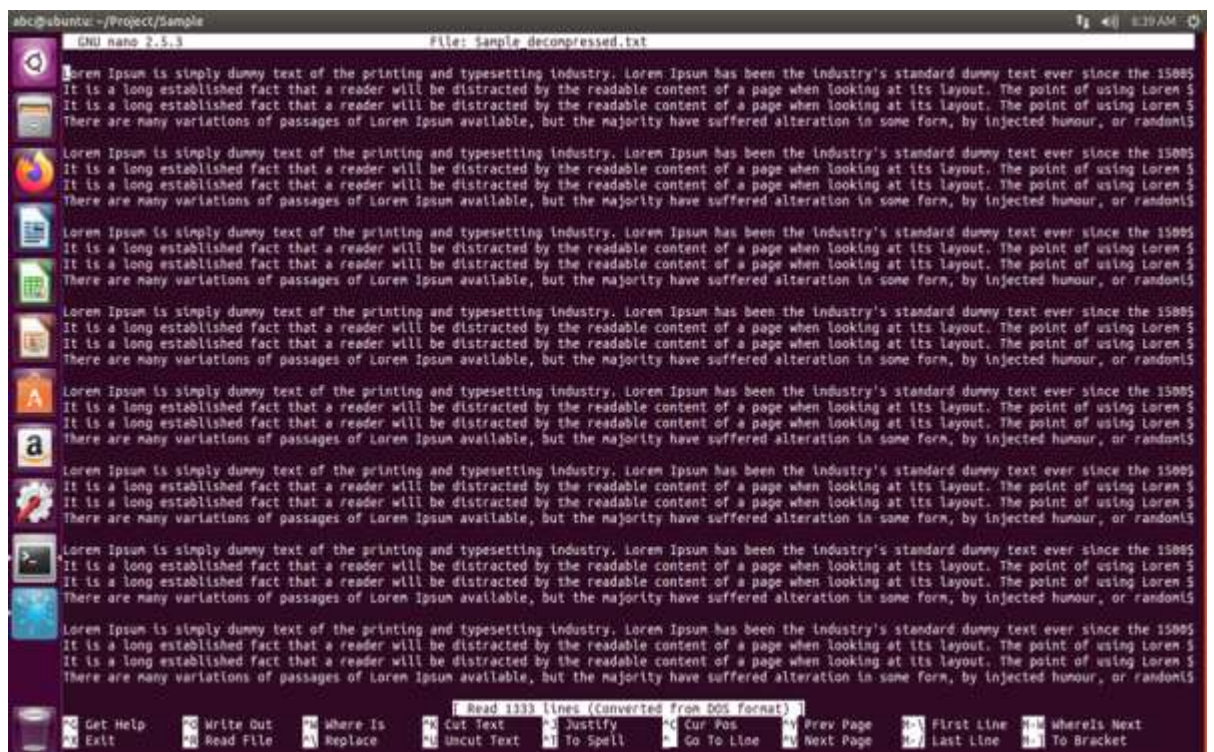
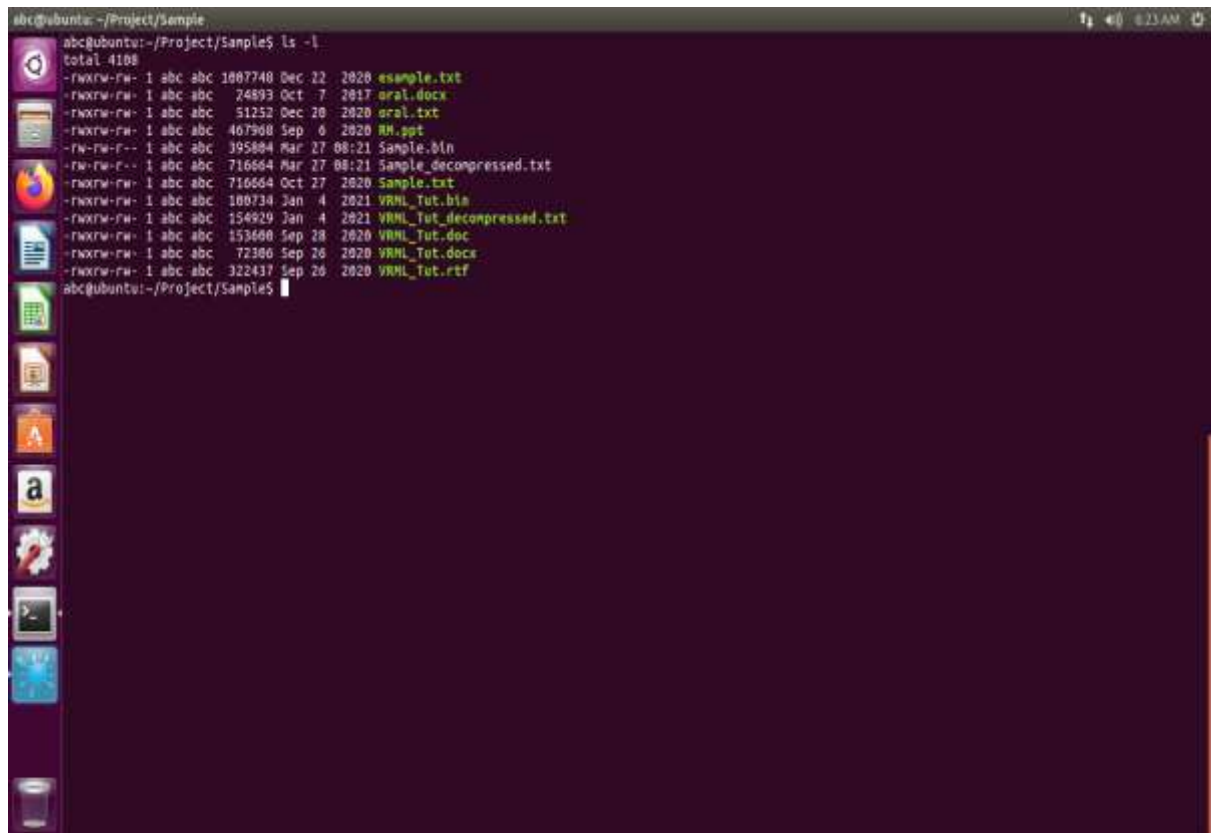


Figure 3.32: Decompressed file after applying Huffman coding decompression algorithm

A terminal window on an Ubuntu system showing the output of the 'ls -l' command in the directory '/Project/Sample'. The output lists several files with their permissions, owner, group, size, date, and name. The files include 'example.txt', 'oral.docx', 'oral.txt', 'RM.ppt', 'Sample.bin', 'Sample\_decompressed.txt', 'Sample.txt', 'VRML\_Tut.bin', 'VRML\_Tut\_decompressed.txt', 'VRML\_Tut.doc', 'VRML\_Tut.docx', and 'VRML\_Tut.rtf'. The sizes of the decompressed files are notably smaller than their compressed counterparts.

```
abc@ubuntu:~/Project/Sample$ ls -l
total 4188
-rwxrwx-rw- 1 abc abc 188740 Dec 22 2020 example.txt
-rwxrwx-rw- 1 abc abc 24893 Oct 7 2017 oral.docx
-rwxrwx-rw- 1 abc abc 51252 Dec 20 2020 oral.txt
-rwxrwx-rw- 1 abc abc 467968 Sep 6 2020 RM.ppt
-rw-rw-r-- 1 abc abc 395884 Mar 27 08:21 Sample.bin
-rw-rw-r-- 1 abc abc 716664 Mar 27 08:21 Sample_decompressed.txt
-rwxrwx-rw- 1 abc abc 716664 Oct 27 2020 Sample.txt
-rwxrwx-rw- 1 abc abc 188734 Jan 4 2021 VRML_Tut.bin
-rwxrwx-rw- 1 abc abc 154929 Jan 4 2021 VRML_Tut_decompressed.txt
-rwxrwx-rw- 1 abc abc 153688 Sep 28 2020 VRML_Tut.doc
-rwxrwx-rw- 1 abc abc 72386 Sep 28 2020 VRML_Tut.docx
-rwxrwx-rw- 1 abc abc 322437 Sep 28 2020 VRML_Tut.rtf
abc@ubuntu:~/Project/Sample$
```

Figure 3.33: Screen shot showing the file size

We utilized VMware Workstation 12 player alongside Ubuntu 16.04 LTS and python 2.7.12 to execute the compression technique.

There are two noteworthy points:

1. The result of compression will be a .bin file.
2. The aforementioned software necessitates the execution of the decompression function using the identical object that generated the compressed file.

Recently, we have discovered that the use of Huffman coding, a compression algorithm, can reduce the size of a file by up to 50% when there is repeated material.

**Observation & Results:** The experiment revealed that files of any kind may include duplicated data, which can be eliminated using a compression algorithm. The compression techniques have the capability to decrease the size of a file by up to 50%.

The decrease in file size is contingent upon the proportion of duplicated data within the file.

### 3.7.3.2 H02 – Case Study -2

**Title:** Implementation of cryptographic algorithms to address data breach concerns.

**Introduction:** A data breach can occur due to human mistake, security vulnerabilities, or intentional actions, resulting in the unauthorized disclosure of protected data to the public by hostile individuals. The extent of damage caused to consumers depends on the sensitivity of the exposed data. The data breach issue originates from the inadequate implementation of encryption mechanisms by the cloud service provider. It leads to the unauthorized acquisition of users' confidential data.

**Objective:** The goal is to establish measures to ensure the security of data transmission.

**Explanation:** After applying the Huffman coding technique to compress the file, we will enhance its security by encrypting it using the RSA encryption algorithm. This will provide an additional layer of protection for data that are transferred over a network. For this experiment, we will utilize an RSA encryption technique. RSA is an asymmetric cryptographic technique, characterized by the presence of two distinct keys: the public key and the private key. This cryptographic method is commonly referred to as public-key cryptography, as it allows for the distribution of one of the keys to a user.

The mechanism underlying the RSA algorithm is explained by Krishna and Manikandan (2020, pp. 321-327).

To create a Public Key for the purpose of encryption:

1. Select two prime numbers, such example 7 and 11.
2. Compute the multiplication of the prime numbers and store the result in a variable.

$$n = 7 * 11 = 77$$

3. Assume a small exponent  $e$  which will lie between 1 to  $\phi(n)$ . Let us assume  $e=3$

Now, we are ready with our public key ( $n = 77$  and  $e = 3$ ).

Encryption:

$$m^e \bmod(n) = 89^3 \bmod 77 = 166 = c$$

The RSA algorithm provides:

- **Key-pair generation:** generate a random private key and public key (the size is 1024-4096 bits).
- **Encryption:** It encrypts a secret message (integer in the range  $[0 \dots \text{key\_length}]$ ) using the public key and decrypts it back using the secret key.
- **Digital signatures:** sign messages (using the private key) and verify message signature (using the public key).
- **Key exchange:** It securely transports a secret key used for encrypted communication.

**Procedure:** After using the RSA encryption technique, it is necessary to choose a bit size ranging from 1024 to 4096 in order to generate random public and private keys. Subsequently, the file is encrypted using the public key and subsequently decrypted using the secret key.

1. It is necessary to import it from the `Crypto.PublicKey` module.
2. Additionally, we are utilizing the OAEP-Padding technique. We have imported the `PKCS1_OAEP` module from the `Crypto.cipher` library.
3. In order to convert binary to ASCII, we have loaded the `binascii` module.
4. Creating key pairs by utilizing the `RSA.generate` function.
5. Computing the public key value.
6. Inputting a message that requires encryption.
7. Employing the `encrypt` function to encode the message.

First Generate a RSA private / public key pair.

```

abc@ubuntu: ~/Project/Ultra
GNU nano 2.5.3      File: ch9_generate_key.py

ch9_generate_key.py
from Crypto.PublicKey import RSA

#Generate a public/ private key pair using 4096 bits key length (512 bytes)
new_key = RSA.generate(4096, e=65537)

#The private key in PEM format
private_key = new_key.exportKey("PEM")

#The public key in PEM format
public_key = new_key.publickey().exportKey("PEM")

print private_key
fd = open("private_key.pem", "wb")
fd.write(private_key)
fd.close()

print public_key
fd = open("public_key.pem", "wb")
fd.write(public_key)
fd.close()

```

Figure 3.34: Code for RSA key generation.

The provided code will produce the public key and private key as shown below.

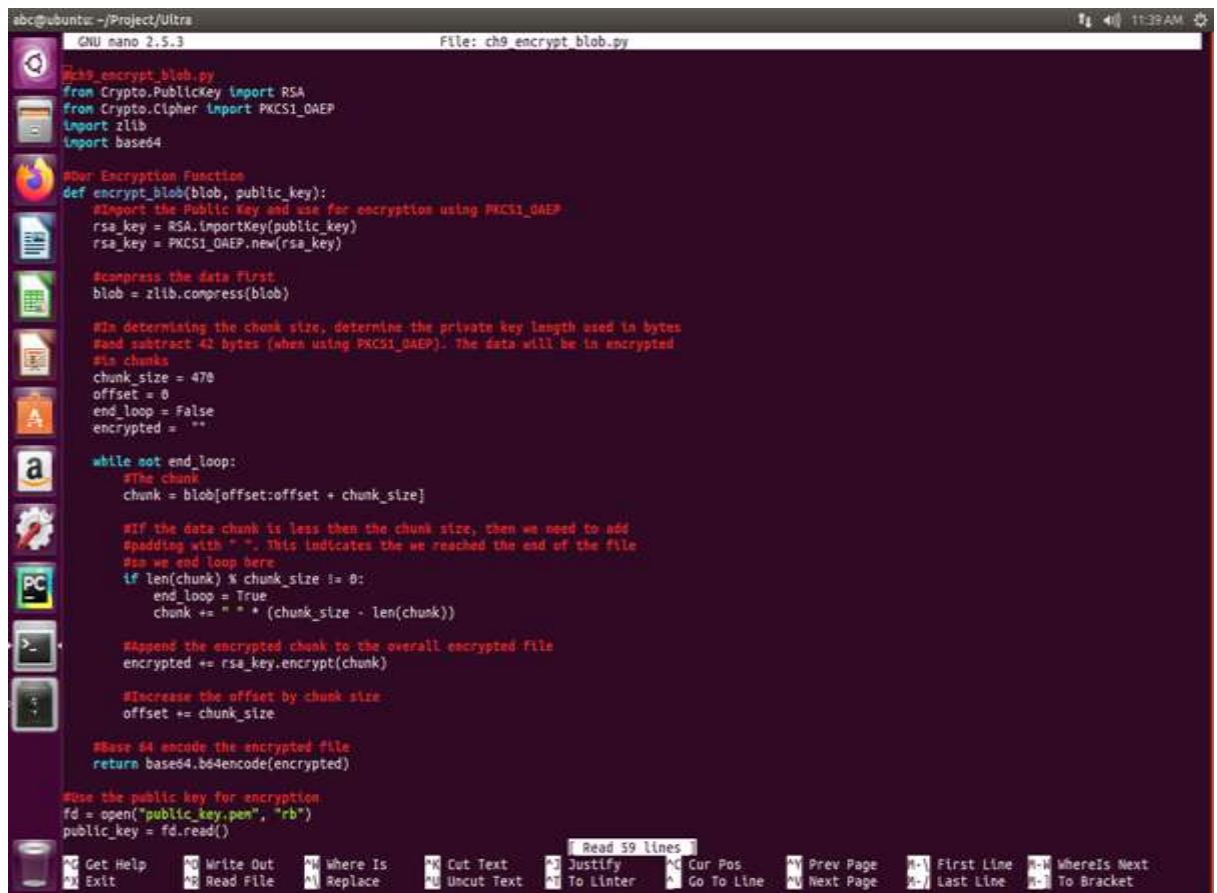
```

abc@ubuntu: ~/Project/Ultra
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAzG8K3ZrDC+P37YcGeDEHwZuPqrVqbQztUsNo91xIb5
nMj5MObPeYq8L5y2xe9Is/Abexw+eUFLJ88dLupf7GLo5UqzcG19J1tT2CAJ3rWA
+hmg07GHLrIe749UgC8vHC59/OvndtPK3p2FuWH/6fHEHEp9ySneLDJ0MKG8e98
UF0VbMdxHMR9ezhEJF+Lryng7rK6P3b37bpsJ5gT3Wx/rvkuU2v3L5T3Evt1WVC
KXGz9FLGHUwJbQPD/16o1OJz+ZcczNno6ScUF4ebFfGFWJ2C5b4a3BuxrNV+vaf
zfdZ8ELx3PXWmgMen46RsaySxLIPxyeYjWx7xLUZ4TLD1GedMITxNM+63a/L2
oN1+sy2ykJS6l1n0Bty+S9vxs8ZARKzKJBCnLxAKT102pntE58GJnTiyKtTL9OL
n0PK4Byahawy/c10EIAEdrZe2eAp1ng3ErNS33dORF9a1MAWQKCAQEAuxP5CLcc
LV4ev11ELH4nbw5NB/8veZAFgBUgFaB+yHCkdFyys1LaqTBAF04Q6/p90+s+088e
FqKX5T0AJr703my6TcsTnuntg1mG0HhEXfnepv2I4N1Dq6E+Fwc1J0wPkgYD/9e
1r7R7EAM34ABEC2+nTvr1q17KZP99CS/xFLNwocPCZKCLCqVvq+L32CKTlyw4L2
gUpPKxvtqLmwtdp8eEonLS7MA1959b008DxaJ08Z1r1eUbj/sMnyonc210Ire
J1Am4cht8g91m3r0BPHWtY1AF0wH0K1pSf20112Hx1B8SePukLEF90qNI
r41q9PpExGQXQKCAQEAyVNERyovFN0wDKWtI3of+J03+FFGUSRenp6Hn+2TW5n
YSyPH0+M09FA13R9dknarsaTagU1lb/8zXHCjX4MHq2UT45dg22ZqIehUMWXTYrb
82VwSU0VLCokbLRTVRfLmN79oe2Pyh3QsxnG1uK809W0q3319U1EB9hepUdLCLA
oEB8RzEQBUr6zc8rEN25khwK01q1t0c67qEXemqkpvvAxy/FA4Jf7PA40CnnsR
Nqf72M0wsU8qVMHVTY6LttXhNTShAG2C80ka450g6mw4K1MvHob1xEx2LIuL6M4
HR0aG2x7J9Kk1ky9A70gJ5a45LHCx/sBP8RhyesGwKCAQAw2Au8MWTIo5AK+31
NtFL3fGn06GK1R3/r1AZAZPD0gwbzj+1dLvUdAKSVYzab3C8WJugstL28q5UuQ7v
gSKKUFvQGLfGT9zL685Xwh2AZdd6D/Sb0wjxlpKw0GwR7xy0TV1L/zGeeCHRYKDS
Uonxe3t9Rw1MtnD93NU0Drq426W15fZRF133kknZ08S8Dlj+KJAKaJjE3pHP/o
Ivq9T3IqrBt9kyDU/r1PUSH2uAP7FRKEG2DMcAtk/Sy2KF51P+ErXKG838PW3qAX
VZUtntXItGagRPdPj+KrPMT2x2FRXQgEFe0LQ9Jh0R5tIZUB8Z8B0XFCWJwQAO
wsv9A01BAEBXhpeUptg2H5CZVn7nXK1B7b2r2EwcuL1JzC7Tfmgp9QmHfYK
Sg4F0uavN3DqysBFKx27z3U1SvelG6053C8Gf+ab590u1c8FQEW45j/TpWNETCk
8qf580ukwplCvYj1H01b11Dq5W7HugKxbpUT38ktw3ub1bpYK5Y3k+2fLWeg
3+abTSg1LwR0uALX1tL0t0p0NZVM1mbs2Wu9jXaxxLPuP2nnV13shpw7N1L3
Kk/jhKGNrtn0uxZAB6L2PAubHMQ5yXI280yvtqJ31Waukqx7FnxAQ11yt46VP
g7WvgtR9XqWpXAGK71ndr51n+YUCpgEAYg5/1XE3oLn2DdhfEnb89XzuL22
n19gsap3ahVpQHAN1zUe664YUHu+tfHK/cD+rfE+Vp7D08C28a1fgn0VJ1VUMd
QKNZwpv6a6jeCL6Kv73J1E2E/9M0bvy6w7df32+36e9QHHSxrtpg29/QC4t11JE
Bs+TFxVKZV72X0ILHK1MYU1vyd5Mk4t9W1Q1KRj+ZySe/Ss07Fc0LS5A2vg0tp
uEseyT0GBjn8ZLn4nb05Jan8R93sQu+cLoqlzj5y1LSUSjG94sxo7p3DVpuT48nJ
kdwEUGvGecGD/9pb12T0cPuqa+sGlocJG28r3r55ZFsFwyosJ86w2bo5GQ==
-----END RSA PRIVATE KEY-----
-----BEGIN PUBLIC KEY-----
MIIEPjBGAEECAQEAzG8K3ZrDC+P37YcGeDEHwZuPqrVqbQztUsNo91xIb5
nMj5MObPeYq8L5y2xe9Is/Abexw+eUFLJ88dLupf7GLo5UqzcG19J1tT2CAJ3rWA
+hmg07GHLrIe749UgC8vHC59/OvndtPK3p2FuWH/6fHEHEp9ySneLDJ0MKG8e98
UF0VbMdxHMR9ezhEJF+Lryng7rK6P3b37bpsJ5gT3Wx/rvkuU2v3L5T3Evt1WVC
KXGz9FLGHUwJbQPD/16o1OJz+ZcczNno6ScUF4ebFfGFWJ2C5b4a3BuxrNV+vaf
zfdZ8ELx3PXWmgMen46RsaySxLIPxyeYjWx7xLUZ4TLD1GedMITxNM+63a/L2
oN1+sy2ykJS6l1n0Bty+S9vxs8ZARKzKJBCnLxAKT102pntE58GJnTiyKtTL9OL
n0PK4Byahawy/c10EIAEdrZe2eAp1ng3ErNS33dORF9a1MAWQKCAQEAuxP5CLcc
LV4ev11ELH4nbw5NB/8veZAFgBUgFaB+yHCkdFyys1LaqTBAF04Q6/p90+s+088e
FqKX5T0AJr703my6TcsTnuntg1mG0HhEXfnepv2I4N1Dq6E+Fwc1J0wPkgYD/9e
1r7R7EAM34ABEC2+nTvr1q17KZP99CS/xFLNwocPCZKCLCqVvq+L32CKTlyw4L2
gUpPKxvtqLmwtdp8eEonLS7MA1959b008DxaJ08Z1r1eUbj/sMnyonc210Ire
J1Am4cht8g91m3r0BPHWtY1AF0wH0K1pSf20112Hx1B8SePukLEF90qNI
r41q9PpExGQXQKCAQEAyVNERyovFN0wDKWtI3of+J03+FFGUSRenp6Hn+2TW5n
YSyPH0+M09FA13R9dknarsaTagU1lb/8zXHCjX4MHq2UT45dg22ZqIehUMWXTYrb
82VwSU0VLCokbLRTVRfLmN79oe2Pyh3QsxnG1uK809W0q3319U1EB9hepUdLCLA
oEB8RzEQBUr6zc8rEN25khwK01q1t0c67qEXemqkpvvAxy/FA4Jf7PA40CnnsR
Nqf72M0wsU8qVMHVTY6LttXhNTShAG2C80ka450g6mw4K1MvHob1xEx2LIuL6M4
HR0aG2x7J9Kk1ky9A70gJ5a45LHCx/sBP8RhyesGwKCAQAw2Au8MWTIo5AK+31
NtFL3fGn06GK1R3/r1AZAZPD0gwbzj+1dLvUdAKSVYzab3C8WJugstL28q5UuQ7v
gSKKUFvQGLfGT9zL685Xwh2AZdd6D/Sb0wjxlpKw0GwR7xy0TV1L/zGeeCHRYKDS
Uonxe3t9Rw1MtnD93NU0Drq426W15fZRF133kknZ08S8Dlj+KJAKaJjE3pHP/o
Ivq9T3IqrBt9kyDU/r1PUSH2uAP7FRKEG2DMcAtk/Sy2KF51P+ErXKG838PW3qAX
VZUtntXItGagRPdPj+KrPMT2x2FRXQgEFe0LQ9Jh0R5tIZUB8Z8B0XFCWJwQAO
wsv9A01BAEBXhpeUptg2H5CZVn7nXK1B7b2r2EwcuL1JzC7Tfmgp9QmHfYK
Sg4F0uavN3DqysBFKx27z3U1SvelG6053C8Gf+ab590u1c8FQEW45j/TpWNETCk
8qf580ukwplCvYj1H01b11Dq5W7HugKxbpUT38ktw3ub1bpYK5Y3k+2fLWeg
3+abTSg1LwR0uALX1tL0t0p0NZVM1mbs2Wu9jXaxxLPuP2nnV13shpw7N1L3
Kk/jhKGNrtn0uxZAB6L2PAubHMQ5yXI280yvtqJ31Waukqx7FnxAQ11yt46VP
g7WvgtR9XqWpXAGK71ndr51n+YUCpgEAYg5/1XE3oLn2DdhfEnb89XzuL22
n19gsap3ahVpQHAN1zUe664YUHu+tfHK/cD+rfE+Vp7D08C28a1fgn0VJ1VUMd
QKNZwpv6a6jeCL6Kv73J1E2E/9M0bvy6w7df32+36e9QHHSxrtpg29/QC4t11JE
Bs+TFxVKZV72X0ILHK1MYU1vyd5Mk4t9W1Q1KRj+ZySe/Ss07Fc0LS5A2vg0tp
uEseyT0GBjn8ZLn4nb05Jan8R93sQu+cLoqlzj5y1LSUSjG94sxo7p3DVpuT48nJ
kdwEUGvGecGD/9pb12T0cPuqa+sGlocJG28r3r55ZFsFwyosJ86w2bo5GQ==
-----END PUBLIC KEY-----
abc@ubuntu: ~/Project/Ultra$

```

Figure 3.35: Generated public and private keys

Subsequently, these keys will be utilized for the purpose of encrypting and decrypting the file. Below is the code for encrypting the file. We are providing the previously compressed file as input to this code.



```
abc@ubuntu: ~/Project/Ultra
GNU nano 2.5.3 File: ch9_encrypt_blob.py

#ch9_encrypt_blob.py
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import zlib
import base64

#Our Encryption Function
def encrypt_blob(blob, public_key):
    #Import the Public Key and use for encryption using PKCS1_OAEP
    rsa_key = RSA.importKey(public_key)
    rsa_key = PKCS1_OAEP.new(rsa_key)

    #Compress the data first
    blob = zlib.compress(blob)

    #In determining the chunk size, determine the private key length used in bytes
    #and subtract 42 bytes (when using PKCS1_OAEP). The data will be in encrypted
    #in chunks
    chunk_size = 470
    offset = 0
    end_loop = False
    encrypted = ""

    while not end_loop:
        #The chunk
        chunk = blob[offset:offset + chunk_size]

        #If the data chunk is less than the chunk size, then we need to add
        #padding with " ". This indicates the we reached the end of the file
        #so we end loop here
        if len(chunk) % chunk_size != 0:
            end_loop = True
            chunk += " " * (chunk_size - len(chunk))

        #Append the encrypted chunk to the overall encrypted file
        encrypted += rsa_key.encrypt(chunk)

        #Increase the offset by chunk size
        offset += chunk_size

    #Base 64 encode the encrypted file
    return base64.b64encode(encrypted)

#Use the public key for encryption
fd = open("public_key.pem", "rb")
public_key = fd.read()
```

Figure 3.36: code for file encryption using RSA

The previously compressed file sample.bin has been given as input to the above code and it will result in encrypted\_sample.bin

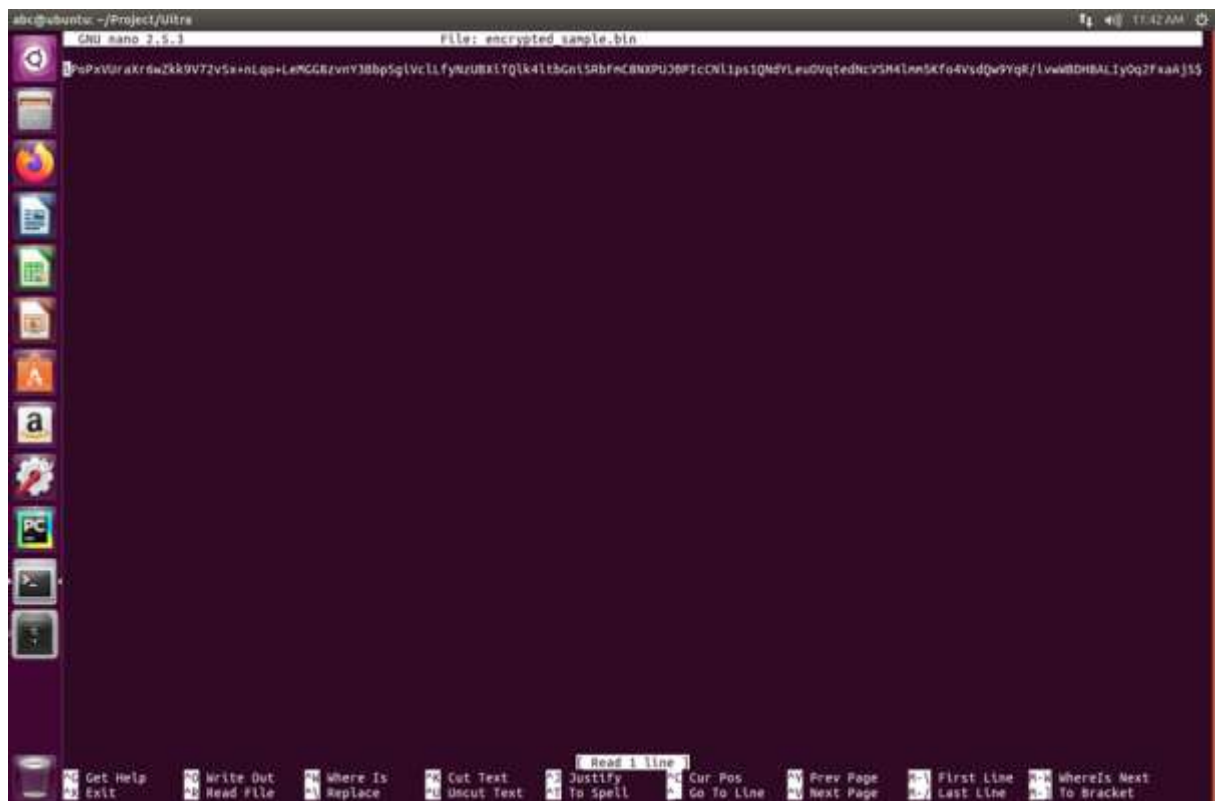


Figure: 3.37: Encrypted compressed file.

The code for decrypting an encrypted file is as shown below:

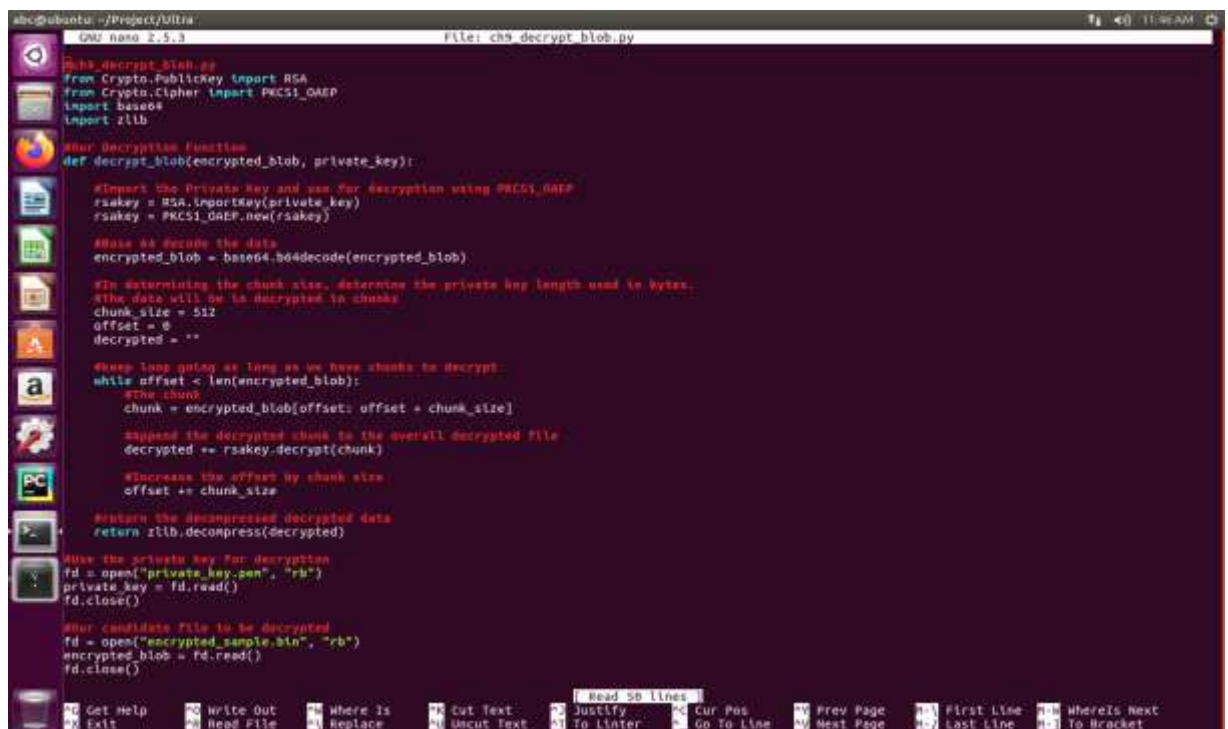


Figure 3.38: Code for file decryption using RSA

The encrypted\_sample.bin file can be decrypted as decrypted\_sample.bin.

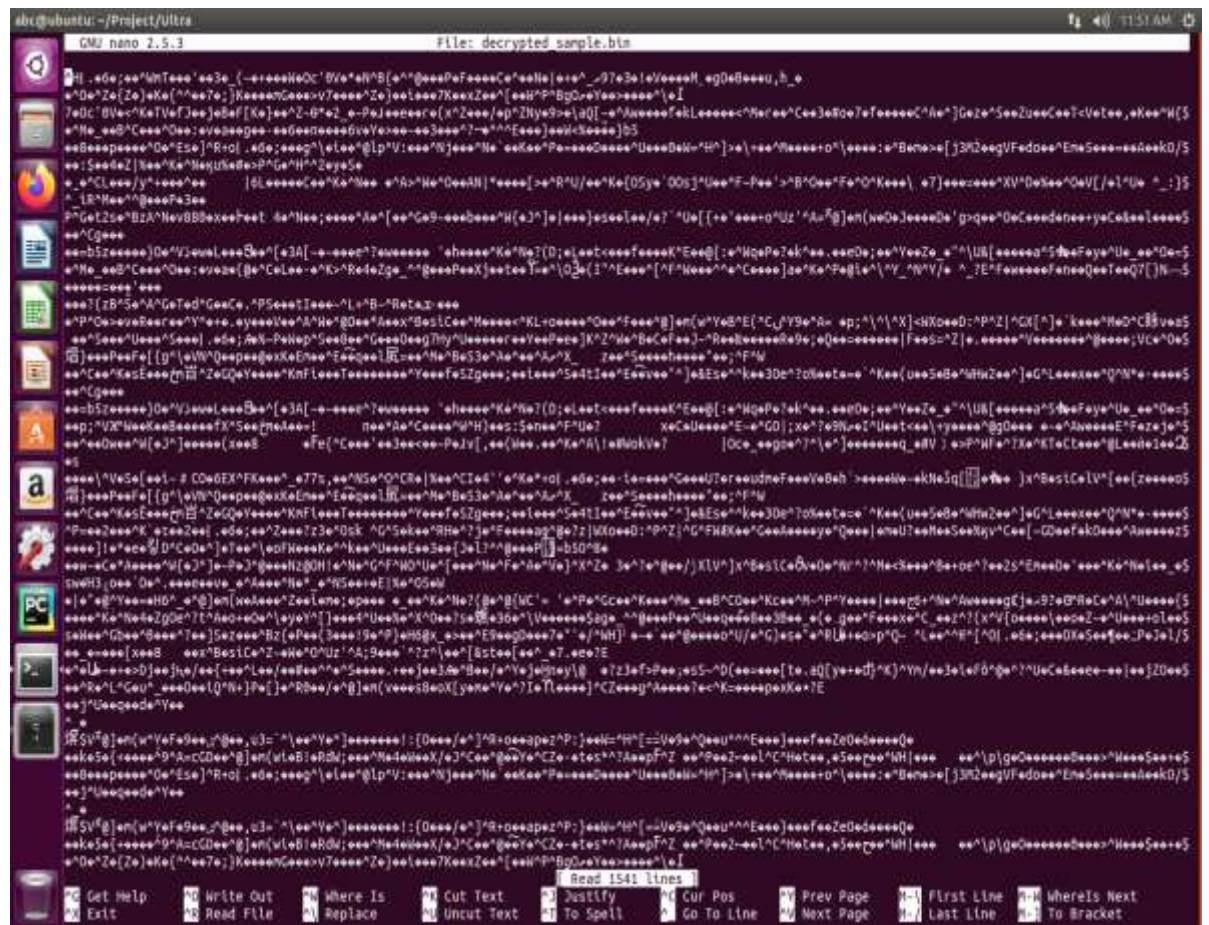


Figure 3.39: decrypted file using RSA

**Observations and Result:** Encrypting a compressed file enhances its security. Therefore, it serves as a potent instrument employed to safeguard files stored on cloud storage. We employ RSA, which is considered the most secure kind of encryption. Cracking RSA encryption is a challenging task.

- A brute force assault is not feasible due to the large number of potential keys that need to be tested. Furthermore, this process requires a significant amount of time.
- The RSA algorithm is not vulnerable to dictionary attacks since its keys include solely of numeric values and do not include any characters.
- Analyzing the frequency of characters is challenging due to the fact that a single encrypted block can represent many characters.
- There are no explicit mathematical techniques to exploit the RSA cryptography.

### 3.7.3.3 H03 – Case Study -3

**Title:** The blockchain technology ensures a transparent and tamper-proof framework for storing the metadata of both data and users.

**Introduction:** Blockchain is a distributed database that consists of a series of interconnected blocks. It is very resistant to tampering, forgery, and traceability. The blockchain securely records all transaction information, and almost no one has the ability to modify the data after it has been recorded. Consequently, blockchain technology exhibits a higher level of simplicity and stability compared to alternative security systems. The integration of blockchain technology into cloud storage architecture offers enhanced security for consumers. Instead of centralizing resources in a singular data center or server, a blockchain network disperses them over multiple nodes.

**Objective:** Blockchain-based cloud storage is to remove the need on a central authority for data maintenance, thereby mitigating the dangers associated with large-scale data breaches.

**Explanation:** This code will divide the encrypted file into segments of 512 KB and then apply a hash function to each segment of the file. The entirety of the file, after having a hash function applied to it, is included as a single unit within a blockchain block. This blockchain will function as an index for files on the web server. As a result, the file is protected from external threats and gains the advantages of cloud computing. Multiple users can utilize this index to upload or download their files on the web server in order to gain the benefit of enhanced security.

**Procedure:** In this experiment, we will utilize an encrypted file named `encrypted_sample.bin`. We are dividing this encrypted file into segments of 512 kilobytes. We are applying the SHA256 hash function to each portion of the file. The code for dividing the file is:

```

#!/usr/bin/python
#####
# split a file into a set of portions; join.py puts them
# back together; this is a customizable version of the
# standard unix split command-line utility; because it
# is written in Python, it also works on Windows and can
# be easily tweaked; because it exports a function, it
# can also be imported and reused in other applications;
#####

import sys, os

kilobytes = 1024
megabytes = kilobytes * 1024
chunksize = 512 * 1024 # default: roughly a floppy

def split(filename, todir, chunksize=chunksize):
    # caller handles errors
    # make dir, read/write parts
    # delete any existing files
    if not os.path.exists(todir):
        os.mkdir(todir)
    else:
        for fname in os.listdir(todir):
            os.remove(os.path.join(todir, fname))
    partnum = 0
    input = open(filename, 'rb') # use binary mode on Windows
    # not-empty string from read
    # get next part <= chunksize
    while 1:
        chunk = input.read(chunksize)
        if not chunk: break
        partnum = partnum + 1
        filename = os.path.join(todir, ('part%04d' % partnum))
        fileobj = open(filename, 'wb')
        fileobj.write(chunk)
        fileobj.close() # or simply open().write()
        input.close()
        assert partnum <= 9999 # join sort fails if 5 digits
    return partnum

if __name__ == '__main__':
    if len(sys.argv) == 2 and sys.argv[1] == '-help':
        print 'Use: split.py [file-to-split target-dir [chunksize]]'
    else:
        if len(sys.argv) < 3:
            interactive = 1
            filename = raw_input('File to be split? ') # input if clicked
            todir = raw_input('Directory to store part files? ')
        else:
            interactive = 0
            filename, todir = sys.argv[1:3] # args to cmdline

```

Figure 3.40: Code for splitting a file

The function has taken the previously encrypted file named encrypted\_sample.bin as input and the output will be a split file named part0001.

```

abc@ubuntu:~/Project/Ultra$ cd
abc@ubuntu:~$ cd Project
abc@ubuntu:~/Project$ cd Sample
abc@ubuntu:~/Project/Sample$ cd encrypted
abc@ubuntu:~/Project/Sample/encrypted$ ls -l
total 20
-rw-rw-r-- 1 abc abc 17068 Sep 22 10:34 part0001
abc@ubuntu:~/Project/Sample/encrypted$

```

Code for applying SHA256 hash on splited file



```
abc@ubuntu: ~/Project/Ultra
GNU nano 2.5.3 File: sha256.py

import hashlib

filename = input("Enter the input file name: ")
sha512_hash = hashlib.sha512()
with open(filename,"rb") as f:
    # Read and update hash string value in blocks of 4K
    for byte_block in iter(lambda: f.read(4096),b''):
        sha512_hash.update(byte_block)
    print(sha512_hash.hexdigest())

Get Help Exit Write Out Read File Where Is Replace Cut Text Uncut Text Read 9 lines Justify To Linter Cur Pos Go To Line Prev Page Next Page First Line Last Line WhereIs Next To Bracket
```

Figure 3.41: Code for applying SHA256 hash on splitted file

The splitted file part0001 has been given as input to above code and it will result in hash code of piece of file.

```
abc@ubuntu:~/Project/Ultra$ python sha256.py
```

```
Enter the input file name: "/home/abc/Project/Sample/encrypted/part0001"
```

```
63758e5ff7bd467b1793631643d5e1e30c2c683740aeb77c3ec44741bf0c472e15f8088
```

```
83c0bcf30717480d8d5f07dd3771fa69df77deb6b4fad672c5e33bb8e
```

```
abc@ubuntu:~/Project/Ultra$
```

Subsequently, the remaining information pertaining to the file, including as the file name, file size, file type, piece counter, hash of file pieces, and server URL, is appended to the blockchain.

Code for generating a blockchain and uploading it to a server.

```

abc@ubuntu: ~/Project/Ultra
GNU nano 2.5.3 File: BC.py
encoding: utf-8
# 30[39]:
from hashlib import sha256
import json
import time

from flask import Flask, request
import requests

class Block:
    def __init__(self, index, transactions, timestamp, previous_hash):
        self.index = index
        self.transactions = transactions
        self.timestamp = timestamp
        self.previous_hash = previous_hash
        self.nonce = 0

    def compute_hash(self):
        """
        A function that return the hash of the block contents.
        """
        block_string = json.dumps(self.__dict__, sort_keys=True)
        return sha256(block_string.encode()).hexdigest()

class Blockchain:
    # difficulty of our pow algorithm
    difficulty = 2

    def __init__(self):
        self.unconfirmed_transactions = []
        self.chain = []
        self.create_genesis_block()

    def create_genesis_block(self):
        """
        """

```

Figure 3.42: Code for creating a block chain and uploading it on server

The above code will give the output as follows:

1. Starting the server

```
abc@ubuntu:~/Project/Ultra$ sudo nano BC.py
```

```
abc@ubuntu:~/Project/Ultra$ python BC.py
```

\* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

\* Restarting with stat

```
127.0.0.1 - - [20/Nov/2021 04:37:26] "GET /chain HTTP/1.1" 200 -
```

2. The Blockchain

```
abc@ubuntu:~$ curl http://127.0.0.1:5000/chain
```

```
{
  "length": 2,
  "chain": [
    {
      "nonce": 0,
      "index": 0,
      "hash": "3ad3ba5f08e0f8d15c81c0ae54bf6e9e9136d3a81d2303649beef37b9e57a3b9",
      "transactions": [],
      "timestamp": 1637433074.778131,
      "previous_hash": "0"
    },
    {
      "nonce": 15,
      "index": 1,
      "hash": "000f3fb6cadacba296685ec6cf0158eef7c8b1b4c9269e65f5615dd7468cc59c"
    }
  ]
}
```

```
"transactions": [{ "url": "http://localhost:8080/gui/web/index.html", "filetype": "text",  
"pic": 1, "filename": "Sample.txt", "filesize": 716664, "phash":  
"63758e5ff7bd467b1793631643d5e1e30c2c683740aeb77c3ec44741bf0c472e15f8088  
83c0bcf30717480d8d5f07dd3771fa69df77deb6b4fad672c5e33bb8e"}], "timestamp":  
1637433074.778207, "previous_hash":  
"3ad3ba5f08e0f8d15c81c0ae54bf6e9e9136d3a81d2303649beef37b9e57a3b9"}]}
```

**Observation & Results:** The experiment demonstrated that by utilizing a dynamically constructed blockchain, it is feasible to effectively safeguard any file using this code. The blockchain generated in this project functions as an index for files, enabling their upload and download on a cloud platform while providing comprehensive security privileges. Subsequently, several files might be uploaded or downloaded in the cloud environment. This code will divide the encrypted file into segments of 512 KB and apply a hash function to each segment of the file. The entirety of the file, including the hash, is incorporated into a singular block on the blockchain. This blockchain will function as an index for files uploaded on the web server. As a result, the file is protected from external threats and gains the advantages of cloud computing.

### 3.7.4 Key Findings of the Hypothesis Experiments

When utilizing cloud storage, the size of the data is not the sole concern. Furthermore, it is plagued by numerous technical challenges such as security vulnerabilities, data integrity concerns, access policy limitations, and key management difficulties. The first hypothesis utilizes a compression approach to effectively control the size of data. The text accurately explains the process of applying compression techniques while storing data. However, many cloud storage systems store data in multiple data centers to accommodate the large quantity of the data. A limited number of studies have addressed the security concerns related to user data, distributed key management, access policies, and integrity issues in cloud storage systems. After conducting both quantitative and qualitative trials, the research team decided to adopt the MIX research approach for this study.

### **3.8 Proposed Framework**

The proposed framework utilizes the CP-ABE algorithm to guarantee the properties of confidentiality, integrity, and availability. Furthermore, it offers a comprehensive decentralized framework for storing crucial information, delegating relevant details, and verifying the integrity of data within the blockchain architecture. The experimental results, analysis, and performance evaluation demonstrate that the proposed framework offers a viable and dependable cloud environment.

#### **3.8.1 Objectives of the Study**

The study aims to achieve the following objectives.

1. To devise and execute an improved, innovative, and safeguarded storage framework.
2. The objective is to enhance information security by guaranteeing confidentiality, integrity, and authentication when storing and retrieving data in data centers.
3. The data can be accessed conveniently, regardless of its physical location in any geographically scattered data center throughout the network.
4. In order to reduce the amount of storage space used on cloud servers, data can be compressed prior to being stored.
5. In order to increase the security of data, one can encrypt the compressed data using CP-ABE with the RSA asymmetric encryption technique.
6. A decentralized and distributed approach is proposed to enhance the security of users' data and eliminate the involvement of a centralized authority.

#### **3.8.2 Significance of Research**

Cloud-based data storage is a modern trend that provides both free and extensive storage capacity. While the cloud offers numerous benefits, consumers are primarily concerned about security due to the storage of extremely sensitive data in the cloud. Data integrity is a crucial necessity in a cloud storage system that relies on sharing. Furthermore, the access control in the current cloud model necessitates one or more fully trusted attribute or central authority to uphold the access policy. If the central

authority is dismantled, the entire system is disrupted. Hence, the task of attaining security and privacy attributes for encrypted data is a substantial obstacle. Decentralized systems play a crucial role in ensuring integrity checking and access control, hence eliminating the potential threat posed by a trusted central authority. Hence, in order to enhance the efficiency of current applications, it is necessary to combine cloud storage and blockchain technology to propose a distributed cloud storage architecture based on blockchain. This architecture would offer secure and dependable cloud storage services for end-users.

### **3.8.3 Need of the Study**

The cloud storage system enables customers to conveniently access their data from any location and at any time, hence facilitating a flexible and cost-effective on-demand and pay-per-use framework. Users can utilize the cloud storage model to rent and pay for storage and computing services according to their own needs. The cloud offers numerous advantages and features to its customers, including cost-effective storage, adaptability, automatic updates, and disaster resilience. Nevertheless, safeguarding user privacy and guaranteeing data security are crucial due to the potential risk of data leakage when consumers save their data in the cloud. Furthermore, when users outsource their data to the cloud, they relinquish control over it, and cloud data is susceptible to potential security breaches and a variety of threats. The existing cloud storage architectures exhibit various shortcomings, including centralized data storage that undermines server security, and the reliance on trusted third parties that impacts user privacy. In addition, the current distributed cloud storage system stores data in several data centers or servers in a distributed way, albeit it is not entirely distributed. Multiple data centers house data at a high level of density. Consequently, a substantial quantity of data will be vulnerable if any of the servers or data centers are breached. Public media globally have extensively reported on security risks associated with cloud storage, including instances where users' confidential files have been unintentionally disclosed on iCloud. Regrettably, there are now no feasible remedies for ensuring the security of cloud systems. The challenges of the current cloud storage system are identified as follows:

**1. Storage Management:** The primary focus of most cloud storage services is the storage of data. The storage managers may encounter additional challenges related to the space needed for storing this data. Mere augmentation of the storage capacity is futile.

**2. Authentication:** Cloud storage providers may pose a threat to user privacy as they regularly collect user data and utilize it for financial gain. The typical customer serves solely as a data supplier, enabling storage providers to safeguard corporations.

**3. Security:** The existing cloud storage system offers services that are managed and processed by a central authority. Nevertheless, this particular arrangement is plagued by a vulnerability in which a single point of failure might occur. This vulnerability has a direct impact on the security feature that ensures uninterrupted access for cloud users.

**4. Privacy:** The design of cloud storage offers convenient services to consumers, but it also presents significant data privacy concerns due to the extensive collection, storage, and management of user data on cloud networks. Cloud customers rely on cloud service providers to handle their data, even if they have limited awareness of the data's transmission and the individuals that access it. Similarly, the distributed cloud storage architectures do not fully disperse the data among all cloud servers, which might lead to privacy concerns if any of the servers are attacked, thereby resulting in the leakage of user data.

**5. Integrity:** Data integrity issues may occur while storing and managing user data in the cloud. Third parties may edit or remove outsourced data without user authorization, posing a danger to its security. Opponents can also manipulate cloud data resources for the purpose of obtaining financial or political advantages, so endangering the integrity of the data. As a result of these issues, numerous solutions utilizing public verification techniques with a third-party auditor have been suggested. However, there are various concerns that may arise, including the possibility of irresponsible verification leading to distorted data integrity caused by auditors with malevolent intent. Hence, it is imperative to devise novel solutions to address the data integrity concerns inherent in the cloud storage system.

**6. Data Breach:** A data breach refers to the unauthorized release of protected data to the public by hostile users, which can occur due to human mistake, security vulnerabilities, or intentional actions. Depending on the level of sensitivity of the disclosed data, it can cause significant harm to the consumers. The data breach issue originates from the inadequate implementation of encryption mechanisms in the cloud.

It leads to the unauthorized acquisition of users' confidential data. Hence, the implementation of security measures such as cryptographic algorithms can effectively address data breach concerns.

**7. Loss of Data Control:** Cloud customers relinquish control over their sensitive or secret data to third parties, assuming that the data centers are adequately secured and establishing a trusting connection with the cloud service providers. However, the cloud users are not provided with the location of the user data. Consequently, the user lacks authority over the location or processing of the data on the cloud platform. Mandatory implementation of enhanced transparency measures is necessary to ensure that cloud users and service providers have clear visibility into the storage and processing of data.

### 3.8.4 Benefit of the Study

With the increasing amount of data and the security concerns associated with cloud storage, there is a need for a new storage system model that utilizes blockchain technology to address these security challenges. A revolutionary paradigm, known as a Blockchain-based Cloud Storage System, emerges from the integration of blockchain and cloud storage systems. The convergence of these advances offers significant advantages to both areas, including enhanced efficiency, decentralization, transparency, security, and improved cloud services. Blockchain technology offers a completely decentralized storage structure without the involvement of any centralized authority. Blockchain technology facilitates the implementation of novel cloud storage capabilities that are impervious to any alterations to the data. This innovation offers numerous advantages to cloud storage systems, which are elucidated as follows:

**1. Decentralization:** The utilization of blockchain technology's decentralized structure offers a viable approach to address the problem of having a single point of failure by removing the requirement for trusted intermediaries. In addition, the peer-to-peer architecture of the blockchain grants all network participants equal rights to validate and verify the accuracy of the data, ensuring the immutability of the system.

**2. Security:** The use of blockchain technology enhances the security of the cloud storage system by using the inherent characteristics of the blockchain network, including confidentiality and availability. The blockchain network securely records and stores all transactions as hash values using cryptographic techniques. Moreover,

blockchain transactions are authenticated by the network participants to ensure that the user's connection with the cloud storage system remains private. The blockchain network also facilitates off-chain storage as a means to support the availability feature.

**3. Privacy:** Blockchain offers robust security to the cloud storage system through its immutable, integrity, and transparent properties. The blockchain network securely saves data by organizing it into a series of interconnected blocks, with each block being linked to the one that came before it. Therefore, due to the safe and unchangeable consensus method, it is not feasible to modify or change the content of the blockchain.

**4. System Complexity:** The incorporation of blockchain technology into a cloud storage system greatly decreases the level of difficulty in implementing the system. Cloud infrastructure can be utilized to perform blockchain algorithms, hence reducing the cost associated with blockchain resources. The integration of blockchain with cloud technology offers numerous alternatives for implementing large-scale, cost-effective, and straightforward blockchain-based cloud storage systems.

**5. Feasibility:** Numerous prominent firms presently implement blockchain as a service initiative to assess the practicality of incorporating blockchain into cloud platforms. Several prominent firms, including Amazon, IBM, Microsoft, and Oracle, have introduced blockchain as a service platforms on cloud computing. The incorporation of blockchain with cloud technology offers significant advantages in diverse fields including the Internet of Things, healthcare, banking, supply chain, and multimedia.

### **3.8.5 The Framework of the Present Study**

Essentially, the task is to provide a framework that offers multiple security solutions for cloud storage, covering four distinct perspectives. The framework must compress the file using a compression technique, then encrypt the compressed file using an encryption algorithm, and then apply the SHA algorithm to a portion of the encrypted file. Lastly, generate a section with the specific information about the file. Subsequently, incorporate this block into the blockchain. The cloud storage architecture, which is based on blockchain technology, utilizes the Honeybee optimization algorithm to efficiently allocate resources and reduce transaction response and execution time.

### **3.8.6 Conceptual Framework**

The conceptual framework is a python-based design for blockchain architecture in cloud storage systems. It aims to address the drawbacks and issues associated with cloud storage by providing a secure environment. The suggested architecture utilizes CP-ABE to provide a key generation mechanism through the use of bilinear mapping-based cryptography and to execute data access control mechanisms. The framework efficiently handles key-related and user access policy details in a decentralized manner by leveraging the blockchain structure, resulting in a more resilient environment. The blockchain network stores metadata of user data and records all access and validation activities. The architecture utilizes a Honeybee optimization algorithm to maximize resource consumption in the cloud storage system and minimize transaction processing time. Through the utilization of blockchain technology, it attains decentralization while ensuring security without the need for a trusted central authority. The cloud storage system offers four main services to guarantee security features:

1. The registration process is created to enroll those who own data and individuals who utilize the data by employing a key generation approach to ensure authentication.
2. The data owners store the metadata of user data in the blockchain framework and establish access rules to ensure authorization.
3. Data owners ensure data integrity through the utilization of the Merkle root concept.
4. The cloud storage system saves the original data and employs an optimization algorithm to minimize the time required for transaction processing.

The experimental results, analysis, and performance evaluation confirm that the proposed work offers a practical and dependable cloud environment. Moreover, the comparative analysis demonstrates that the proposed approach surpasses the existing techniques. This research work has effectively developed a cloud storage system that is efficient, ideal, and safe by utilizing blockchain technology.

### **3.8.7 Proposed Model for UCCS**

The suggested model integrates the execution and evaluation of the many stages involved in creating a secure model.

Figure 3.43 depicts the architecture of a decentralized permissioned blockchain for a cloud storage system, which incorporates security measures. The suggested system model comprises four fundamental entities:

- **Decentralized Network:** A decentralized network preserves a transparent and tamper-proof framework for storing the metadata of both data and users. Each function call stores the relevant information in the decentralized structure. Hence, the data transmission among the users of the suggested framework is secure and easily comprehensible.
- **UCCS:** The primary objective of UCCS is to produce system parameters and keys for users based on their traits or identities. The UCCS is tasked with the outsourcing of encrypted data, the administration of access policies, the verification of integrity, and the upkeep of Meta information throughout the decentralized network.
- **User:** The data can be accessed by users only if their attributes meet the access criteria of the ciphertext. The user can utilize the decryption algorithm to decipher the file by employing the confidential key, resulting in the acquisition of the plaintext.
- **Distributed Storage:** The task of distributed storage is to store compressed and encrypted files that are uploaded by the UCCS. The data files are initially partitioned into several shards of equal size and then spread across multiple servers. The suggested architecture employs the Honeybee optimization method to efficiently handle resources during the data transaction process and reduce processing time.

| Notations                       | Description                                                  |
|---------------------------------|--------------------------------------------------------------|
| $GP$                            | Global parameters ( $p, \mathbb{G}, g, e, \mathbb{G}_T, H$ ) |
| $CF$                            | Compressed file                                              |
| $PK$                            | 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                                             |
| $\{A_1, A_2, A_3, \dots, A_n\}$ | User's attribute list                                        |
| $DO$                            | Data owner                                                   |

|                          |                                             |
|--------------------------|---------------------------------------------|
| $Fid$                    | SHA256 generated file hash                  |
| $TransID$                | Blockchain network generated transaction-ID |
| $S$                      | Number of servers                           |
| $\{B_1, B_2 \dots B_n\}$ | Number of blocks in each server             |

Table 3.31: Notation and Description

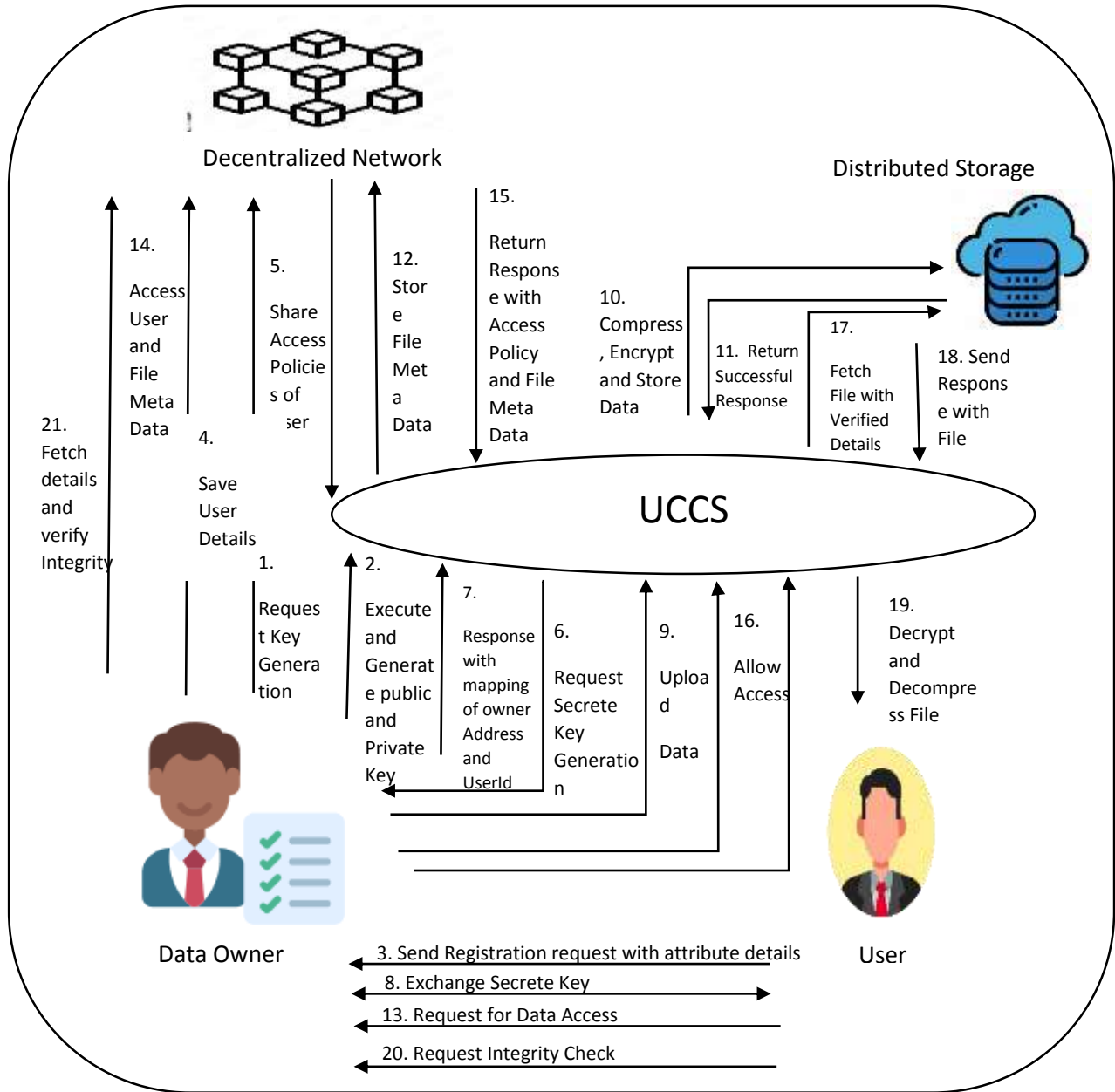


Figure 3.43: System Model for Proposed Framework

**Data Owner:** The data owner serves as an intermediary between UCCS, the decentralized network, and the user.

The suggested architectural workflow, seen in Figure 3.43, is as follows:

1. Data owners submit key generation requests to generate global parameters, as well as public and master keys, for the purpose of registering the proposed design.
2. The UCCS performs the key generation process, resulting in the creation of a public key and a master key for the data owner.
3. The user submits a registration request to the data owner, providing an attribute list that includes the user ID, email ID, contact number, address, date of birth, and other relevant information.
4. The data owner stores user information in the decentralized network.
5. User access policies share to UCCS by decentralized network.
6. Utilizing the CP-ABE technique to match its attribute list. The public key corresponds to the user ID that is shared in the attribute list. The data owner utilizes the bilinear pairing-based CP-ABE technique to create a confidential key.
7. The data owner creates user keys and establishes mappings between them using the produced keys. This information is then stored in the suggested architecture.
8. The responsibility of exchanging a secret key with the user via the Diffie-Hellman method lies with the data owner. The RSA algorithm use this shared secret key for both encrypting and decrypting the data.
9. The individual or entity who possesses the data transfers the data to the Ultra Compression Cloud Storage (UCCS).
10. UCCS utilizes file compression and employs the resulting secret key to encrypt data. The encrypted data is then stored on distributed storage using the capabilities of the proposed architecture.

11. Distributed storage Servers efficiently store data and optimize resources to minimize execution time and overhead during storage. The distributed storage system employs an optimization strategy to evenly spread the data load among servers and minimize the time required for transactional processes. And provides a successful response from the data store to UCCS.
12. UCCS creates the metadata of the file, including the file hash, Merkle root, mapping details of the file and owner/user, and then stores this information in the decentralized network.
13. The user submits the access request to the relevant data owner, including the file ID.
14. The data owner retrieves the user and file mapping information and the appropriate access policy from the decentralized network.
15. A decentralized network verifies access data and sends a response to UCCS.
16. UCCS may allow or deny access to user.
17. Following the access verification phase, UCCS retrieves the encrypted data from the distributed storage.
18. Distributed storage transmitted the requested file was transmitted to UCCS.
19. The user employs the secret key to decipher the encrypted data, decompress it, and gain access to the original content.
20. The user verifies the integrity of the received data by transmitting a request to the data owner.
21. The data owner retrieves the user and file information from the decentralized network, conducts the integrity verification procedure using the Merkle root mechanism, and then sends the outcome back to the user.

### **3.8.8 Objectives of the Framework implementation:**

1. To detect and address risk issues that arise during the storage and retrieval of data.
2. Analyzing the behavior and security technique of live Multifactor authentication model.
3. Comprehending the functioning of compression and decompression algorithms for the purpose of generating compressed files.
4. Comprehend the functioning of the key generation algorithm.
5. Describe the behavior of CP-ABE with RSA Encryption and Decryption Algorithms during the process of creating an encrypted file.
6. Optimize the resource in order to decrease both the execution time and the overhead associated with its storage.
7. The system carries out the integrity verification process utilizing the Merkle root technique.
8. Reports the behavior of distributed and decentralized servers during file upload and download operations.
9. Presenting an optimal implementation of this paradigm inside a decentralized setting.
10. Evaluation of the suggested framework, including its constraints and potential for future development.

### **3.8.9 Result and expected outcomes:**

#### **1) Core implementation:**

This is the initial stage of the proposed framework in which we implemented the proposed architecture in the python programming language.

#### **2) Ensure confidentiality, integrity and availability:**

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.

#### **3) Optimize the recourses:**

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.

4) Analyze security and performance:

Nextcloud is used to analyse and model a cloud computing environment.

5) Testing and Analysis:

The cost factors, performance factors, security factors, technical complication factors and other management factors form part of further assessment and analysis of the proposed framework.

6) Results and Findings:

The conclusions shall be based on the results of the proposed framework on basis of technical procedures and limitations and the overall benefits.

### **3.9 Algorithms used for building the framework**

Here we present the diverse algorithms that we have developed for constructing the framework. Firstly, we present the ultimate algorithm for the process of storing and retrieving data, as well as the interaction between the client and server.

#### **3.9.1 The ultimate algorithm**

The ultimate Algorithm is entirely derived from the five primary steps of the suggested framework.

Assuming that UCCS represents the multifactor set

$$UCCS = \{ f(a) \cup f(b) \cup f(c) \cup f(d) \cup f(e) \}$$

Where,

UCCS = Ultra Compression Cloud Storage

$f(a) \rightarrow$  Algorithm for key Generation

$f(b) \rightarrow$  Data outsourcing Algorithm

$f(c) \rightarrow$  Access Control Algorithm

$f(d) \rightarrow$  Algorithm for integrity checking

$f(e) \rightarrow$  Optimization Algorithm

**Global setup (GP):** The architecture under consideration employs a CP-ABE algorithm. The decentralized network chooses a bilinear group  $\mathbb{G}$  with a prime order  $p$  and a generator  $g$ . It also selects two random numbers  $a, b \in \mathbb{Z}_p$ . The data owner performs the global setup process, using the global parameter  $GP$  as input, and obtains the public key  $PK$  and master key  $MK$  as output. Let  $p$  represent a prime number, and let  $\mathbb{G}$  and  $\mathbb{G}_T$  denote multiplicative cyclic groups with an order of  $p$ . A mapping  $e: \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  that satisfies the following criteria is referred to as 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  $(g, g)$  is a generator of  $\mathbb{G}_T$
- $(u, v)$  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 ( $p, \mathbb{G}, g, e, \mathbb{G}_T, H$ ):** The UCCS employs a key generation method to create both the public and master key for registering the proposed system. Additionally, users submit registration requests to data owners, including attributes such as user ID, email ID, contact information, and so on. The user attribute list is denoted as  $\{A_1, A_2, A_3, \dots, A_n\}$ . UCCS receives global parameters  $(p, \mathbb{G}, g, e, \mathbb{G}_T, H)$  and master key  $MK$  as input. It then assigns an attribute policy to the user and generates a secret key  $SK$  and public key  $User_{id} = \{A_1 \in A_n\}$ . The key generation process is completed by UCCS through the execution of Algorithm 3.1 of the proposed architecture. The key generation process is represented using mathematical notation as follows:

1. Global parameters  $(p, \mathbb{G}, g, e, \mathbb{G}_T, H)$
2. Each user attribute list is represented as:  $A_i = \{A_1, A_2, A_3, \dots, A_n\} = \{1, 2, 3, \dots, n\}$
3. Then parameter list is denoted as:  $params = [p, g, e, g_2, h, Y =$

$$(g, g_2)^\alpha, T_1, T_2, T_3, \dots, T_n] \text{ where } \alpha \xleftarrow{R} \mathbb{Z}_p, g_2, h, T_i \xleftarrow{R} \mathbb{G}$$

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:  $MK_i = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$

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

$$SK_A = [A, d_1 = g^r, d_2 = g^{a_2}h^r, d_i = T^r_i, \forall i \in A] \text{ where } r \xleftarrow{R} \mathbb{Z}_p$$

**Compress (GP, F):** The UCCS employs the Huffman Coding compression algorithm for data compression. The compression algorithm accepts the global parameters GP and file F as input and produces the compressed file CF. Prior to encrypting the information using the CP-ABE algorithm, the file undergoes compression to decrease its size.

**Encrypt (GP, CF, SK<sub>A</sub>):** (UCCS) employs the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) algorithm for Encrypting the data. The encryption method receives global parameters GP, compresses file F, and utilizes secret key SK<sub>A</sub> as input to produce ciphertext CT. Prior to uploading the file onto the distributed storage, UCCS divides the file into many shards and calculates the Merkle<sub>root</sub> using the SHA256 algorithm. This Meta information is then stored in the decentralized network. The user details, specifically the application binary interface owner public key, are encoded as a JSON object. The process of uploading file F involves the following steps.

1. The UCCS employs the Diffie-Hellman algorithm to facilitate the exchange of a shared key, denoted as SK, by utilizing the user's public key and identity. Next, a shared key is employed to execute a methodical encryption process utilizing an RSA algorithm. Algorithm 3.2 is used to store this information on the decentralized network.
2. The UCCS algorithm partitions the file into several shards in order to determine the Merkle<sub>root</sub>, as specified in Algorithm 3.2. The computed Merkle<sub>root</sub> are stored on the decentralized network.

3. The UCCS utilizes the encryption algorithm to encrypt the compressed file  $CF$  using the shared secret key using the RSA asymmetric encryption technique. The user ID, also known as the public key, is saved along with the encoding output and the  $F_{id}$  on the distributed server, denoted as  $\{User_{id}, CT\}$ . The encryption procedure adheres to Equation 3.1.

*Encrypt* ( $[p, g, e, g_2, h, Y = (g, g_2), T_1, T_2, \dots, T_n], W = i_1 \wedge i_2 \wedge \dots \wedge i_k, CF \in \mathbb{G}_T$ )

$$CT = [W, C_1, C_2, C_3] \text{ Where } c_1 = F.Y^s, C_2 = g^s, C_3 = (h \prod_{j=1}^k T_{ij})^s, s \leftarrow^R \mathbb{Z}_p \quad (3.1)$$

**Decrypt**( $GP, CT, SK_A$ ): The decryption algorithm,  $\text{Decrypt}(GP, CT, SK_A)$ , necessitates the global parameters  $GP$ , encrypted text  $CT$ , and the common shared key  $SK_A$ . The UCCS initially acquires the User Identifier ( $User_{id}$ ) and File Identifier ( $F_{id}$ ) from the decentralized network, and subsequently verifies the existence of the requested file within the decentralized network. If the specified file is not there, the decryption procedure is abandoned. Alternatively, the UCCS permits a user to track the decryption procedure and acquire the compressed file by utilizing the produced common key, global parameters, and ciphertext. Decryption of the attribute set will result in a compressed file only if it satisfies the access policy criteria. Otherwise, the decryption process will fail. The decryption procedure of the CP-ABE with RSA, combined with a proof of correctness, can be described as follows:

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

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

$$CF = \frac{C_1 \cdot e(d_1, C_3)}{e(d, C_2)}$$

2. *Correctness* ( $SK_A, CT$ )

- $SK_A = [A, d_1 = g^r, d_2 = g^{a_2} h^r, d_i = T_{r_i}, \forall i \in A]$

- $CT = [W, C_1 = F.Y^s, C_2 = g^s, C_3 = (h^{k_{j=1}} T_{ij})^s],$

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

$$\begin{aligned} \frac{C_1 \cdot e(d_1, C_3)}{e(d, C_2)} &= \frac{CF \cdot e(g, g_2)^{\alpha s} \cdot e(g^r, h \prod_{j=1}^k T_{ij})^s}{e(g_2^\alpha h^r \prod_{i=1}^k T_{ij}^{r_{ij}}, g_s)} \\ &= \frac{CF \cdot e(g, g_2)^{\alpha s} \cdot e(g^r, (h \prod_{j=1}^k T_{ij})^s)}{e(g_2^\alpha, g^s) \cdot e(h \prod_{j=1}^k T_{ij}, g)^{rs}} \\ &= CF \end{aligned}$$

**Decompress (GP, CF):** The decompression procedure necessitates the utilization of the global parameter GP and the compressed file CF. The Huffman Coding decompression algorithm is used to extract the plaintext from a compressed file.

### 3.9.2 Smart Contract Function

This section provides a detailed explanation of the reasoning and computational procedures behind the proposed architecture. The operations of the proposed decentralized network are implemented using the Python programming language. This architecture incorporates a range of smart contract capabilities that are designed and implemented to offer diverse functionalities. The function initializes several variables and structures upon their creation. UCCS assumes ownership over the planned architecture, with its public key serving as the designated address of the owner. Any network address that performs distinct duties is classified as a sender.

1. The suggested decentralized framework preserves the metadata of users and data. The data owner uses this information to manage the access control and ensure the integrity validation procedure. The data owner use a hashing algorithm to maintain the Metadata in the decentralized structure, incorporating the user and file details.

2. The suggested architecture enables two primary functions: storing the physical file on distributed servers and storing user and file Meta information in a decentralized network to create an efficient and dependable environment.

The suggested decentralized network-based architecture incorporates the following functions:

The user registration function involves the execution of the key generation algorithm by UCCS to generate public and master keys for each data owner. The system employs the CP-ABE algorithm through the utilization of the bilinear mapping technique. Once the master key is generated, the UCCS employs the identical procedure to generate the secret key for the users. The secret key is distributed to the user through the utilization of the Diffie-Hellman algorithm, which relies on the  $User_{id}$  (public key) shared in the attribute list. The registration process for data owners and users is depicted in Figure 3.44.

i. Data owner sends a registration request to the UCCS.

$$DO \rightarrow \text{UCCS: } Registration_{request}$$

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

$$\text{UCCS} \rightarrow DO: Key_{generation}(p, \mathbb{G}, g, e, \mathbb{G}_T, H)$$

iii. UCCS 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.

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

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

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

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

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

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

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

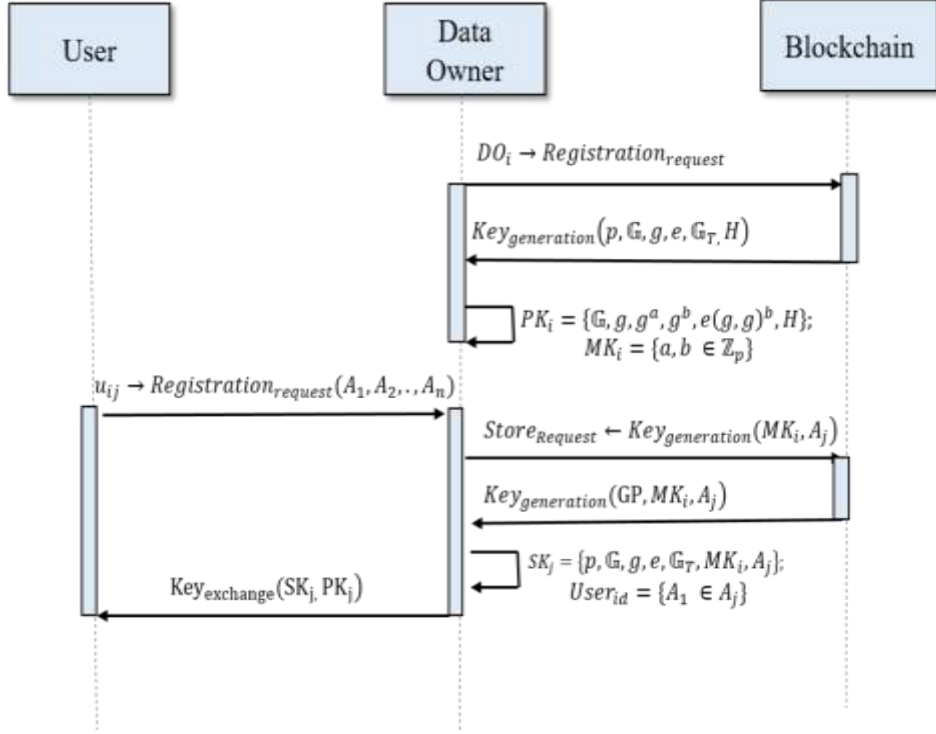


Figure 3.44: Registration Flow of Data Owner and User

---

### Algorithm 3.1: Key Generation

---

*Key<sub>generation</sub>*(GP)

---

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

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

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

**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*
-

Algorithm 3.1 involves the development of a master key for the data owner in line 2, which is accomplished by a key generation process. On the other hand, line 6 produces the confidential access codes for individuals who are part of distinct groups of data owners, such as  $DO_1 = \{u_{11}, u_{12}, \dots\}$ ,  $DO_2 = \{u_{21}, u_{22}, \dots\}$  and so on. The algorithm has a time complexity of  $O(n^2)$ .

**Data Outsourcing Function:** The data owners utilize this function to store the user data on distributed storage and manage Meta information on the decentralized network. UCCS is tasked with computing the  $Merkle_{root}$  by partitioning the file into numerous shards, and this data is stored on the decentralized network. This function utilizes a *SHA256* hashing technique to produce a hash and root hash. The diagram illustrating the entire sequence of the data outsourcing procedure may be seen in Figure 3.45, accompanied by a detailed description below:

- i. Data owner requests UCCS to outsource data on distributed storage.

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

- ii. In response, UCCS compress and encrypt data and store on distributed storage and Metadata details to be stored on decentralized network.

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

- iii. UCCS generates data Meta information using a *SHA256* hashing algorithm.

Initially, the data is partitioned into several shards, and a hashing method is employed to produce a hash value for each shard and compute the  $Merkle_{root}$ . The data owner stores both the  $user_{id}$  and data  $F_{id}$  in the decentralized network and distributed storage.

$UCCS: F_{ij} = \{f_1, f_2, \dots, f_n\}; F_{id} \leftarrow SHA256(F_{ij});$

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

- iv. UCCS sends generated Meta details to decentralized network for storage.

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

- v. After storing Meta details, the UCCS generates ciphertext using a shared secret key- based RSA encryption algorithm.

$UCCS: CT \leftarrow (GP, CF, SK_A)$

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

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

- vii. UCCS sends the data to the cloud for storage.

*UCCS → Cloud: Outsource<sub>data</sub>(F<sub>id</sub>, User<sub>id</sub>, CT)*

- viii. Distributed storage stores and optimizes the resource using Honeybee optimization algorithm to minimize execution and response time as explain in Algorithm 3.5.

*Cloud: Optimize<sub>resources</sub> (S)*

- ix. On successful storage of data distributed storage return a true value to the UCCS system.

*Cloud → UCCS: Return True*

- x. Decentralized network returns transaction ID to the data owner.

*Blockchain → DO: Return TransID*

Algorithm 3.2 outlines the sequential actions involved in the data outsourcing procedure. Lines 2-4 illustrate the procedures involved in generating Meta information, whereas lines 7 and 8 indicate the process of encryption and outsourcing. Line 9 signifies the invocation of the optimization technique employed by the distributed storage system to reduce the time taken for transaction processing. The temporal complexity of Algorithm 3.2 is linear, denoted as O(n).

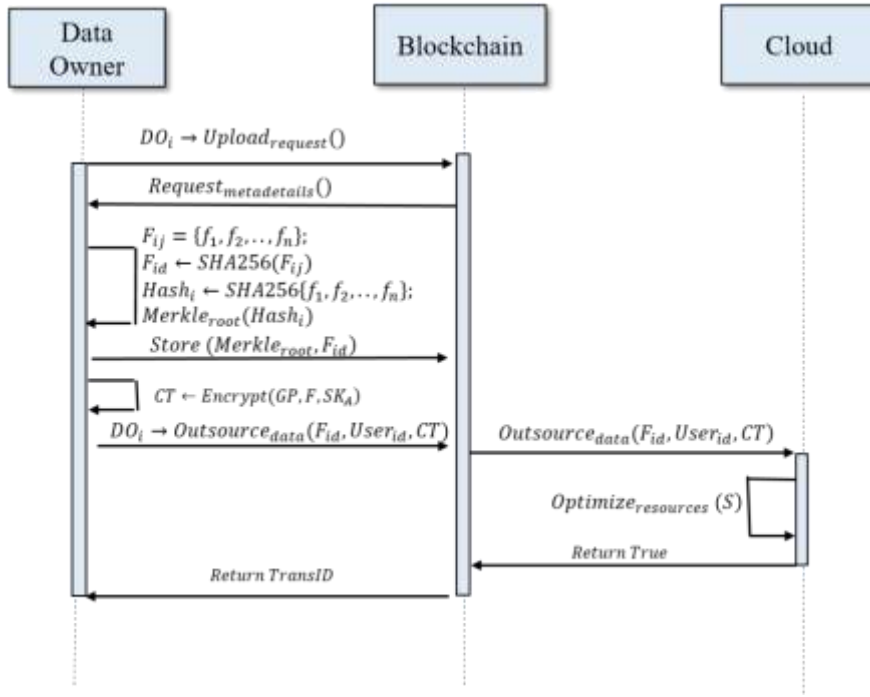


Figure 3.45: Data Outsource Workflow

---

### Algorithm 3.2: Data Outsourcing

---

*Upload<sub>request</sub>( )*

---

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

File Set  $F = \{F_{11}, F_{12}, F_{13}, \dots, F_{ij}\}$  where  $i \in DO$  number and  $j \in User$  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,  $F_i = \{f_1, f_2, \dots, f_n\}$
  3.     Calculate File Hash,  $Hash\ File_i \leftarrow SHA256(F_i)$
  4.     Generate and Store Merkle Root,  $Merkle_{Root}(Hash\ File_i)$
  5. *end for*
  6. *for Each File in F do*
  7.     Compress file Data  $CF \leftarrow Compress(GP, F)$
  8.     Encrypt File Data  $CF, CT \leftarrow Encrypt(GP, CF, SK_A)$
  8.     Outsource Data,  $Outsource_{Data}(CT)$
  9.     Optimize Resources,  $Optimize_{resources}( )$
  10. *end for*
-

**The Access Control Function:** It enables the UCCS to verify the access control policy prior to decrypting and decompressing the requested data. Users are granted permission to retrieve the data stored on the distributed storage. The UCCS initially verifies if users have the necessary access privileges based on the access rules published on the decentralized network. If the users possess rights, it implies that the attributes they share are a subset of the stored attribute list. In such cases, the requested file undergoes a process of decryption and decompression, resulting in plaintext. If the users do not have the necessary rights, the request is rejected and discarded. Figure 3.46 illustrates the overall flow of the access control system.

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 decentralized network.

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

iii. Decentralized network sends requested details to the UCCS.

$$Blockchain \leftarrow UCCS: (i_1, i_2, i_3, \dots, i_k); A_j$$

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

$$UCCS: if \{i_1, i_2, \dots, i_k\} \subset A_j$$

v. After access policy verification, the UCCS generate ciphertext.

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

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, decompress the received ciphertext using an RSA algorithm.

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

Algorithm 3.3 depicts the procedure for access control. The access policy verification process by the data owner is represented by line 2-3. Line 4 represents the process of decrypting the obtained ciphertext. Algorithm 3.3 has a constant time complexity for executing the simple statements.

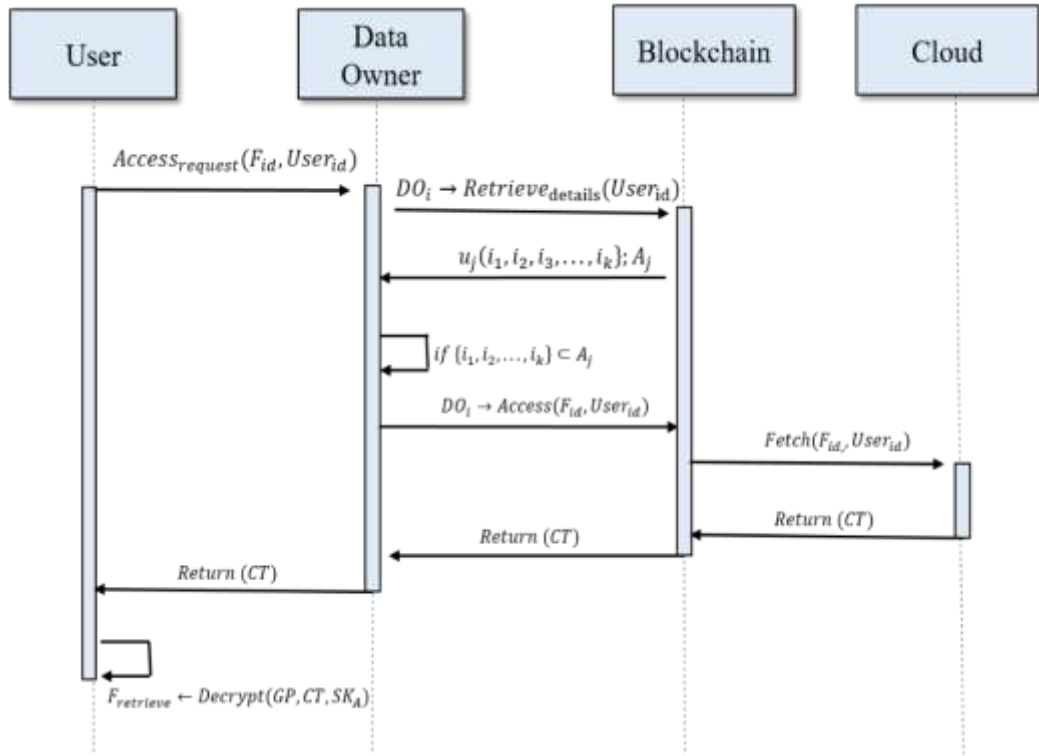


Figure 3.46: Access Control Workflow

---

### Algorithm 3.3: Access Control

---

*Accessrequest*( $F_{id}, User_{id}$ )

---

**Input:** Data Owner Set,  $DO = \{DO_1, DO_2, DO_3, \dots, DO_i\}; 1 \leq i \leq n$ ,  
 User Set,  $u = \{u_1, u_2, u_3, \dots, u_j\}$  where  $DO_i \in \{u_1, u_2, u_3, \dots, u_j\}$ , Attribute  
 List, *Attributelist*( $A_1, A_2, \dots, A_n$ ), Ciphertext  $CT$ , File ID  $F_{id}$

**Output:** File,  $F_{retrieve}$

---

1. User Send Access Request,  $u_i \leftarrow Accessrequest(F_{id}, User_{id})$
  2.  $DO$  Retrieve  $u_i$  Attribute Details *attribute<sub>details</sub>*{ $i_1, i_2, i_3, \dots, i_k \in User_{id}$ }
  3. if  $\{i_1, i_2, \dots, i_k\} \not\subset A$ , Access fails
  4. if  $\{i_1, i_2, \dots, i_k\} \subset A$  then  $F_{retrieve} \leftarrow Decrypt(GP, CT, SK_A)$  AND  
 Decompress(GP,  
     CF)
  5.  $u_i \leftarrow access(F_{retrieve})$
- 

**Integrity Checking Function:** It is responsible for verifying the integrity of the data saved in the cloud. Users have the ability to solicit the data owner for an integrity check of the file. The data owner retrieves the user and file information stored on the

blockchain and proceeds with the verification procedure. The data owner obtains the stored Merkle root of the requested file, recalculates the Merkle root of the file, and then compares the two to check for a match. If the results are matched, the integrity is confirmed; otherwise, it returns false. The integrity verification procedure consists of the phases depicted in Figure 3.47.

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.

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

Algorithm 3.4 outlines the specific operations involved in the integrity verification process. Line 2 indicates the process of accessing the stored Merkle root that corresponds to the file ID. Line 3 and 4 depict the process of deciphering the obtained encrypted text and reevaluating the Merkle root. Line 6 executes the process of comparing the stored and created Merkle roots. If the two roots match, the integrity is confirmed. Otherwise, it is not checked. Algorithm 3.4 has a constant time complexity for executing the simple statements.

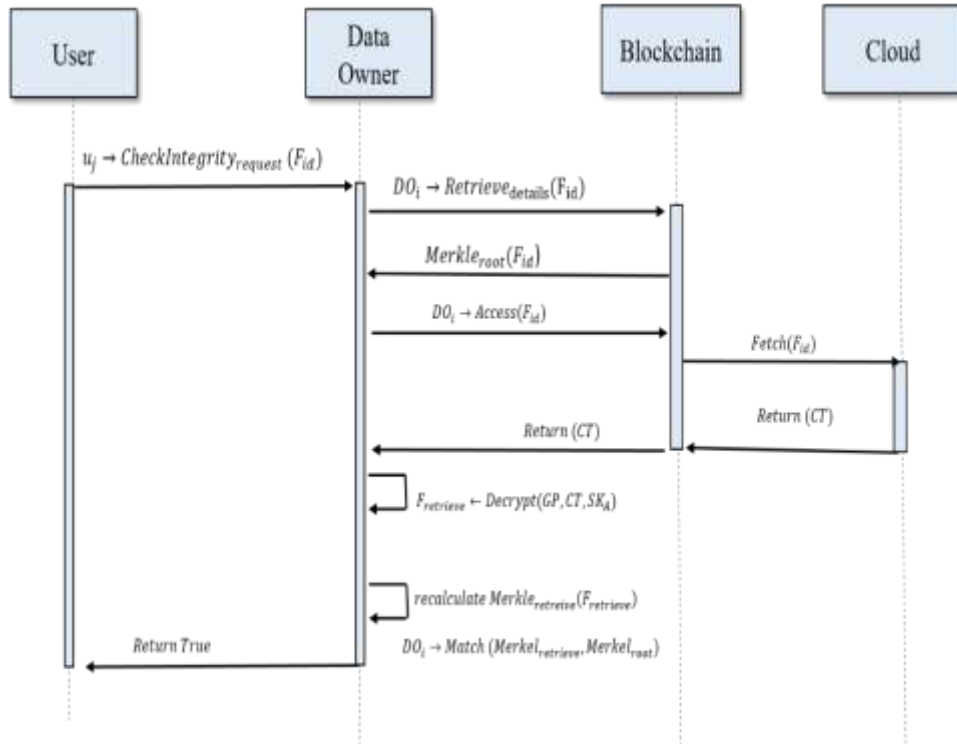


Figure 3.47: Integrity Checking Process

---

**Algorithm 3.4: Integrity Checking**

---

*Ch*  $eckIntegrity_{request}(F_{id})$

---

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

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

Merkle Root,  $\text{Merkle}_{root}$

**Output:** *True/False*

---

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

**Optimization Function:** This function is responsible for optimizing the process of cloud servers to efficiently utilize resources and decrease response time. The cloud data can be accessed by several users or authorities in different ways, including data uploading, data downloading, data requests, and receiving returned results. In this

study, a Honeybee optimization model is implemented to optimize the process of transactional data and effectively minimize the time required. This algorithm seeks to optimize the memory allocation of each server (virtual machine). Consequently, an equitable amount of bandwidth is assigned to each server, so preventing the burden of excessive tasks.

Algorithm 3.5 first chooses a cloud environment consisting of  $S$  servers, each of which is partitioned into  $n$  blocks (virtual computers). Therefore, the algorithm has a temporal complexity of  $O(n)$ . The program is designed to identify the blocks or virtual machines in the overloaded server and locate the less overloaded blocks. It then evenly distributes the load among these blocks, resulting in minimized response time. The task can be likened to a bee that is searching for a less burdened block (food source). Once it discovers the appropriate block, the task is assigned to it. The subsequent task also attempts to allocate the identical block, persisting until the block's capacity hits the threshold value. After obtaining the threshold value, the algorithm identifies the block with the lowest resource utilization and redirects the task to that block.

The process commences by randomly dispatching the  $S$  scout bees to choose sites that are either underloaded or heavily overloaded. The fitness value is assessed for each site and arranged in descending order. The local search algorithm identifies the optimal place or sites that are most suitable for a given purpose. The finest locations are divided into two categories:  $\text{Group}_{\text{foragerbees}}$  and  $\text{NonGroup}_{\text{foragerbees}}$ . The quantity of bees engaged in foraging activities is denoted as  $B_{\text{GFA}}$ , while the quantity of bees not engaged in foraging activities is denoted as  $B_{\text{NGFA}}$ . The local search process commences by enlisting the forager bees situated in the most optimal vicinity. The global search process is a stochastic search procedure at non-optimal locations. Ultimately, the places are arranged based on their fitness value, and this process continues until the global optimum is discovered.

---

**Algorithm 3.5: Optimization**

---

*Optimize*<sub>resources</sub>(*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),  $S = \{B_1, B_2 \dots B_n\}$
  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*<sub>foragerbees</sub> to Select from  $B_{avg.block}$  as  $B_{NGFA}$
  6. Declare  $Set_{neigh\ bourservers}$  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 *Neigh borh ood*<sub>search</sub> (*Group*<sub>foragerbees</sub>, *NonGroup*<sub>foragerbees</sub>)
  13.       Select *Best*<sub>foragerbees</sub> as *average block size* as  $B_a$
  14.       Re-estimate  $Fitness_{value}$  for each *Average*<sub>datablock</sub>
  15.       Sort Results of *Average*<sub>datablock</sub>
  16.       Reallocate Remaining *forager bees* to other Locations
  17.       Evaluate  $Fitness_{value}$  for each *Average*<sub>datablock</sub>
  18.       Arrange *bees* from *local* and *global* search
  19. Process Continue until  $Max_{iter}$
- 

### 3.10 Example Scenario

Consider, ABC University seeks to autonomously manage its data in the cloud in a decentralized manner, without relying on external entities. Initially, it will establish a blockchain network to facilitate a registration procedure for its personnel. The heads of departments of ABC University, such as CSE, ME, EE, ECE, CE, etc., function as data owners inside the proposed blockchain network, represented as  $\{DO_1, DO_2, DO_3, \dots, DO_n\}$ . The department is comprised of faculty members, denoted as users  $\{u_1, u_2, u_3, \dots, u_n\}$ , who are affiliated with the department and report to the head or data owner. Hence, the suggested framework enables data owners to enroll in the network and

thereafter take on the responsibility of enrolling their department's faculty/users utilizing the key generation method  $K()$  of the proposed system. The heads/data owners are responsible for managing the information pertaining to faculty and department files. The blockchain network stores information about heads and staff details. Head also manages the access policy for their employees or users and grants access based on attribute data, such as  $attributede\{i_1, i_2, i_3, \dots, i_k\}$ , within the  $User_{id}$ . The access control algorithm  $A()$  and integrity checking method  $CheckIntegrity_{request}()$  guarantee privacy and allow users to access department data through the corresponding data owners. Consequently, the suggested design includes numerous departmental heads or data owners, with each department housing multiple users.

### 3.11 Security Analysis

This section examines the security attributes of the proposed architecture in order to mitigate security vulnerabilities. The suggested framework implements additional measures to guarantee security and privacy features for cloud users. The suggested architecture offers the following security methods to address security threats:

**Authentication:** Users of the proposed architecture must complete the registration process in order to acquire the public keys. Public keys serve as the users' addresses for their authentication in order to carry out the fundamental operations of the proposed architecture. Prior to granting access to its services, the blockchain network validates the identity of users.

**Privacy Protection:** The suggested framework guarantees the secrecy of the information retained by the data proprietor. It guarantees that the data of the data owner will remain inaccessible to unauthorized users. Furthermore, it employs encryption methods to securely communicate data between the data owner, user, blockchain, and cloud. The data owner keeps encrypted user and data information within the blockchain network. Hence, anyone without proper authorization are prohibited from accessing the data stored on the network. The blockchain structure guarantees that all operations are both non-destructive and non-repudiable. The access policy and user details are stored securely in the blockchain network, guaranteeing data confidentiality and privacy.

**Key Security:** The suggested architecture implements the conventional CP-ABE algorithm within the blockchain network to provide key security. The system allows for the registration of numerous data owners to produce user keys, rather than relying on a single authority or key management authority. This approach reduces the workload and ensures a dependable environment. The data owner, who is highly regarded, creates the key for the data user. Due to the potential hazards associated with transmitting keys, both the data owner and the user utilize the Diffie-Hellman algorithm, a key exchange protocol, to communicate the key. Hence, the suggested framework ensures the protection of the cryptographic keys.

**Access Control:** The suggested architecture employs a fine-grained access control system and eliminates a central authority, resulting in a more decentralized access mechanism. The system employs an attribute-based access control mechanism to grant user access permissions. During the registration procedure, users furnish attribute details to the data owner. The data owner uploads the attribute list to the blockchain network and defines the attribute policy using the subset operator. Access to the requesting user is granted if the attributes shared by the user are included in the saved attribute list. Once access is granted, users proceed with the decryption and decompression procedure to acquire the plaintext by utilizing a confidential key.

**Integrity Protection:** The user has the ability to request that the data owner validate the integrity of the data. The suggested architecture use the Merkle root idea to verify the integrity of the data. The data owner preserves the metadata of the data while uploading it to the blockchain network. Thus, the individual who possesses the data retrieves the stored Merkle root, recalculates the Merkle root of the requested data, and carries out the verification process. If the hash matches for both, then the integrity of the data is preserved; otherwise, it returns false to the user.

# CHAPTER 4

## ANALYSIS AND INTERPRETATION

### 4.1 Introduction

Due to the exponential expansion of Information Technology, a substantial volume of data is produced annually, posing challenges in managing this data at a local level. Hence, the cloud storage system is regarded as the optimal and cost-effective solution to address this problem. Storing confidential information on cloud servers raises concerns about trust. Therefore, it is necessary to employ reliable cryptographic methods to safeguard user privacy, enforce access control, and encrypt the data prior to uploading it to the cloud. In this work, an innovative model is developed to accurately forecast data security and data security categorization. In addition, machine learning techniques such as support vector machine (SVM), random forest, and deep learning approach called artificial neural network (ANN) are utilized for data security analysis.

Conventional cryptography systems are unable to offer a versatile information exchange mechanism and precise access control.

To construct a model with a decentralized network, several strategies such as attribute-based encryption, attribute-based access control, and key management processes are employed to establish access control for distributed storage and assure data security. The suggested architecture utilizes CP-ABE to provide a key generation mechanism through the use of bilinear mapping-based cryptography and to execute data access control mechanisms. The data owner and UCCS jointly handle the key-related and user access policy details in a distributed way, leveraging a decentralized network to create a stronger and more resilient environment. The developed architecture securely saves publicly accessible information in a decentralized network while also managing the storage of dispersed data.

This chapter is structured into two sections: the first section focuses on quantitative

analysis and its interpretation, while the second section delves into qualitative analysis and its interpretation.

## 4.2 Quantitative Analysis and its Interpretation

Quantitative analysis employs a dataset that comprises twenty-seven relevant parameters. The constructed model was subsequently assessed using these parameters. The presented models have the ability to accurately anticipate data security and suggest the appropriate data security category based on a greater level of resilience.

### 4.2.1 System Architecture

Five unique models were employed to carry out data security prediction analysis in order to evaluate their performance.

This system design offers a comprehensive elucidation of the complete procedure of data management.

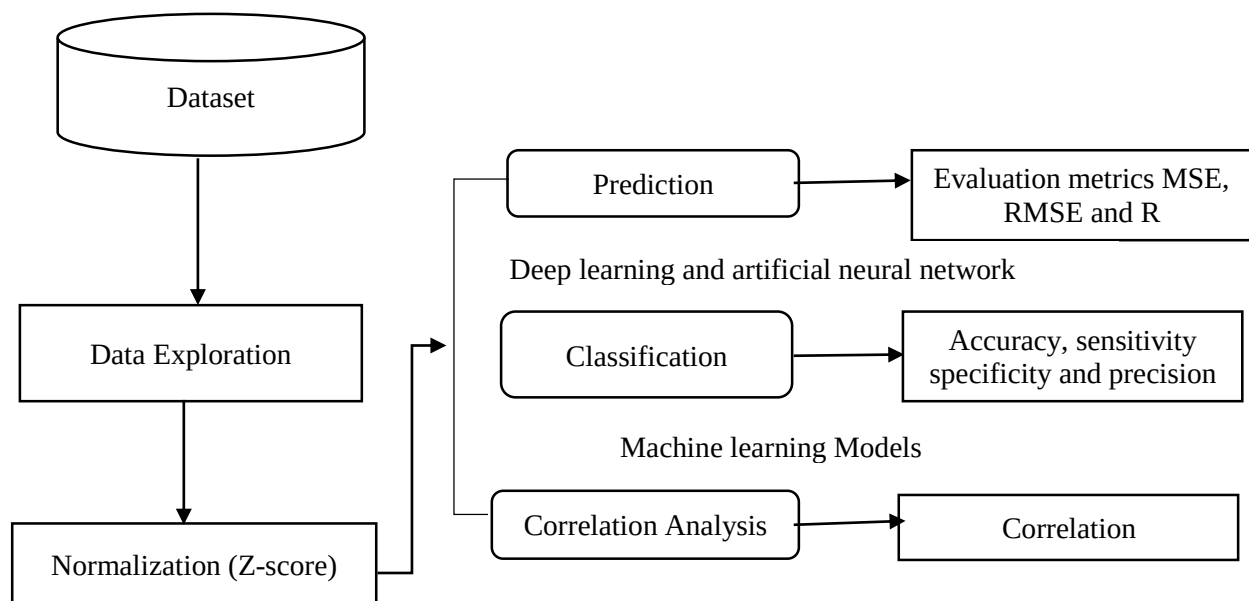


Figure 4.1 System Architecture

- During the beginning stage, it is essential to consider the dataset for data exploration, as shown in the picture.
- In the context of normalization, the z-score is a metric utilized to calculate the deviation between a data point and the mean.
- The implementation of the prediction pointer will improve the process of data analysis and make prediction tasks easier. It has the ability to enhance the

evaluation metrics of RMSE and MSE.

- Classification is essential for maintaining the accuracy, sensitivity, specificity, and precision.
- Correlation analysis is a statistical method used to identify and maintain the relationship between variables.

#### 4.2.2 Parameter Simulation

1. **Define parameters:** Define the data security settings for simulation, such as the desired level of protection, the allocated budget for cloud service, the size of data to be uploaded, the frequency of data uploads, and the required bandwidth for data transfer.
2. **Set up the simulation:** Develop a Python script that establishes a simulation by specifying the equations or models required to simulate the parameters.
3. **Run the simulation:** Execute the Python script to carry out the simulation.
4. **Visualized the result:** Utilize Python's plotting and visualization features to generate graphs and charts that accurately depict the simulation outcome.

#### 4.2.3 Evaluation of proposed framework

The experiments have been conducted using Python 3.x series along with a range of scientific libraries.

##### 1. Dataset

The dataset utilized in this study is obtained from various individuals engaged in cloud computing. The dataset consists of 10 major factors, specifically the Level of Security, Factors to consider while evaluating cloud service options include the budget allocated for cloud services, the level of dependency on cloud services, the volume of data to be uploaded to the cloud, the frequency of data uploads, and the necessary bandwidth for data uploads.

Challenges encountered by an organization, whether sensitive data is uploaded or not, it is important to specify the encryption standard and authentication method utilized.

## **2. Data preprocessing**

The preprocessing phase is crucial in data analysis since it enhances the quality of the data. The z-score method has been employed as a data normalization tool to achieve higher accuracy.

## **3. Data security measures**

To assess data security, one must consider multiple criteria that have a major impact. This study utilizes a published dataset to evaluate the proposed model, incorporating seven crucial data security characteristics.

The list of algorithms used for the result comparison are as follows:

1. Multivariate Regression
2. Decision Tree
3. Random Forest
4. SVM
5. ANN

### **4.2.3.1 Multivariate Regression**

Multivariate regression is a supervised machine learning approach that analyzes many data variables. Multivariate regression is a statistical technique that expands upon multiple regression by allowing for the analysis of one dependent variable in relation to several independent variables. We attempt to forecast the output by considering the quantity of independent factors.

Multivariate regression aims to establish a mathematical equation that can elucidate the simultaneous response of variables to changes in one another.

Utilizing multivariate regression analysis can help to examine the links between dependent and independent factors and their impact on outcomes, as the collected data includes multiple variables.

Mean Squared Error: 0.27

R-squared: 0.47

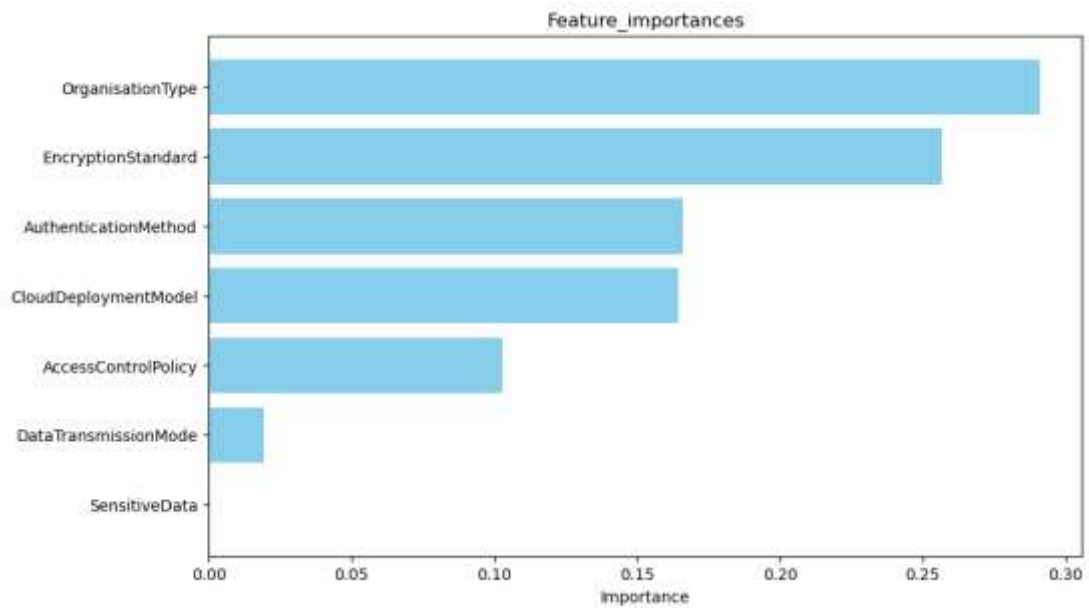


Figure 4.2: Multivariate Regression Feature Performance plot

MSE = 0.27 and R-squared = 0.47

Evaluate the model's quality of fit to the data and its ability to forecast the target variable.

#### 4.2.3.2 Decision Tress Classifier

Decision Tree is a supervised learning algorithm that can handle both classification and regression issues. However, it is primarily utilized for classification tasks.

A decision tree consists of two fundamental components: The Decision Node and the Leaf Node. A choice node is utilized for decision-making and contains several branches, while a Leaf node represents the result of that decision and does not have any additional branches.

Accuracy: 0.71

precision recall f1-score support

|   |      |      |      |    |
|---|------|------|------|----|
| 1 | 0.82 | 0.82 | 0.82 | 11 |
| 2 | 0.00 | 0.00 | 0.00 | 1  |
| 3 | 0.33 | 0.50 | 0.40 | 2  |

accuracy 0.71 14

|              |      |      |      |    |
|--------------|------|------|------|----|
| macro avg    | 0.38 | 0.44 | 0.41 | 14 |
| weighted avg | 0.69 | 0.71 | 0.70 | 14 |

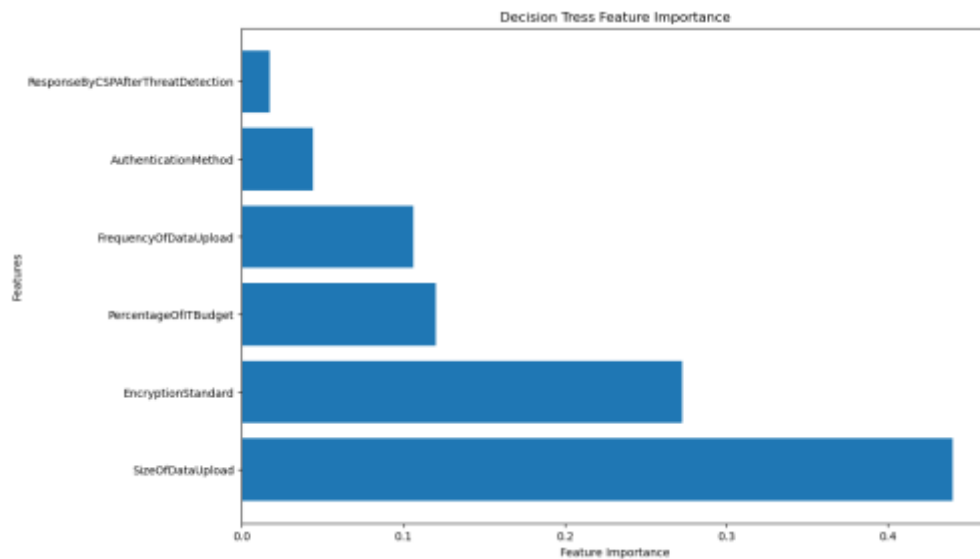


Figure 4.3 Decision Tree Classifier Feature Performance plot

An Accuracy score of 0.71 signifies that your classification model accurately identified the class for 71% of the cases in the dataset.

#### 4.2.3.3 Random Forest

Random forest, a popular supervised machine learning technique, is commonly used for classification and regression issues. The algorithm employs diverse samples to build decision trees, relying on their collective agreement for classification and their mean value for regression. The Random Forest Algorithm is particularly notable for its capacity to effectively handle data sets containing both continuous components, such as in regression, and categorical variables, such as in classification. It performs more effectively for tasks involving classification and regression.

Accuracy: 0.8571428571428571

| precision | recall | f1-score | support |
|-----------|--------|----------|---------|
|-----------|--------|----------|---------|

|   |      |      |      |    |
|---|------|------|------|----|
| 1 | 0.85 | 1.00 | 0.92 | 11 |
| 2 | 0.00 | 0.00 | 0.00 | 1  |

|   |      |      |      |   |
|---|------|------|------|---|
| 3 | 1.00 | 0.50 | 0.67 | 2 |
|---|------|------|------|---|

|              |      |      |      |    |
|--------------|------|------|------|----|
| accuracy     |      |      | 0.86 | 14 |
| macro avg    | 0.62 | 0.50 | 0.53 | 14 |
| weighted avg | 0.81 | 0.86 | 0.82 | 14 |

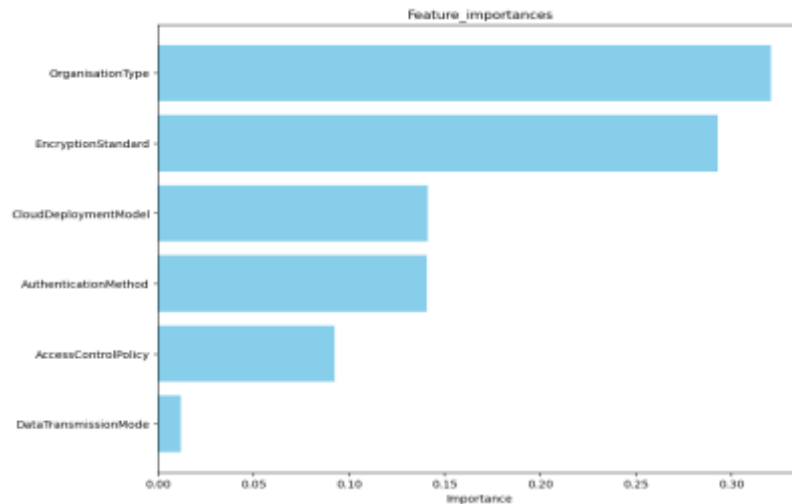


Figure 4.4: Random Forest Feature Performance plot

Applying the Random Forest algorithm to the dataset yields an accuracy of 0.8571428571428571.

#### 4.2.3.4 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a highly popular supervised learning algorithm utilized for resolving classification and regression problems. However, its primary application is in addressing Machine Learning Classification problems.

The primary goal of the SVM method is to determine the optimal line or decision boundary that can effectively separate an n-dimensional space into distinct groups. This enables us to efficiently categorize additional data points in subsequent instances.

Accuracy: 0.7857142857142857

|           |        |          |         |
|-----------|--------|----------|---------|
| precision | recall | f1-score | support |
|-----------|--------|----------|---------|

|   |      |      |      |    |
|---|------|------|------|----|
| 1 | 0.83 | 0.91 | 0.87 | 11 |
| 2 | 0.00 | 0.00 | 0.00 | 1  |
| 3 | 0.50 | 0.50 | 0.50 | 2  |

|              |      |      |      |    |
|--------------|------|------|------|----|
| accuracy     |      |      | 0.79 | 14 |
| macro avg    | 0.44 | 0.47 | 0.46 | 14 |
| weighted avg | 0.73 | 0.79 | 0.75 | 14 |

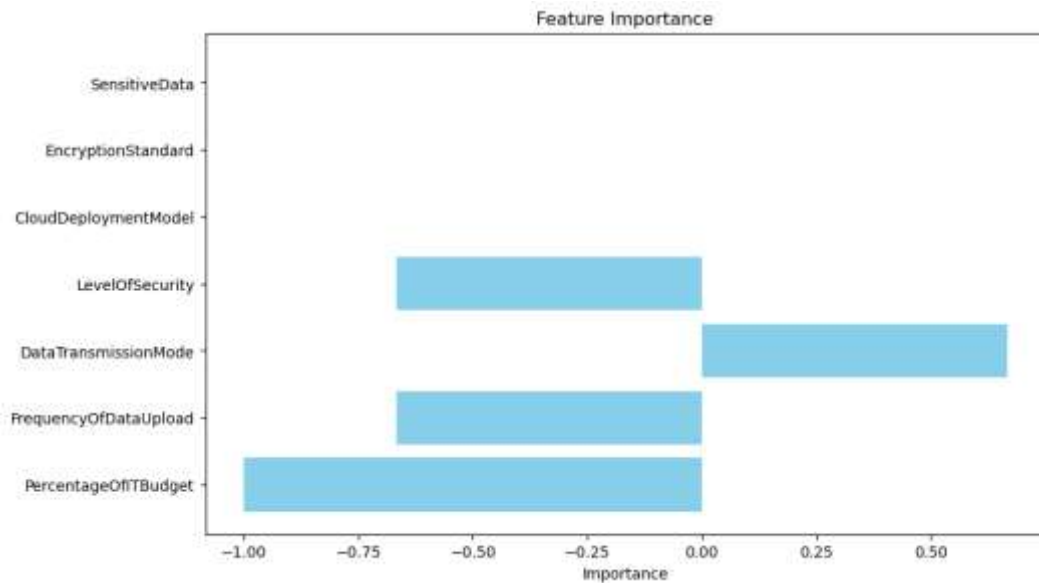


Figure 4.5: SVM Feature Performance Plot

Applying Support Vector Machine to the dataset yields an accuracy of 0.7857142857142857.

#### 4.2.3.5 Artificial Neural Networks (ANN)

Artificial neural networks (ANNs) are computational models that draw inspiration from the complex neural networks found in the human brain. Neural networks are an integral component of deep learning and are employed for a multitude of tasks, such as classification, regression, clustering, and even generative tasks.

Artificial Neural Networks (ANNs) are very effective models that possess the ability to acquire intricate patterns and correlations inside datasets. Neural networks are composed of interconnected layers of neurons that analyze data and generate output. Artificial neural networks (ANNs) are trained by employing the backpropagation method and optimization techniques to decrease mistakes. Various categories of artificial neural networks (ANNs) are well-suited for specific data kinds and tasks, rendering them very adaptable instruments in contemporary machine learning and

artificial intelligence (AI) applications.

Test Accuracy: 0.6428571343421936

precision    recall    f1-score    support

|   |      |      |      |    |
|---|------|------|------|----|
| 0 | 0.80 | 0.73 | 0.76 | 11 |
| 1 | 0.00 | 0.00 | 0.00 | 1  |
| 2 | 0.25 | 0.50 | 0.33 | 2  |

|              |      |      |      |    |
|--------------|------|------|------|----|
| accuracy     |      |      | 0.64 | 14 |
| macro avg    | 0.35 | 0.41 | 0.37 | 14 |
| weighted avg | 0.66 | 0.64 | 0.65 | 14 |

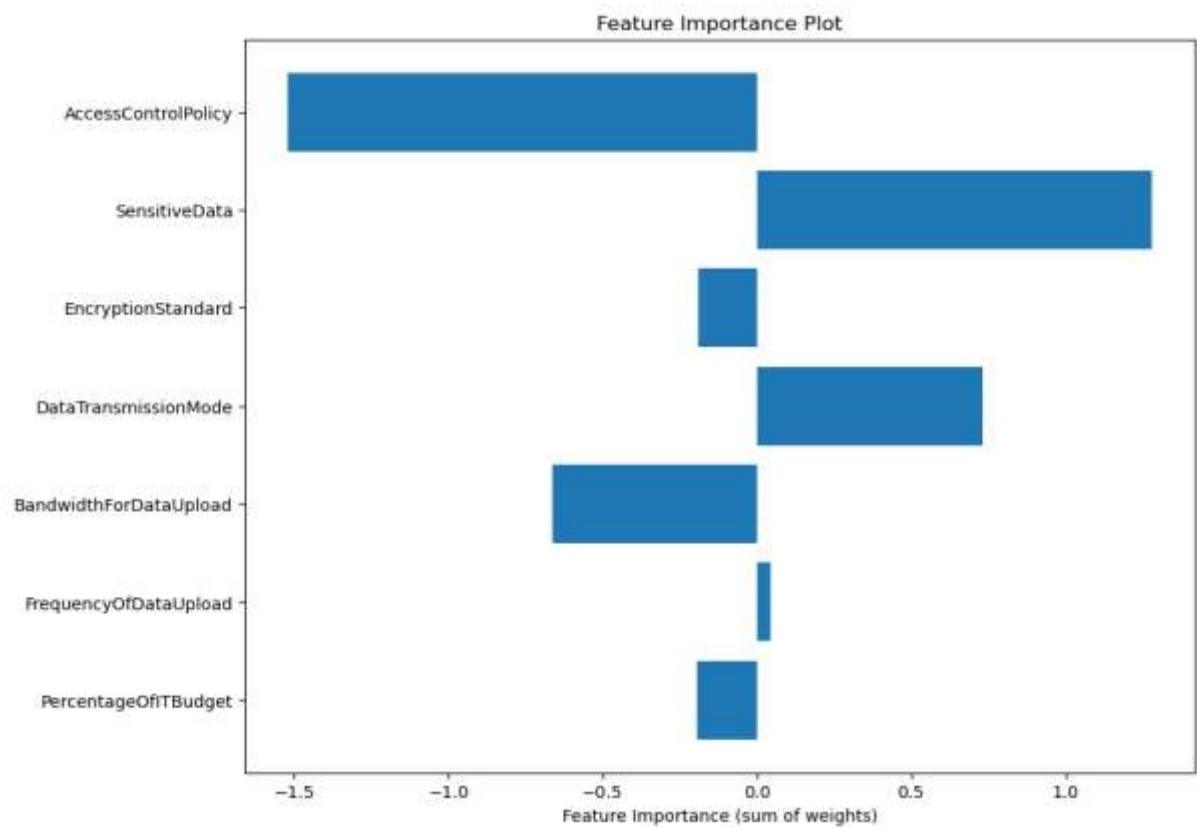


Figure 4.6: ANN Feature Performance Plot

**Observations:**

After implementation of all the five algorithms

| Sr. No. | Model                   | Accuracy Score                              |
|---------|-------------------------|---------------------------------------------|
| 1       | Multivariate Regression | Mean Squared Error: 0.27<br>R-squared: 0.47 |
| 2       | Decision Tree           | 0.71                                        |
| 3       | Random Forest           | 0.8571428571428571                          |
| 4       | SVM                     | 0.7857142857142857                          |
| 5       | ANN                     | 0.6428571343421936                          |

Table 4.1: Statistical models and its accuracy

### 4.3 Algorithms used in proposed framework

The suggested system utilizes the Huffman coding algorithm for data compression in order to effectively manage space on distributed storage. In addition, it utilizes the Ciphertext Policy Attribute-based Encryption (CP-ABE) with the RSA algorithm to guarantee secrecy, integrity, and availability. In addition, the Honeybee optimization method was implemented on a distributed storage system to enhance resource utilization and reduce transaction response and execution time. Below, we will explain the experimental findings of all the algorithms mentioned above.

#### 4.3.1 Compression Technique used in UCCS - Huffman Coding

One effective method for managing and analyzing large quantities of data is through the use of compression techniques, which can significantly reduce file size. The file with diminished size is convenient to store and manipulate. The UCCS employs the Huffman coding technique for data compression. The algorithm is outlined as follows: The link provided is: <https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/>

#### Algorithm Huffman (c)

```
{
  n = |c|
  Q = c
  for i <- 1 to n-1
  do
  {
    temp <- get node ()
    left [temp] Get_min (Q) right [temp] Get Min (Q)
    a = left [temp] b = right [temp]
    F [temp] <- f[a] + [b]
    insert (Q, temp)
  }
  return Get_min (0)
}
```

This is an example of the compression algorithm for a file

**The original File is:** Huffman coding is a data compression algorithm.

**The encoded File is:**

```
110001101011000000000110010011110000111110110011111011100011001111101
110001010011001010110110101000011111001101101010010110000101110111111
11100111100010101010000111100010111111011110100011010100
```

**The decoded File is:** Huffman coding is a data compression algorithm.

Note that the input file's storage is  $47 \times 8 = 376$  bits, but our encoded file only takes 194 bits, i.e., about 48% of data compression.

**Compression Ratio:** It is defined as the ratio of size of the compressed file to the size of the source file.

**Compression Ratio =  $C2 / C1 * 100\%$**

C1 – File size before Compression

C2 – File size after Compression

**Compression Ratio =  $194 / 376 * 100\% = 51.59$**

**Saving Percentage:** Saving Percentage calculates the shrinkage of the source file as a percentage

$$\text{Saving Percentage} = (C1 - C2 / C1) * 100\%$$

$$\text{Saving Percentage} = (376 - 194 / 376) * 100\% = 48.40$$

**Compression Factor:** It is the inverse of compression ratio. That is the ratio between the size of source file and the size of the compressed file

$$\text{Compression Factor} = C1 / C2$$

$$\text{Compression Factor} = 376 / 194 = 1.93$$

Huffman coding (Huffman, 1952) is the famous coding technique which effectively compress data in almost all file formats.

The above Huffman coding algorithm when applied on file it will give the result as follows:

| Sr. No. | Name of File   | Size of file (KB) | Size of file after compression (KB) | Compression Ratio | Saving Percentage | Compression Factor |
|---------|----------------|-------------------|-------------------------------------|-------------------|-------------------|--------------------|
| 1       | Actpendin.txt  | 678               | 412                                 | 60.76696          | 39.23303          | 1.64563            |
| 2       | Boc3.txt       | 771               | 461                                 | 59.79247          | 40.20752          | 1.67245            |
| 3       | Carrier.txt    | 1842              | 914                                 | 41.61997          | 50.38002          | 2.01531            |
| 4       | Inhist.txt     | 4706              | 2840                                | 60.34849          | 39.65150          | 1.65704            |
| 5       | NEPEC.txt      | 1009              | 618                                 | 61.24876          | 38.75123          | 1.63268            |
| 6       | Property.txt   | 2                 | 1                                   | 50                | 50                | 2                  |
| 7       | README.txt     | 3                 | 1.20                                | 40                | 60                | 2.5                |
| 8       | Revocation.txt | 191               | 107                                 | 56.02094          | 43.97905          | 1.78504            |
| 9       | Statsgo.txt    | 63                | 39                                  | 61.90476          | 38.09523          | 1.61538            |
| 10      | Zopener.txt    | 3                 | 1.55                                | 51.66666          | 48.33333          | 1.93548            |

Table 4.2: Table showing result of Huffman coding algorithm

Based on the information provided in the table, it can be inferred that, on average, the Huffman coding technique is capable of compressing 50% of the data.

#### 4.3.2 Encryption Technique used in UCCS – RSA based CP-ABE

UCCS adheres to the Attribute Based Encryption (ABE) standard. Attribute-based encryption (ABE) is an extension of public-key encryption that allows for precise control over access to encrypted data through the use of authorization policies<sup>1</sup>. The user's secret key and the ciphertext are determined by factors such as their email address, country of residence, and subscription type<sup>1</sup>. ABE is a contemporary method that reevaluates the notion of public-key cryptography<sup>2</sup>. Within the context of ABE (Attribute-Based Encryption), only individuals with the specific keywords encoded in their cryptographic key are able to successfully unlock the lock<sup>3</sup>.

##### **RSA based CP-ABE encryption is used**

UCCS utilizes CP-ABE to encrypt data and implement a detailed access control by incorporating an access policy. In order to enhance performance efficiency, this study opted for a CP-ABE algorithm based on RSA and including an access-tree structure.

A CP-ABE system typically consists of four primary algorithms. The four steps involved are setup, key generation, encryption, and decryption, respectively. During the setup process, the CP-ABE system employs the RSA algorithm to select two prime numbers. Prime numbers are utilized to generate a master public key and a master private key. The key generation algorithm generates a secret key for a certain set of attributes by utilizing the master private key. During the encryption process, a ciphertext is generated using an access tree. The decryption method can only decode the encrypted data if the qualities of the user's decryption key meet the requirements of the access policy. This approach utilizes the creation of a lightweight cryptographic system that is based on RSA. The system also supports a versatile monotone tree access structure, which allows for the implementation of complicated access control as a more general system. By employing this technique, the encryption and decryption procedures are enhanced in terms of efficiency and security.

- **RSA:** This algorithm is applicable for the encryption and signature of data. The encryption and signature procedures are executed via a sequence of modular multiplications.
- *The fundamental RSA algorithm for authentication can be elucidated as follows.*

$\text{ciphertext} = (\text{plaintext})^d \bmod n$

$\text{plaintext} = (\text{ciphertext})^e \bmod n$

private key = {d, n}

public key = {e, n}

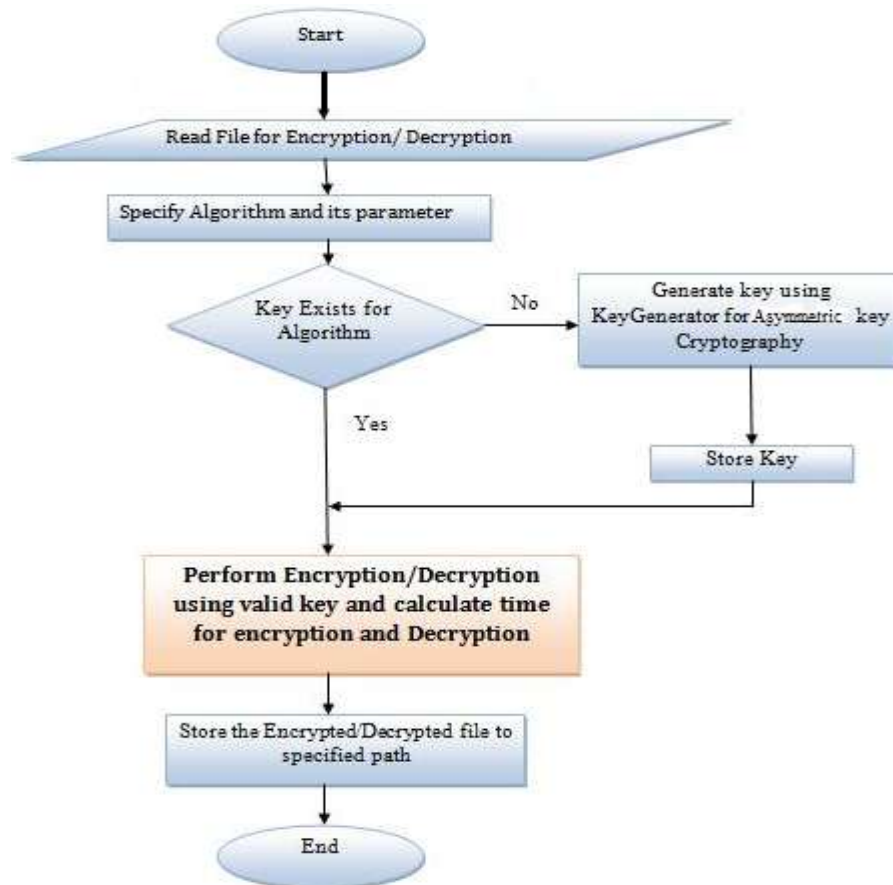


Figure 4.7: RSA Encryption Flowchart

The flow chart provides a concise overview of how program flow control is executed. The program reads the input file for the purpose of encryption or decryption. The program verifies the presence of the key for the algorithm. If the key is absent, the GenerateTheKey technique is employed to create the key for executing the cipher operation. The created key is saved to the designated location. If the key already exists at the supplied path, it is utilized. The resulting file is subsequently saved to the designated location.

The effectiveness of the proposed upgraded CP-ABE with RSA method and existing methods is evaluated based on parameters such as encryption, decryption, and execution time. The discussion of these approaches is shown below.

- **The encryption time:** is used to measure the throughput of a model, providing information on the pace of encryption. The throughput of the encryption model is determined by dividing the total number of bytes in the plaintext by the time it takes to encrypt.

- **Decryption time:** refers to the duration required to transform encrypted data back into its original format. Decryption is the procedure that reverses encryption, allowing authorized users to decode encrypted data using a secret key.

- **Execution time:** The execution time, also known as central processing unit (CPU) time, refers to the amount of time the system spends executing a job or the time it takes for a program to complete or terminate.

- **Comparative analysis**

The results of the proposed CP-ABE with RSA scheme are compared to earlier approaches, including AES and CP-ABE, in terms of dynamic and secure public auditing of user data in the cloud. The comparison is conducted to evaluate the encryption and decryption time for key lengths ranging from 8 to 256 bits. The table 4.3 presents a comparison of the encryption time between the current methods.

| Key Length | AES[11] | HMCA-ECC [12] | LSSS [13] | CP-ABE[16] | SDVC [17] | Enhanced CP-ABE withRSA Method |
|------------|---------|---------------|-----------|------------|-----------|--------------------------------|
| 8          | 27241   | 24631         | 23834     | 23783      | 22151     | 21070                          |
| 16         | 19158   | 17643         | 15203     | 15010      | 15006     | 14994                          |
| 32         | 9920    | 9370          | 8963      | 8794       | 8526      | 7795                           |
| 64         | 4972    | 4631          | 4421      | 4365       | 4512      | 4436                           |
| 128        | 2742    | 2621          | 2567      | 2542       | 2556      | 2544                           |
| 256        | 1648    | 1647          | 1742      | 1721       | 1713      | 1638                           |

Table 4.3: The comparison of encryption time with existing methods for different key lengths

| Key Length | AES[11] | HMCA-ECC [12] | LSSS[13] | CP-ABE[16] | SDVC [17] | Proposed Enhanced CP-ABE withRSA Method |
|------------|---------|---------------|----------|------------|-----------|-----------------------------------------|
| 8          | 24747   | 24316         | 23247    | 23005      | 22378     | 20299                                   |
| 16         | 17873   | 16423         | 15642    | 14512      | 14453     | 14382                                   |
| 32         | 9042    | 8061          | 7783     | 7592       | 7431      | 7394                                    |
| 64         | 4593    | 4326          | 3921     | 3836       | 3821      | 3813                                    |
| 128        | 2265    | 2167          | 2126     | 2053       | 2035      | 2019                                    |
| 256        | 1175    | 1163          | 1137     | 1134       | 1117      | 1102                                    |

Table 4.4: The comparison of decryption time with existing methods for different key length.

| Key Length | AES [11] | HMCA-ECC [12] | LSSS [13] | CP-ABE [16] | SDVC[17] | Proposed Enhanced CP-ABE with RSA Method |
|------------|----------|---------------|-----------|-------------|----------|------------------------------------------|
| 8          | 52617    | 50341         | 47234     | 46717       | 45436    | 41977                                    |
| 16         | 37509    | 34216         | 32016     | 30112       | 30106    | 29990                                    |
| 32         | 19468    | 17482         | 16995     | 16993       | 16473    | 15822                                    |
| 64         | 10061    | 9234          | 8921      | 8859        | 8858     | 8856                                     |
| 128        | 5494     | 5361          | 5206      | 5196        | 5196     | 5198                                     |
| 256        | 3449     | 3452          | 3450      | 3459        | 3453     | 3437                                     |

Table 4.5: The comparison of execution time with existing methods for different key length.

Table 4.3 displays the duration it takes to encrypt using the improved CP-ABE with RSA technique, in comparison to the AES and CP-ABE methods currently in use. The encryption duration grows as the number of the data bits increases when using ways to secure public auditing of user data files in the cloud. The proposed method demonstrated a shorter encryption time compared to existing methods and is well-suited for ensuring the confidentiality of user's sensitive outsourced data. The proposed method achieved an encryption time of 1638 milliseconds for a key length of 256 bits. The current AES and CP-ABE algorithms took 1648 ms and 1721 ms, respectively, for a key length measured in bits. Figure 4.8 displays a graphical depiction of the encryption time comparison between the proposed CP-ABE with RSA scheme and existing approaches.

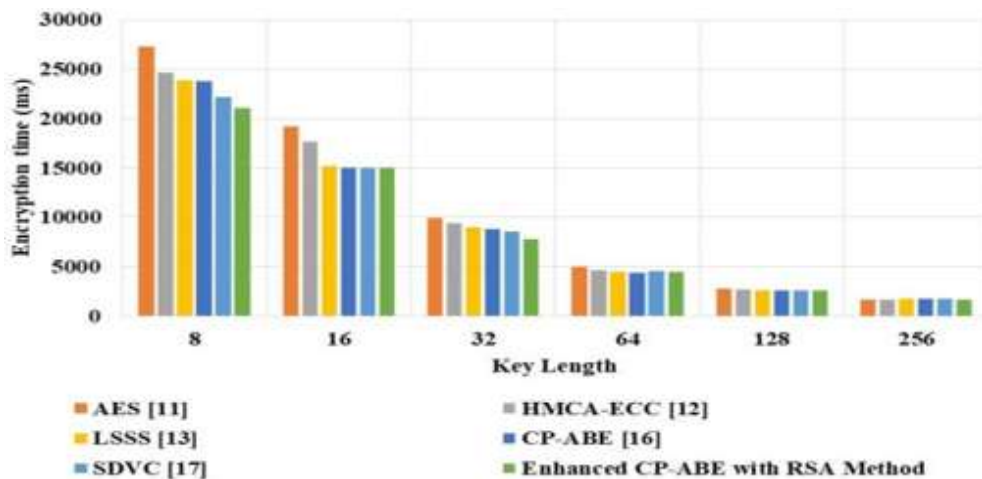


Figure. 4.8: Comparison of proposed enhanced CP-ABE with RSA method and existing methods for encryption time

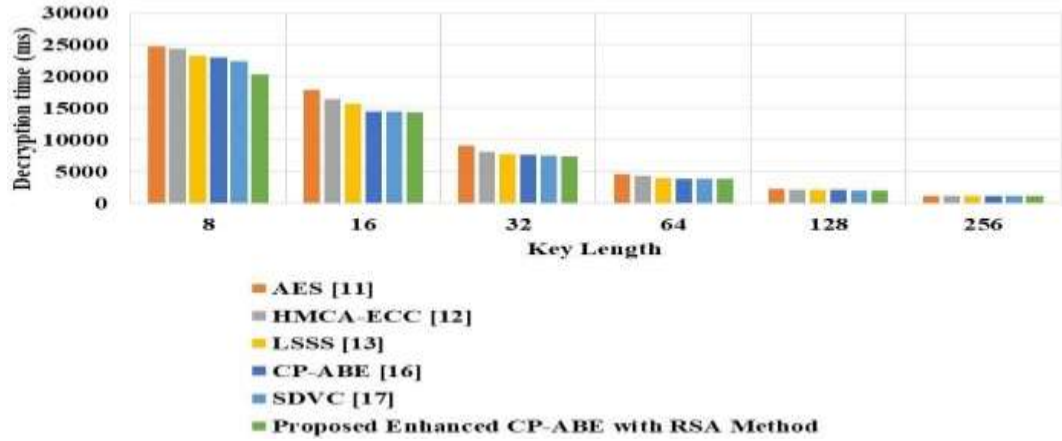


Figure. 4.9: Proposed enhanced CP-ABE with RSA method for decryption time with existing methods.

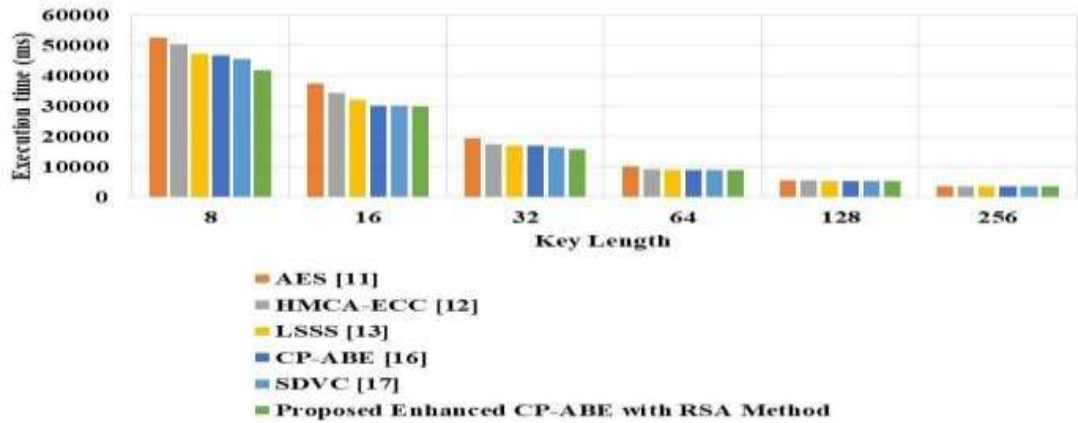


Figure. 4.10: Proposed enhanced CP-ABE with RSA methodfor execution time with existing methods.

Figure 4.9 displays a graphical depiction of the decryption time comparison between the proposed Enhanced CP-ABE with RSA scheme technique and the existing approach. Table 4 displays the duration of the implemented improved CP-ABE using the RSA technique, as well as the currently used AES and CP-ABE. The execution time of security techniques for user data files in the cloud decreases as the size of data bits rises. The proposed method demonstrated a shorter execution time compared to existing methods and is well-suited for ensuring the confidentiality of user's outsourced sensitive data. The proposed approach achieved an execution time of 3437 milliseconds for a key length of 256 bits. The AES [11], HMCA-ECC [12], LSSS [13], CP-ABE [16], and SVDC [17] achieved execution times of 3449 ms, 3452 ms, 3450 ms, 3459 ms, and 3453 ms, respectively, for a key length of 256 bits. Figure 4.10 displays a graphical

depiction of the comparison between the execution time of the proposed enhanced CP-ABE with RSA method and existing methods. This study effort offered an upgraded CP-ABE using the RSA approach to ensure security for outsourced data in the cloud. The proposed upgraded CP-ABE utilizing the RSA method accurately detects the user responsible for decrypting the ciphertexts during the decryption process. The data owner generates public key (PK), master secret key (MSK), and unique identifier (UID) using the setup process. The data owner transmits the confidential key to the user and stores it in the server proxy. The individual who possesses the data applies encryption to the information and transmits the resulting encrypted text to the cloud environment using the CP-ABE technique. The RSA technique is employed to decrypt the user's encrypted sensitive data. The cloud environment verifies the users' harmful intent by utilizing their unique identification codes. The malevolent users were unable to acquire the encrypted texts from the cloud environment. After retrieving the ciphertext from the cloud environment, the user obtains the relevant portion of the secret key and proceeds to perform the decryption process. During the revocation process, the encrypted message will receive the update from the public key (PK) and the master secret key (MSK). The experimental results demonstrate that the proposed and modified CP-ABE with RSA scheme achieved reduced implementation time for the encryption and decryption procedures. The upgraded CP-ABE with RSA technique has an execution time of 3437 ms, while the existing CP-ABE method has an execution time of 3459 ms. The proposed technique exhibits efficient performance as a result of its utilization of a tree structure and its implementation of privilege control.

### **4.3.3 UCCS Optimization Algorithm - Honeybee**

The UCCS design utilizes the Honeybee optimization algorithm on a cloud storage system to enhance resource utilization and reduce transaction response and execution time.

The HoneyBee algorithm is as follows:

Input: List\_statehost , List\_stateVM

Output: VM(j)

// return the number (i) of host which have minimum processing time

// i is the number of specified host

```

1: i ← -1
2: minPT ← Integer.MAX_VALUE
3: For each host(i) in List_statehost
4: If host(i) available then
5: If (PThost(i) < minPT) then
6: minPT = PThost(i)
7: i ← number of the current host
8: End if
9: End if
10: End for
// return the number (j) of VM which has minimum count of requests
// j is the number of specified VM
11: j ← -1
12: mincount ← Integer.MAX_VALUE
13: For each VM(j) in List_VMhost(i)
14: If VM available then
15: If (Count_REQVM(j) < mincount) then
16: mincount = Count_REQVM(j)
17: j ← number of the current VM
18: End if
19: End if
20: End for
21: if (j = -1) then
22: append coming task in waiting queue until one VM become available.
23: else
24: Allocate the task to VM(i).
25: Update allocated information i.e.; how many tasks are being processed, current
processing time of the host and VM and check the availability of VM and host
26: De-allocate the task from this VM after end of the task execution.
27: Update allocated information i.e.; how many tasks are being processed, current
processing time of the host and VM and check the availability of VM and host.
28: end if

```

After implementation of above algorithm in python the output is as follows:

**Best solution:** [-0.39260649 0.44259909 0.55036851 -1.10824326 -0.53879438  
0.1142193 -1.06572317 0.18143159 0.82686861 1.6803034]

**Best fitness:** 0.6629554699388609

#### **4.3.4 UCCS Integrity Verification Technique – Merkle Root**

A Merkle Tree, also known as a hash tree, is a data structure employed in the fields of computer science and cryptography to effectively and securely validate the integrity of extensive data collections. The process operates by the iterative hashing of node pairs until only one hash remains, referred to as the root hash or Merkle root.

##### **Merkle Tree Algorithm Overview**

###### **1. Leaves:**

Begin with discrete data components, referred to as individual leaves. A leaf node hash is generated by hashing each individual leaf.

###### **2. Hashing Pairs:**

Aggregate pairs of leaf hashes (or hashes from the preceding level) into a unified hash. This is achieved by combining the two hashes and subsequently hashing the resulting combination. This represents the subsequent tier of the tree.

###### **3. Repeat:**

Continue iteratively applying the hashing process to pairs of nodes from the previous level until only a single hash value remains. The Merkle root is represented by this final hash.

###### **4. Handling Odd Numbers:**

To ensure an even number of nodes at each level, duplicate the last node if the count is odd before pairing.

##### **Example**

Let's break down a basic example with four leaves:

###### **1. Leaves:**

Data: D1, D2, D3, D4

Hash each leaf: H1, H2, H3, H4

2. Level 1:

Combine and hash pairs:  $H1 + H2 \rightarrow H12$ ,  $H3 + H4 \rightarrow H34$

3. Level 2:

Combine and hash the results:  $H12 + H34 \rightarrow H1234$

4. Root:

H1234 is the Merkle root.

After implementing the Merkle Tree in python the output will be as follows:

Merkle Root: [computed\_merkle\_root\_hash]

Leaf hash found in leaves.

Data 'Transaction 2' is valid.

#### **4.3.5 UCCS Multifactor Authentication Technique**

Time-based One-Time Password (TOTP) is a widely used mechanism for Multi-Factor Authentication (MFA). The system produces a unique password by combining the current time with a secret key that is shared. Both the server and the client must possess knowledge of the key.

The output obtained after implementing the MFA algorithm in Python is as follows:

Enter username: user

Enter password: .....

Current OTP: 496978

Enter OTP: 496978

OTP verification successful!

Multi-factor authentication successful!

#### **4.3.6 UCCS Access Control Policy - ABAC**

UCCS employs ABAC, which stands for Attribute Based Access Control Policy. The ABAC implementation is as follows:

### **1. Define Attributes:**

Subject Attributes: (userID, userTitle, DegreeOfResponsibility)

Resource Attribute: (fileID, fileType, Owner, Sensitive)

Environmental Attributes: (networkState)

### **2. Define Policies**

If user.userTitle == "Executive" AND resource.Sensitive == "Yes" AND

environment.networkState == "Secure"

THEN allow access

### **3. Evaluate Access Request**

Input: userAttributes, resourceAttributes, environmentAttributes

### **4. Policy Evaluation**

```
function evaluate_access_request ( userAttributes, resourceAttributes,  
environmentAttributes):
```

```
    for policy in policies:
```

```
        if evaluate_policy (policy, userAttributes, resourceAttributes,  
environmentAttributes):
```

```
            return "Access Granted"
```

```
    return "Access Denied"
```

### **5. Policy Evaluation Logic**

```
function evaluate_policy(policy, userAttributes, resourceAttributes,  
environmentAttributes):
```

```
    conditions = policy['conditions']
```

```
    for condition in conditions:
```

```
        If not check_condition(condition, userAttributes, resourceAttributes,  
environmentAttributes):
```

```
            return False
```

```
    return true
```

## 6. Condition Check

```
function check_condition(condition, userAttributes, resourceAttributes,  
environmentAttributes):
```

```
    If condition['attribute'] == "user.userTitle":
```

```
        return userAttributes['userTitle'] == condition['value']
```

```
    elif condition['attribute'] == "resource.sensitive":
```

```
        return resourceAttributes['sensitive'] == condition['value']
```

```
    elif condition['attribute'] == "environment.networkState":
```

```
        return condition['value'][0] <= environmentAttributes['networkState'] <=  
condition['value'][1]
```

On implementation of above algorithm in python it gives the output as:

Access Denied

## 4.4 Comparative Study of UCCS with GFS and HDFS

| Features                              | File System |     |      |
|---------------------------------------|-------------|-----|------|
|                                       | UCCS        | GFS | HDFS |
| Distributed File System               | ✓           | ✓   | ✓    |
| Decentralized Network                 | ✓           | ✗   | ✗    |
| Attribute Based asymmetric encryption | ✓           | ✗   | ✗    |
| Compression Technique                 | ✓           | ✗   | ✓    |
| Resource Optimization                 | ✓           | ✓   | ✓    |
| Integrity Verification                | ✓           | ✓   | ✓    |
| Multifactor Authentication            | ✓           | ✓   | ✓    |
| Access Control Policy                 | ✓           | ✓   | ✓    |
| Secure Storage                        | ✓           | ✓   | ✓    |

Table 4.6: Comparison of UCCS with GFS and HDFS

#### 4.4.1 Distributed File System:

##### a. UCCS distributed File System:

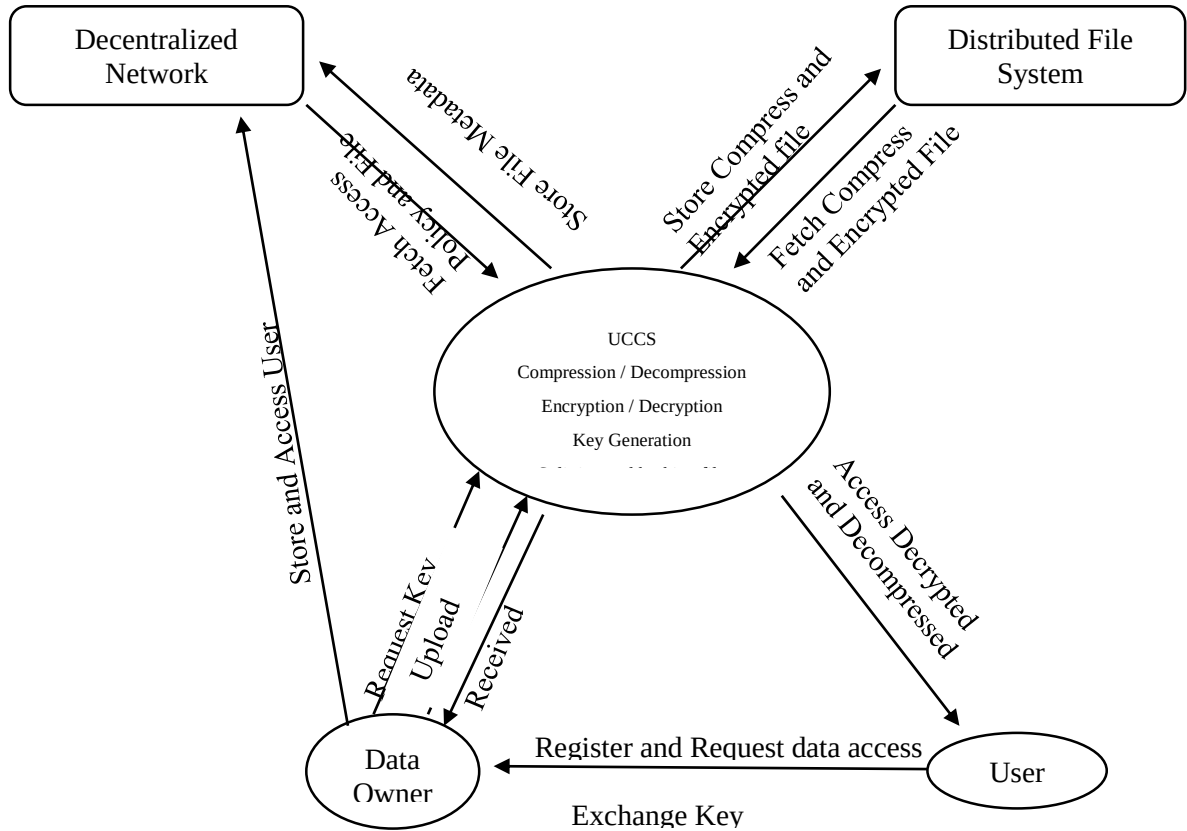


Figure 4.11: UCCS architecture

The UCCS utilizes a distributed file system to store files that have been both compressed and encrypted. The files are initially partitioned into numerous shards of equal size and then dispersed across multiple servers.

##### Components of UCCS:

**Decentralized Network:** A decentralized network preserves a transparent and tamper-proof structure for storing the metadata of both data and users. Each function call stores the relevant information in the decentralized structure. Hence, the exchange of data among the users of the suggested framework is secure and easily comprehensible.

**UCCS:** The objective of UCCS is to generate global parameters and keys for the data owner and users based on their attributes / identities. The UCCS is tasked with

outsourcing compressed and encrypted files, overseeing access controls, and keeping Meta information in the decentralized network.

**User:** Users are able to retrieve the data only if their attributes meet the access criteria of the ciphertext. The user can retrieve the unencrypted and uncompressed file and acquire the original text.

**Distributed File System:** The distributed file system is tasked with the storage of compressed and encrypted files that are submitted by the UCCS. The initial step involves partitioning the data files into several shards of equal size, which are then spread across several servers. The suggested architecture employs the Honeybee optimization method to efficiently handle resources during the data transaction process and reduce processing time.

**Data Owner:** The data owner is accountable for initiating the request for user key creation to the UCCS, facilitating the exchange of the produced key with the user, and securely storing the user's metadata on a decentralized network.

#### **b. Google File System:**

Google Inc. created the Google File System (GFS), which is a scalable distributed file system (DFS), in order to address the increasing demands for data processing within the corporation. GFS provides robustness, reliability, scalability, availability, and efficiency to large networks and interconnected nodes. The GFS consists of many storage systems built using affordable components of commodity hardware. Google's search engine generates vast amounts of data that need to be stored. This is just one example of how the search engine is tailored to match Google's specific data usage and storage needs.

The Google File System mitigated hardware deficiencies while capitalizing on the advantages of commercially available servers.

GoogleFS is a synonym for GFS. The system handles two distinct categories of information, specifically file metadata and file data.

The GFS node cluster has a solitary master and multiple chunk servers that are regularly accessed by different client systems. Chunk servers store data on local disks in the format of Linux files. The stored data is divided into large (64 MB) segments and duplicated at least three times throughout the network. The larger chunk size leads to a decrease in network overhead.

GFS is designed to fulfill Google's extensive cluster needs without impeding applications. Files are stored using hierarchical directories that have path names. The master is responsible for overseeing the management of metadata, which includes tasks such as handling namespaces, controlling access, and mapping data. The master establishes communication with each chunk server through periodic heartbeat messages and monitors their status updates.

The largest GFS clusters consist of over 1,000 nodes and have a disc storage capacity of 300 TB. This is readily accessible to a multitude of clients at all times.

### **Components of GFS**

GFS is comprised by a collection of computers. A cluster is a collection of interconnected computers. Each cluster has the potential to contain a large number of computers, ranging from hundreds to even thousands. The GFS cluster consists of three fundamental entities:

- **GFS Clients** refer to computer programs or applications that are used to request files. Users can submit requests to view and alter existing files or add new files to the system.
- **The GFS Master Server** functions as the coordinator for the cluster. The operation log maintains a record of the cluster's actions. Furthermore, it maintains a record of the information that describes chunks, often known as metadata. The metadata to the master server indicates the position of the chunks within the overall file and the files to which they belong.
- **GFS Chunk Servers:** These servers are the primary components responsible for performing the majority of the work in the Google File System (GFS). They store file portions that are 64 MB in size. The master server does not receive any data segments from the chunk servers. Instead, they transmit the desired pieces directly to the client. The GFS replicates each chunk multiple times and distributes them over different

chunk servers to ensure stability. The default number of copies is three. Each duplicate is designated as a singular entity.

### Features of GFS

- Namespace management and locking.
- Fault tolerance.
- Reduced client and master interaction because of large chunk server size.
- High availability.
- Critical data replication.
- Automatic and efficient data recovery.
- High aggregate throughput.

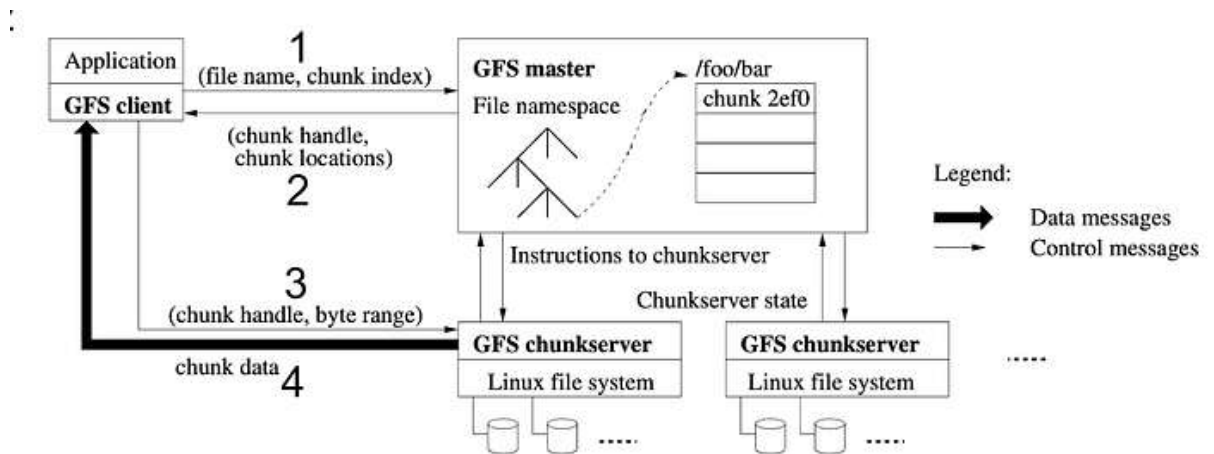


Figure 4.12: GFS Architecture

### Advantages of GFS

1. Enhanced availability Even if a few nodes fail, the data remains accessible. Replication Component failures are prevalent, as the adage suggests.
2. High volume of data processing. Multiple nodes functioning simultaneously.
3. Reliable storage. Corrupted data can be identified and replicated.

### Disadvantages of GFS

1. Not suitable for small files.
2. The master may serve as a bottleneck.
3. Incapable of typing randomly.
4. Appropriate for methods or data that are written once and subsequently only read (appended).

## b. HDFS

### HDFS Architecture

HDFS is specifically engineered to possess exceptional scalability, dependability, and efficiency, hence facilitating the storing and manipulation of colossal datasets. The architecture has multiple essential components:

1. NameNode
2. DataNode
3. Secondary NameNode
4. HDFS Client
5. Block Structure

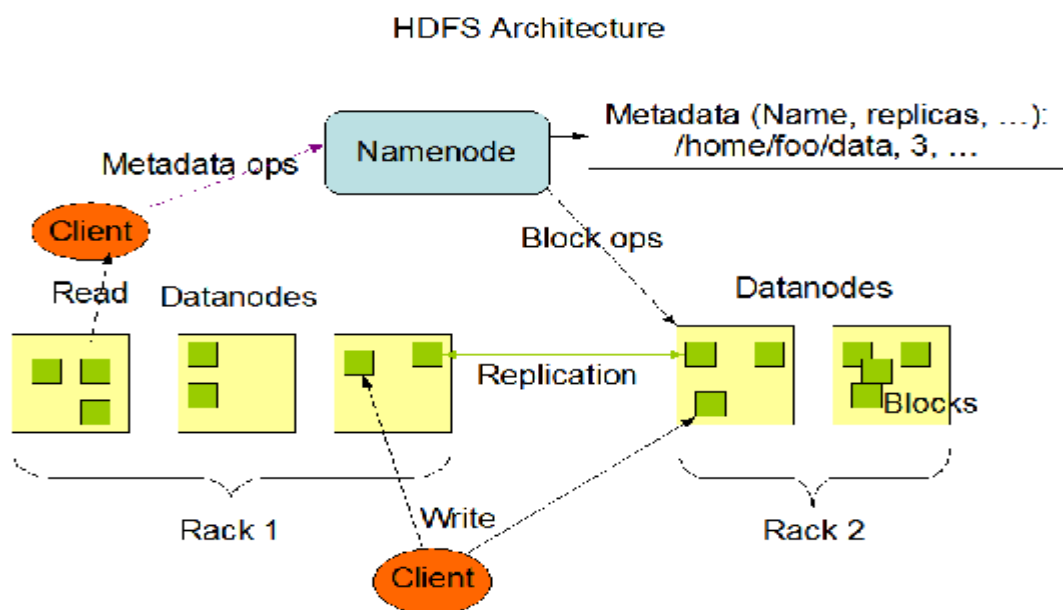


Figure 4.13: HDFS Architecture

### NameNode

The NameNode serves as the primary server responsible for overseeing the filesystem namespace and regulating clients' access to files. It executes functions like as initiating, terminating, and altering the names of files and directories. In addition, the NameNode assigns file blocks to DataNodes, while also managing the metadata and the overall organization of the file system. The metadata is maintained in memory to provide quick retrieval and is also saved on disk to ensure reliability.

**Key Responsibilities:**

- Managing the structure and information about files and directories in the filesystem.
- Overseeing the assignment of file blocks to DataNodes.
- Verifying the accuracy and consistency of data and managing the duplication of data blocks.

**DataNode**

DataNodes are the operational nodes in HDFS that are accountable for saving and retrieving real data blocks according to the instructions given by the NameNode. Every DataNode is responsible for managing the storage devices connected to it and regularly provides the NameNode with a report containing the list of blocks it stores.

**Key Responsibilities:**

- The process of storing data blocks and fulfilling read/write requests from clients.
- Executing block creation, deletion, and replication as directed by the NameNode.
- Regularly transmitting block reports and heartbeats to the NameNode to verify its operational state.

**Secondary NameNode**

The Secondary NameNode functions as an auxiliary to the primary NameNode, principally tasked with merging the EditLogs with the current filesystem image (FsImage) in order to alleviate the potential burden on the NameNode. The purpose of creating checkpoints in the namespace is to guarantee that the filesystem metadata remains current and can be restored in the event of a NameNode failure.

**Key Responsibilities:**

- Combining EditLogs with FsImage to generate a fresh checkpoint.
- Assisting in the management of the namespace metadata of the NameNode.

**HDFS Client**

The HDFS client serves as the medium for users and applications to engage with the HDFS. It enables the execution of file creation, deletion, reading, and writing actions.

The client establishes communication with the NameNode to ascertain the DataNodes that store the blocks of a file. It then directly interacts with the DataNodes to perform read and write operations on the actual data.

**Key Responsibilities:**

- Enabling communication between the user/application and HDFS.
- Establishing communication with the NameNode to retrieve metadata and with DataNodes to access data.

**Block Structure**

HDFS partitions files into sizable blocks, usually ranging from 128MB to 256MB in magnitude. Blocks are kept separately across several DataNodes, enabling parallel processing and ensuring resilience to failures. The NameNode maintains a record of the block locations and their duplicates.

**Key Features:**

- A larger block size minimizes the burden of handling a substantial quantity of blocks.
- Blocks are duplicated across numerous DataNodes to provide data accessibility and resilience against faults.

**HDFS Advantages**

HDFS provides numerous benefits that establish it as the ideal option for handling extensive datasets in distributed computing environments:

**Scalability**

HDFS is extremely scalable, enabling the storing and processing of petabytes of data across a large number of devices. The system is engineered to efficiently manage a growing number of nodes and storage capacity without experiencing noticeable performance decline.

**Key Aspects:**

- Linear scalability enables the incorporation of new nodes into the system without the need to reconfigure the entire system.

- It supports horizontal scaling by adding more DataNodes.

### **Fault Tolerance**

HDFS guarantees robustness and resilience by employing data replication to maintain high availability. Data replication is implemented by distributing each block of data among many DataNodes, hence guaranteeing data accessibility in the event of node failures.

#### **Key Features:**

- Automatic block replication guarantees the duplication of data to ensure redundancy.
- The ability to configure the replication factor allows administrators to optimize the balance between storage efficiency and fault tolerance.

### **High Throughput**

HDFS is designed to efficiently handle the retrieval of big datasets, making it well-suited for applications that need processing massive amounts of data. It enables concurrent processing of data across numerous nodes, greatly enhancing the speed of data read and write operations.

#### **Key Features:**

- Enables efficient handling of extensive data transfers and batch processing.
- Designed to maximize efficiency in accessing sequential data, resulting in reduced seek times and improved throughput.

### **Cost-Effective**

HDFS is specifically engineered to operate on inexpensive hardware, resulting in a substantial decrease in the expenses associated with establishing and managing a storage system of considerable size. The fact that it is open-source also decreases the overall cost of ownership.

**Key Features:**

- Employs cost-effective hardware, minimizing capital expenditure.
- Utilizes open-source software, eliminating licensing fees.

**Data Locality**

HDFS optimizes data processing by leveraging data locality, which involves executing computations in close proximity to the storage location of the data. This optimizes data transmission across the network, decreasing latency and enhancing overall system efficiency.

**Key Features:**

- Data-aware scheduling guarantees that tasks are allocated to nodes where the data is located.
- Enhances network efficiency by minimizing congestion and optimizing processing speed.

**Reliability and Robustness**

HDFS is designed to effectively manage hardware problems. The NameNode and DataNodes are engineered to effectively recover from failures without compromising data integrity, and the system consistently monitors the well-being of nodes to avert any potential data loss.

**Key Features:**

- Automated identification and restoration of node failures.
- Routine health assessments and validation of data integrity.

**HDFS Use Cases**

HDFS is extensively utilized in diverse industries and applications that necessitate the processing of data on a massive scale:

- **Big Data Analytics:** HDFS serves as a fundamental element in Hadoop-powered big data systems, facilitating the storing and examination of enormous datasets to derive valuable insights and support decision-making processes.
- Enterprises utilize HDFS for the purpose of storing and overseeing substantial

amounts of historical data, specifically for reporting and business intelligence.

- **Machine Learning:** HDFS offers a strong storage layer for machine learning frameworks, making it easier to train models on extensive datasets.
- **Log Processing:** HDFS is utilized for the storage and processing of log data originating from web servers, apps, and devices, facilitating instantaneous monitoring and analysis.
- **Media firms** utilize HDFS for content management, storing and distributing huge multimedia assets to provide optimal accessibility and availability.

#### **4.4.2 Decentralized Network**

A decentralized network is a network setup in which there are several authorities acting as a central hub for participants. In the realm of computers, a decentralized network design allocates workloads among multiple devices rather than depending on a solitary central server. Each of these individual computers functions as a miniature central unit that independently interacts with other nodes. Consequently, in the event of a master node failure or breach, the remaining servers can still offer data access to users, ensuring that the network remains operational with minimal or no interruption.

##### **A. UCCS Decentralized Network**

UCCS adopts a decentralized structure based on blockchain technology, which offers a viable way to address the problem of a single point of failure by removing the requirement for trusted intermediaries. Moreover, the blockchain's peer-to-peer structure grants all network participants equal rights to validate and verify the accuracy of data, ensuring the integrity of the system.

A decentralized network preserves a transparent and tamper-proof structure for storing the metadata of both data and users. Each function call stores the relevant information in the decentralized structure. Hence, the data flow among the users of the proposed architecture is secure and easily observable.

## B. GFS

The Google File System (GFS) does not adhere to a decentralized network concept. On the contrary, GFS is specifically built to function as a centralized distributed file system. In a centralized distributed file system such as GFS, there is a sole master server responsible for managing metadata, such as file locations and access control. Additionally, there are numerous chunk servers that store the actual file data.

The main attributes of GFS encompass:

1. **Single Master:** A master server exists to oversee file system access and handle metadata management.
2. **Chunk Servers:** Several chunk servers are responsible for storing data in the form of chunks that have a certain size.
3. **Chunk Replication:** Data chunks are duplicated across many chunk servers to ensure resilience against failures and to enhance accessibility.
4. **Client Interaction:** Clients engage with the master server for metadata operations and directly communicate with chunk servers for data reads and writes.

Decentralized networks, in contrast, disperse control and data management among numerous nodes without a central locus of authority. IPFS (InterPlanetary File System) and BitTorrent are two examples of decentralized file systems.

GFS is a centralized distributed file system that is managed by a single master server, responsible for handling metadata, and numerous chunk servers, responsible for storing data. This differs from the decentralized nature of other network and file system architectures.

## C. HDFS

The Hadoop Distributed File System (HDFS), the main storage system utilized by Apache Hadoop, does not exactly adhere to a decentralized network paradigm. On the contrary, HDFS is specifically designed with a distributed architecture that can be described as follows:

1. **Master-Slave Architecture:** HDFS follows a master-slave architecture, consisting of two primary node types:

- **NameNode:** is the primary node responsible for overseeing the file system's namespace and metadata, including the data block locations on DataNodes.
  - **DataNode:** These are subordinate nodes responsible for storing the factual data blocks of files.
2. **Centralized NameNode:** The NameNode is a critical weak spot in a standard HDFS implementation. The metadata for the entire file system is stored in it, which includes information about the allocation of data blocks to specific DataNodes.
  3. **Data Replication:** HDFS utilizes data replication to guarantee both fault tolerance and high availability. Data blocks are duplicated across several DataNodes throughout the cluster. The replication factor can be adjusted and is commonly set to three.
  4. **Data Accessibility:** Clients have the ability to directly retrieve data from DataNodes, which contain replicas of the data blocks. Nevertheless, it is necessary for them to establish communication with the NameNode in order to obtain information regarding the structure of the file system and the positions of blocks.
  5. **Decentralized Storage and Processing:** Although HDFS has a central control or administration component called the NameNode, the processing frameworks like Apache MapReduce or Apache Spark that are developed on top of HDFS can be considered as decentralized processing. This occurs because processing jobs, such as MapReduce or Spark tasks, are divided throughout the cluster and carried out simultaneously on the DataNodes where the data is stored.

Essentially, HDFS adheres to a distributed storage approach, utilizing a centralized NameNode to handle metadata management. This is in contrast to a completely decentralized network model, which lacks a one point of control or coordination.

#### 4.4.3 UCCS's encryption technique:

This is a comparison of the UCCS encryption technology with the GFS and HDFS encryption techniques.

1. The Ultra Compression Cloud Storage (UCCS) employs the Cipher-Policy

- Attribute-Based Encryption (CP-ABE) scheme using the RSA encryption algorithm.
2. The design of GFS does not include built-in encryption.
  3. HDFS does not inherently have encryption as a built-in functionality.

#### 4.4.4 UCCS compression technique.

The comparison between the UCCS compression technique and the compression techniques employed in GFS and HDFS is as follows:

1. UCCS employs Huffman coding as its compression technique.
2. The primary feature of GFS does not include direct file compression.
3. HDFS supports various compression algorithms including GZip, BZip2, Snappy, LZO, and Zstandard, Deflate to decrease in volume or size.

Gzip is a software program used for compressing and decompressing files. It employs the DEFLATE algorithm, which is a combination of LZ77 (Lempel–Ziv 1977) and Huffman coding. DEFLATE employs a technique where it substitutes duplicate strings with references to prior instances and use Huffman coding to condense the lengths of recurring strings.

The parameters must be checked in order to compare two compression algorithms are as follows.

1. Compression ratio
2. Saving percentage
3. Compression Factor

| File<br>Format | LZO |       | Huffman |       | Deflate |       | LZ77 |       | Gzip |       |
|----------------|-----|-------|---------|-------|---------|-------|------|-------|------|-------|
|                | CT  | CR    | CT      | CR    | CT      | CR    | CT   | CR    | CT   | CR    |
| <b>AIFF</b>    | 82  | 2.47  | 3541    | 19.86 | 3380    | 19.67 | 1023 | 2.87  | 3440 | 19.80 |
| <b>WAV</b>     | 94  | 2.06  | 3900    | 23.37 | 3743    | 23.17 | 1017 | 2.63  | 3849 | 23.11 |
| <b>WMA</b>     | 87  | 38.04 | 2941    | 39.3  | 2338    | 39.12 | 743  | 36.49 | 2530 | 39.2  |
| <b>BMP</b>     | 249 | 24.4  | 3135    | 45.89 | 2583    | 44.52 | 855  | 25.7  | 2782 | 44.92 |

|            |     |       |      |       |      |       |      |       |      |       |
|------------|-----|-------|------|-------|------|-------|------|-------|------|-------|
| <b>TIF</b> | 218 | 22.68 | 2577 | 39.94 | 2750 | 39.01 | 917  | 22.06 | 2812 | 39.86 |
| <b>TXT</b> | 57  | 94.18 | 1393 | 85.44 | 844  | 84.88 | 580  | 43.06 | 856  | 84.99 |
| <b>DCM</b> | 56  | 55.97 | 1237 | 62.33 | 978  | 60.56 | 207  | 58.72 | 1279 | 61.09 |
| <b>AVI</b> | 60  | 6.64  | 3387 | 6.94  | 3176 | 6.94  | 1063 | 5.74  | 3206 | 6.94  |

Table 4.7: Compression Ratio of Different File Formats

The file undergoes compression using a compression algorithm, and the compression ratio for each algorithm is determined. The results are subsequently recorded. These data are applicable for analyzing the compression algorithms employed on various files. Based on the aforementioned conclusion, it can be observed that the compression technique employed in UCCS is superior to that of HDFS.

#### 4.4.5 UCCS Resource Optimization Technique

The comparison between the resource optimization technique used in UCCS and the resource optimization techniques employed in GFS and HDFS is as follows:

1. UCCS employs the HoneyBee resource optimization approach.
2. GFS employs many resource optimization strategies such as data replication, chunking, a Master-Slave architecture, efficient data access, and garbage collection.
3. HDFS essentially enhances resource utilization by employing data replication and data localization.

#### Features of HoneyBee

1. It is designed to replicate the foraging behavior of honeybees, specifically their quest for food and communication methods.
2. This method is employed to address optimization problems, particularly in search spaces, with the aim of identifying optimal or nearly optimal solutions.
3. Valuable for dynamically improving the allocation of resources based on real-time data and changes in the environment.
4. Achieves a balance between exploration and exploitation, resulting in efficient resource optimization.
5. The algorithm has the ability to be adjusted for use in large-scale issues, although its

performance may differ depending on the complexity of the problem and the fine-tuning of its parameters.

**Features of Data Replication:**

1. The main objective is to guarantee the availability and dependability of data by storing multiple copies of data across many nodes.
2. Safeguard against data loss caused by node failures, guaranteeing the ability to recover data even in the event of certain nodes becoming inoperative.
3. Having several copies can enhance read speed by enabling load balancing and reducing access latency.
4. The presence of many copies results in storage overhead, which can cause inefficiencies in the utilization of storage resources if not adequately handled.
5. Implementing and managing it is simpler compared to intricate algorithms such as HoneyBee.

The aforementioned comparison leads to the following conclusion:

1. Honeybee is better suited for dynamic optimization problems, whereas data replication is primarily focused on enhancing reliability and performance in data storage systems.
2. Honeybees possess the ability to adjust to limited resources in a flexible manner, whereas data replication might result in inefficiencies if the replication factor is not tuned.
3. The optimization parameters of Honeybee necessitate adjustment and comprehension, while data replication procedures are often uncomplicated but can contribute additional workload.

Ultimately, the decision between Honeybee and data replication hinges on the unique demands of the application, such as the preference for dynamic optimization versus the importance of reliability and performance.

Due to the greater requirement for dynamic optimization in the UCCS framework, the honeybee optimization method has been used for resource optimization. Another justification for choosing HoneyBee algorithms is that UCCS is primarily intended for the management of storage space. Therefore, UCCS recommends the policy of avoiding

data replication due to the resulting increase in storage expense. The procedure for preventing data replication with UCCS is as follows:

- When the data owner uploads a file, retrieve the metadata of the file from the decentralized network.
- If the file already exists, retrieve the user details associated with the file.
- Verifying the integrity of the file if the file name and owner name are the same.
- If the integrity verification fails, it indicates that the file may have been modified.
- Enable the data owner to upload the file onto distributed storage.
- If the file name is identical but the owner name does not match.
- The file has been replicated.
- Create the URL for the file.

4. Store the file URL and user metadata on a decentralized network.

By following the aforementioned methods, UCCS is able to prevent file replication and effectively handle storage overhead.

#### **4.4.6 UCCS Integrity Verification Technique:**

UCCS employs the Markel root idea, which is based on blockchain technology, to ensure the integrity of data. The user has the ability to request an integrity check for a file from the data owner. The data owner retrieves the user and file metadata from a decentralized network and proceeds with the verification process. The UCCS obtains the cached Merkle root of the requested file, recalculates the Merkle root of the file, and performs the comparison. If the results are matched, then the integrity is validated; otherwise, it returns false.

#### **Merkle Root Tree**

The Merkle tree employs cryptographic hash algorithms to construct a binary tree structure by merging the leaf nodes to generate intermediate nodes and the root node, as depicted in Figure 4.14. The transaction data in the blockchain is partitioned into various pieces, which are stored as leaf nodes at the lowest level of the tree structure. Intermediate nodes are then generated by merging the hash values of the child nodes.

Ultimately, the Merkle root merges its two offspring nodes and produces the highest node of the tree. The Merkle root is a crucial component in the validation of data stored in leaf nodes as hash values. It takes into account all transaction signatures that are included in the block inside the blockchain network. The Merkle structure is characterized by its efficient use of storage space and its ease of computation.

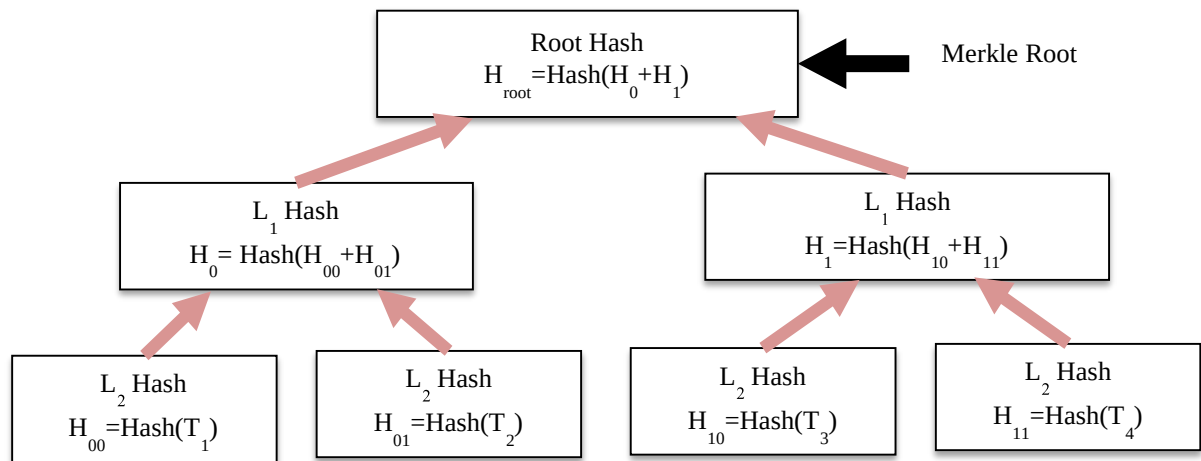


Figure 4.14: Structure of the Merkle Tree

GFS utilizes checksums to guarantee the integrity of data stored within the system. Integrity verification in GFS operates as follows:

- 1. Checksums:** Every data chunk saved in GFS is linked to a checksum. A checksum is a concise data fragment obtained by applying a cryptographic hash function to the original data chunk. The checksum is put in close proximity to the data block.
- 2. Verification:** When a client accesses data from GFS, it simultaneously obtains the checksum linked to that data chunk. Subsequently, the client calculates the checksum of the received data and verifies its equivalence with the stored checksum. If the calculated checksum corresponds to the stored checksum, it signifies that the data is unaltered and has not suffered any corruption.
- 3. Detection and Recovery:** In the event that the checksums do not correspond during a read operation, it signifies that the data might have been compromised during its transmission or storage. In such instances, the Google File System (GFS)

can employ its replication mechanism to obtain an unaltered duplicate of the data from another replica, guaranteeing the integrity and dependability of the data.

GFS utilizes integrity verification through checksums to ensure data consistency and stability, even in the presence of hardware failures, network problems, or data corruption. Ensuring data integrity across various servers and data centers is a fundamental concern in large-scale distributed file systems such as GFS, making this method vital.

**HDFS:** It ensures integrity verification mostly through the use of checksums, as opposed to traditional cryptographic methods used in encryption.

Integrity verification in HDFS operates as follows:

**1. Checksums:** HDFS employs checksums to authenticate the integrity of data contained within the file system. Every data block stored in HDFS is accompanied with a checksum, often implemented using CRC32, which is stored independently. When a client retrieves data from HDFS, it also reads and verifies the checksum. If the calculated checksum of the read data matches the stored checksum, it verifies that the data has not undergone any corruption during storage or transmission. If the checksums do not correspond, it signifies data corruption.

**2. Block Verification:** HDFS does regular integrity checks on the blocks stored on DataNodes, which are machines in the Hadoop cluster responsible for storing data blocks. This aids in identifying any corruption that may have arisen as a result of hardware malfunctions or other problems.

**3. Replication and Data Recovery:** HDFS ensures data replication and recovery by storing several copies of each data block on various DataNodes. In the event that a corrupted block is identified during the process of reading or verifying blocks, HDFS has the capability to substitute the corrupted block with a functional replica from a different DataNode, hence guaranteeing the dependability and soundness of the data.

Although HDFS prioritizes data integrity by employing checksums and block verification, it lacks the capability to perform end-to-end cryptographic integrity verification, such as digital signatures or cryptographic hashes commonly found in

some storage systems or protocols. On the contrary, its methodology is specifically crafted to effectively identify and restore data damage occurring in the distributed storage system of a Hadoop cluster.

| Aspect               | Checksum                                   | Markel Root                                                           |
|----------------------|--------------------------------------------|-----------------------------------------------------------------------|
| Purpose              | Validate data integrity                    | Ensure data integrity and authentication in structure.                |
| Definition           | A small size value derived from data.      | A hash that summarized all transactions in a block                    |
| Structure            | Simple hash of data                        | Tree structure (Market tree) with hashes of child nodes               |
| Applications         | Networking, file integrity checks, storage | Blockchain, version control system, peer-to-peer network.             |
| Error Detection      | Detects error or corruption                | Validates data integrity in large data sets                           |
| Efficiency           | Fast computation, low overhead             | More complex due to tree structure, but efficient for large datasets. |
| Output               | A single fixed-size checksum value         | A root hash representing multiple data blocks.                        |
| Common Algorithm     | CRC, MD5, SHA                              | SHA-256, SHA-3 for hashing in Markel trees.                           |
| Granularity          | Typically checks small data blocks         | Can efficiently check large data sets or blocks of transaction.       |
| Structure Complexity | Simple linear structure                    | Hierarchical tree structure                                           |

Table 4.8: Comparison of Checksum and Markel Root

Therefore, the final outcome is as follows:

**Checksum:** is predominantly utilized for rapid mistake detection and is easily calculable, rendering it appropriate for smaller datasets.

**Markel Root:** is a tool that guarantees the accuracy and consistency of data in huge datasets, particularly in blockchain technology. It condenses several transactions and facilitates rapid verification.

#### **4.4.7 UCCS Multifactor Authentication Technique:**

The registration process in UCCS is specifically designed to enroll data owners and users in order to establish an authentication mechanism, which includes features such as loginID, password, TOTP, and biometrics for multifactor authentication.

The Google File System (GFS), a distributed file system developed by Google, largely use Access Control Lists (ACLs) for authentication and authorization purposes.

Access Control Lists (ACLs) enable the system to define the authorized users or groups who have permission to access or edit files and folders. Authentication of this kind is essential in distributed systems such as GFS to efficiently handle rights across numerous nodes and users.

In addition, GFS utilizes OAuth 2.0 for user authentication when accessing resources via APIs, guaranteeing secure access to file system data over the network.

These technologies aid in preserving security and regulating access control within the infrastructure of the Google File System.

**HDFS**, also known as the Hadoop Distributed File System, largely employs Kerberos authentication to ensure the security of its services. Kerberos is a network authentication protocol that use secret-key cryptography to offer robust authentication for client/server applications.

Here's how Kerberos authentication works in HDFS:

1. **Ticket-based Authentication:** When a user desires to retrieve data stored in HDFS, they verify their identity to the Kerberos Key Distribution Center (KDC) by providing their credentials (usually a username and password). After the user is successfully authenticated, the Key Distribution Center (KDC) provides the user with a ticket-granting ticket (TGT).

2. **Service Tickets:** The user submits the TGT to the HDFS NameNode or other Hadoop services, such as the ResourceManager for YARN, in order to seek access. The NameNode authenticates the user's identity by validating the Ticket Granting Ticket (TGT) with the Key Distribution Center (KDC) and provides a service ticket for accessing the desired Hadoop Distributed File System (HDFS) data or resources.
3. **Mutual Authentication:** Both the client (user) and the HDFS service authenticate each other using these tickets, guaranteeing that only authorized users with valid tickets can access HDFS data.

The utilization of Kerberos authentication in HDFS offers numerous security advantages:

- **Robust Authentication:** Users undergo authentication through tickets provided by a trusted Key Distribution Center (KDC), guaranteeing that only authorized users can gain access to HDFS data.
- **Authentication Protection:** Kerberos employs encryption to protect the tickets that are shared between clients and services, ensuring that unauthorized individuals cannot intercept or tamper with them.
- **Single Sign-On (SSO):** After authentication, users can access different Hadoop services without the need to enter their credentials again, as long as their tickets are valid.

HDFS offers various authentication techniques, including LDAP, alongside Kerberos. However, Kerberos is the preferred and generally adopted authentication method because to its strong security features.

| Aspect  | MFA                                                                    | OAuth2.0                                                                       | Kerberos                                                    |
|---------|------------------------------------------------------------------------|--------------------------------------------------------------------------------|-------------------------------------------------------------|
| Purpose | Enhance security by requiring multiple verification factors for access | Delegates access permissions between applications without sharing credentials. | Provide secure authentication over a network using tickets. |
| Type    | Authentication method                                                  | Authorization framework                                                        | Authentication Protocol                                     |

|                              |                                                                |                                                                     |                                                                                    |
|------------------------------|----------------------------------------------------------------|---------------------------------------------------------------------|------------------------------------------------------------------------------------|
| Components                   | Something you know, have, or are                               | Access tokens, refresh tokens, scopes                               | Tickets, Ticket Granting Ticket(TGT)                                               |
| User Interaction             | Requires user involvement for every login (or occasionally)    | User consent for granting access to applications                    | Transparent to users after initial login                                           |
| Use Cases                    | Securing user accounts, applications and systems               | Third-party applications accessing user data.                       | Secure access in enterprise networks                                               |
| Security Level               | High, especially when using multiple factors                   | Moderate, relies on secure token handling                           | High, uses symmetric encryption and tickets                                        |
| Implementation Complexity    | Moderate, requires additional setup and user education         | Moderate, involves client-server communication and token management | Complex, requires a dedicated KDC (Key Distribution Center)                        |
| Scalability                  | Generally scalable, but can be cumbersome for large user bases | Highly scalable, widely used for web and mobile applications        | Scalable within enterprise environments but can be complex in distributed systems. |
| Token Expiration             | Not typically involved in token management                     | Access token expire refresh tokens are used for renewal             | Tickets have a set lifespan and can be renewed.                                    |
| Common protocols / standards | NA                                                             | OAuth 2.0 specification, OpenID connect                             | Kerberos protocol (RFC 4120)                                                       |

Table 4.9: Comparison of MFA, OAuth2.0 and Kerberos

In summary,

**MFA:** system improves security by necessitating numerous verification elements and primarily focuses on authenticating the user.

**OAuth 2.0:** is an authorization framework that enables programs to access user resources without the need to provide credentials. It is extensively utilized for third-party services.

**Kerberos:** is a network authentication protocol specifically created for ensuring safe communication within enterprise contexts. It utilizes tickets to authenticate users.

#### **4.4.8 Access Control Policy used in UCCS**

UCCS initially verifies if users have the ability to retrieve the data based on the access rules posted on the decentralized network. If the users possess privileges, indicating that the attributes they share are a subset of the stored attribute list, then the requested file will be transformed into plaintext after undergoing decryption and decompression. Otherwise, the request will be rejected.

UCCS employs ABAC, which stands for Attribute Based Access Control Policy.

**GFS:** The Google File System (GFS) primarily adapts an **Access Control List (ACL)-based access control policy**.

Access Control Lists (ACLs) are utilized to administer authorizations for files and directories within the Google File System. Access Control Lists (ACLs) enable the precise allocation of permissions, such as read, write, and execute, to particular users or groups. This mechanism effectively regulates the access and modification of files and directories.

ACL-based access control in GFS operates in the following manner:

**1. Permissions Management:** Every file and directory in GFS is assigned an Access Control List (ACL) that determines the specific operations (such as read, write, execute) that can be performed by individual users or groups.

**2. Granular Control:** ACLs provide precise control over access permissions, allowing for granular management. For instance, various individuals or groups may possess distinct levels of authorization to access the identical file or directory.

**3. Effective access control:** is achieved through the use of Access Control Lists (ACLs), which ensure that only individuals or groups with proper authorization can access sensitive data or carry out certain operations within the file system.

This strategy is efficient in distributed file systems such as GFS because it enables administrators to centrally manage access permissions and guarantees security across numerous nodes and users who access the system.

**HDFS:** The Hadoop Distributed File System (HDFS), specifically built for storing vast quantities of data across numerous devices, has the subsequent access control policy:

### **Access Control Lists (ACLs)**

The access control policy employed by the Hadoop Distributed File System (HDFS), which is specifically built for the storage of vast quantities of data across numerous devices, is as follows: Access Control Lists (ACLs) in HDFS enable administrators to establish precise and detailed permissions for files and directories. This involves setting the individual users and groups that have the ability to conduct certain activities, such as reading, writing, or executing, on particular files or directories. Access Control Lists (ACLs) offer greater flexibility compared to standard Unix-style permissions, which are also partially supported by HDFS. ACLs provide more comprehensive control over access rights.

To summarize, HDFS predominantly use Access Control Lists (ACLs) as its modified access control policy for overseeing file and directory permissions.

Below is a comparison between ABAC and ACL.

### **1. Definition:**

**ABAC:** Access control decisions are made based on the attributes of persons, systems, and resources. It assesses regulations and protocols to ascertain whether a user should be given authorization.

**ACL:** A permission list is a collection of permissions associated with an entity. The purpose of this is to determine the specific people or system processes that are given permission to access and the specific sort of access that is permitted.

## **2.Scalability:**

**ABAC:** It has excellent scalability as it is capable of efficiently managing a substantial quantity of attributes and policies. As the environment evolves and undergoes changes, it has the ability to adjust by assimilating new characteristics and regulations.

**ACL:** As the number of users and resources increases, managing can get increasingly challenging. Each resource usually necessitates its own Access Control List (ACL), resulting in increased administrative burden.

## **3.Granularity:**

**ABAC:** Provides a more detailed level of analysis by taking into account several attributes when making access decisions. This level of flexibility enables a more detailed and precise management of access to resources based on certain criteria.

**ACL:** The granularity of access control in this context is generally lower, as it focuses on specific items and defines access permissions for individual persons or groups. This reduces the flexibility for intricate access scenarios.

## **4.Flexibility:**

**ABAC:** Is more suited for dynamic contexts where access control requirements may alter often due to its increased flexibility. Modifications or additions to policies can be made without the need to directly edit specific resources.

**ACL:** It lacks flexibility since making changes generally necessitates updating each individual Access Control List (ACL) linked to specific resources.

## **5.Use Cases:**

**ABAC:** Is appropriate for settings that require access control decisions to be determined based on many attributes, including user roles, location, device type, and time of access. It is frequently utilized in cloud computing settings and dynamic IT architectures.

**ACL:** is commonly employed in less complex settings or situations where basic access control is acceptable. File systems and network devices often incorporate access control mechanisms to regulate the permissions for reading, writing, and executing certain files or directories.

To summarize, ABAC offers greater flexibility and scalability in comparison to ACLs, thanks to its policy-based methodology and consideration of numerous attributes.

Nevertheless, ACL continues to be effective in less complex situations when direct control over access rights is sufficient.

#### **4.4.9 Security Technique for storage used in UCCS**

UCCS guarantees the security of storage by implementing many principles, such as confidentiality, integrity, access control, authentication, and encryption. The system utilizes the CP-ABE algorithm to guarantee the presence of confidentiality, integrity, and availability functionalities. Furthermore, it ensures confidentiality by employing encryption techniques to encode user data. The integrity is ensured through the utilization of the blockchain-based Merkle root concept.

**GFS:** The Google File System (GFS) incorporates various fundamental principles to guarantee the security and dependability of stored data:

1. **Replication and Distribution:** The GFS system utilizes a distributed approach to store data, dividing it into smaller chunks and replicating these chunks across several computers. This duplication aids in guaranteeing the accessibility of data even in the event of server failures or data corruption.
2. **Checksums and Data Integrity:** GFS employs checksums to ensure the integrity of data. GFS uses checksums to identify and manage data corruption or mistakes that may occur with each data chunk.
3. **Access Control:** The GFS system utilizes access control features to limit the ability of anyone to read from or write to particular files or directories. This guarantees that only individuals or software programs with proper authorization can retrieve confidential information.
4. **Authentication and Authorization:** Users and programs that want to access GFS need to verify their identity, usually by using Google's authentication procedures. Authorization systems guarantee that users possess the necessary permissions to carry out operations on files and folders.
5. **Encryption:** Google is anticipated to use encryption mechanisms to safeguard data stored and transmitted within the GFS infrastructure, however particular information about this is not publicly released. This guarantees protection in the event of unauthorized interception or improper access to data, it will remain incomprehensible without the appropriate decryption keys.

6. **Monitoring and Auditing:** GFS is equipped with monitoring and auditing features to oversee data access and trace actions conducted on the file system. This aids in the identification of unauthorized entry or questionable behaviors.

The security design of Google File System incorporates redundancy, access control, encryption, and monitoring to safeguard stored data from illegal access, corruption, and loss. These safeguards are implemented as part of a comprehensive strategy to ensure the security and accessibility of data within Google's infrastructure.

**HDFS:** Hadoop Distributed File System (HDFS) ensures security through several mechanisms:

1. **Authentication:** HDFS provides support for authentication techniques, including Kerberos. Prior to accessing the HDFS cluster, both users and services are required to verify themselves.
2. **Authorization:** HDFS use Access Control Lists (ACLs) and file permissions, like to conventional file systems, to govern the authorization of file and directory access. This guarantees that only authorized users and applications have the ability to access and modify certain files.
3. **Encryption:** HDFS provides the capability to encrypt data both when it is stored and when it is being transferred. One way to protect data that is not actively being used is to encrypt it using techniques such as AES (Advanced Encryption Standard). This assures that even if the physical storage is accessed by unauthorized individuals, the data will still be kept secure. Protocols such as TLS/SSL can be utilized to encrypt data while it is being transmitted between nodes in the cluster.
4. **Auditing:** HDFS offers auditing features to monitor and record file access and updates. Audit logs serve the purpose of monitoring and investigating security issues.
5. **Firewalls and Network Security:** Administrators have the ability to customize firewalls and network configurations in order to limit access to HDFS nodes and regulate communication between clients and nodes.
6. **Secure Cluster Communication:** Secure communication between nodes in the HDFS cluster can be achieved by using protocols such as SASL (Simple Authentication and Security Layer). This ensures that eavesdropping and manipulation are prevented.

7. **Integration with Security Ecosystems:** HDFS can seamlessly connect with wider security ecosystems with its support for authentication systems such as LDAP (Lightweight Directory Access Protocol) and Kerberos. Additionally, it has the capability to integrate with Apache Ranger for the implementation of centralized permission policies.

In summary, HDFS offers strong security measures to safeguard data stored in the Hadoop ecosystem from unwanted access, guaranteeing the confidentiality, integrity, and availability of the data.

The table below illustrates a comparison between UCCS, GFS, and HDFS in terms of several security considerations.

| Security Aspects | UCCS | GFS | HDFS |
|------------------|------|-----|------|
| Distribution     | ✓    | ✓   | ✓    |
| Replica          | ×    | ✓   | ✓    |
| Checksum         | ×    | ✓   | ✓    |
| Data Integrity   | ✓    | ✓   | ✓    |
| Access Control   | ✓    | ✓   | ✓    |
| Authentication   | ✓    | ✓   | ✓    |
| Authorization    | ✓    | ✓   | ✓    |
| Encryption       | ✓    | ✓   | ✓    |

Table 4.10: Comparison of security aspect of UCCS, GFS and HDFS

## 4.5 Deploying UCCS Server on VMWare

VMware Workstation Player is a very suitable tool for operating a solitary virtual machine on a Windows or Linux personal computer. A virtual machine (VM) is a software entity that executes programs or applications independently of any actual hardware. A virtual machine (VM) instance allows for the execution of one or several guest machines on a host computer. Each virtual machine (VM) operates independently with its own operating system, functioning autonomously from other VMs, even if they are hosted on the same physical server.

Virtual machines (VMs) are fundamental components of virtualized computing resources and are essential for producing applications, tools, or environments. Virtual machines (VMs) serve as the fundamental building blocks for establishing a cloud infrastructure in conjunction with an on-premises setup. Deploying virtual machines

(VMs) to replicate systems in cloud environments can offer an additional level of protection and assurance.

The five most popular types of virtualizations include:

- **Desktop.** Enable concurrent execution of many virtual machines to host cloud-based desktops on a shared physical or virtual server.
- **Application.** Generates a virtual representation of essential business application instances, hence avoiding the necessity for installing app software on local operating systems.
- **Server.** Enables the substitution of a physical server with a virtual one, facilitating server administration via the cloud.
- **Storage.** Utilizes secure cloud storage to store the enterprise's data, eliminating the requirement for physical data storage and potentially decreasing costs related to data center space.
- **Network.** Utilizes both physical and virtual components to establish a hybrid network, enabling network management using software switches.

The model utilizes a combination of storage and network virtualization. Virtual storage space utilizes cloud technology to store data, eliminating the need for physical storage space. By storing the data remotely, it has the capability to get information from any virtual storage sources and organize them according to certain preferences, allowing for customization of the storage of important data based on specific criteria. Network virtualization decouples the virtual network from the underlying physical hardware, enabling the control of network traffic and resource management. Network virtualization is essential for the development of new software. By doing rigorous testing of our model on a virtual network prior to its public release, we can easily identify any potential faults and defects, and optimize the performance of our model depending on its performance in a virtual environment.

## 4.6 System Performance

In order to assess the effectiveness of a UCCS (Ultra Compression Cloud Storage) solution, we will establish and measure the performance of an Ethereum-based system.

This system will include a cloud storage application that will be deployed on VMware Workstation 12, which will be running Ubuntu 18.04 LTS.

## 1. Environment Setup

**VMware Workstation 12:** Host machine running VMware Workstation 12.

**Ubuntu 18.04 LTS VM:**

**CPU:** 4 cores

**RAM:** 8 GB

**Disk:** 100 GB

## 2. Components

**Blockchain Platform:** Ethereum

**Cloud Storage:** Nextcloud

**Benchmarking Tools:** AB (Apache Benchmark) for web requests, fio for disk I/O, ethperf for Ethereum performance

## 3. Installation and Configuration

### a. Install Ethereum Client (Geth):

```
#bash
```

Copy code

```
sudo add-apt-repository -y ppa:ethereum/ethereum
```

```
sudo apt-get update
```

```
sudo apt-get install geth
```

### a. Install Nextcloud: Follow the Nextcloud installation guide for Ubuntu.

```
#bash
```

Copy code

```
sudo apt update
```

```
sudo apt install apache2 php libapache2-mod-php mysql-  
server php-mysql
```

```
sudo systemctl start apache2
```

```
sudo systemctl start mysql
```

### **Download and configure Nextcloud:**

```
#bash
```

Copy code

```
wget
```

```
https://download.nextcloud.com/server/releases/nextcloud-25.0.0.zip
```

```
unzip nextcloud-25.0.0.zip -d /var/www/html/
```

- b. **Integrate Ethereum with Nextcloud:** Create a basic smart contract to store metadata of files and deploy it using Geth. Enable seamless integration between Nextcloud and this smart contract through an API.

## **4. Performance Testing**

### **a. Blockchain Performance:**

**Setup:** Start a private Ethereum network or use a local testnet.

**Benchmarking:** Use `ethperf` or similar tools to measure transaction throughput and latency.

### **Sample commands for testing transaction throughput:**

```
#bash
```

Copy code

```
geth --networkid 1234 --datadir /path/to/your/data --rpc -  
-rpcapi="eth,web3,personal" console
```

Employ a script to dispatch several transactions and quantify the throughput in terms of transactions per second.

### **b. Cloud Storage Performance:**

**i. Disk I/O Performance:** Use `fio` to test disk read/write speeds:

```
#bash
```

Copy code

```
fio --name=readtest --ioengine=sync --rw=read --bs=4k --size=1G --numjobs=4 --time_based --runtime=1m
```

**ii. Web Requests Performance:** Use ab to test Nextcloud's handling of multiple file upload/download requests:

```
#bash
```

Copy code

```
ab -n 100 -c 10 http://localhost/nextcloud/index.php
```

### **c. Integrated System Performance:**

**i. Data Storage:** Upload and store files in Nextcloud, then use the Ethereum smart contract to record metadata.

**ii. Measure Interaction Latency:** Measure the duration required for a file upload to finish and for metadata to be documented on the blockchain.

## **5. Results and Analysis**

### **a. Blockchain Metrics:**

- **Transaction Throughput:** Number of transactions per second.
- **Latency:** Time taken for a transaction to be confirmed.

### **b. Cloud Storage Metrics:**

- **Disk Throughput:** Speed of disk reads and writes.
- **Web Response Time:** Latency for handling multiple file operations.

### **c. Integrated System Metrics:**

- **End-to-End Latency:** The duration between the commencement of a file upload and the recording of its metadata on the blockchain.
- **Impact on Cloud Storage:** The impact of integrating blockchain on the performance of cloud storage.

## **6. Optimization**

**Based on the results:**

- **Blockchain Configuration:** Modify the configuration of the Ethereum client or enhance the efficiency of interactions with smart contracts.
- **Storage Configuration:** Enhance the performance of Nextcloud or the storage settings to efficiently manage the additional workload caused by the blockchain.

## 4.7 Comparative Analysis

This section presents a comparison between the proposed architecture and advanced approaches like block secure, secure access control, and deduplication with blockchain. An assessment and comparison of the many components of the proposed architecture are conducted in relation to the existing body of work. The network performance of the proposed architecture is evaluated by comparing it to the block secure. The efficiency of the suggested architecture access control method is assessed and compared to the previous work. Additionally, the storage performance of the proposed architecture is evaluated and contrasted with the performance of deduplication with blockchain. Three distinct literary works are chosen for comparative study in order to evaluate the varied characteristics of the proposed design utilizing diverse measurement factors. Initially, the suggested architecture is evaluated in relation to the literature, specifically in terms of transmission time. This evaluation takes into account two distinct architectures: one with the implementation of an optimization method and one without. Subsequently, the performance of the suggested architecture is assessed by altering the quantity of user attributes in order to gauge the runtime of transactions. This is then juxtaposed with the safe access control system. Finally, the performance of the suggested architecture is evaluated by measuring the encoding and decoding time and comparing it with a deduplication design based on blockchain. This comparison demonstrates the importance of the proposed architecture in relation to existing architectures.

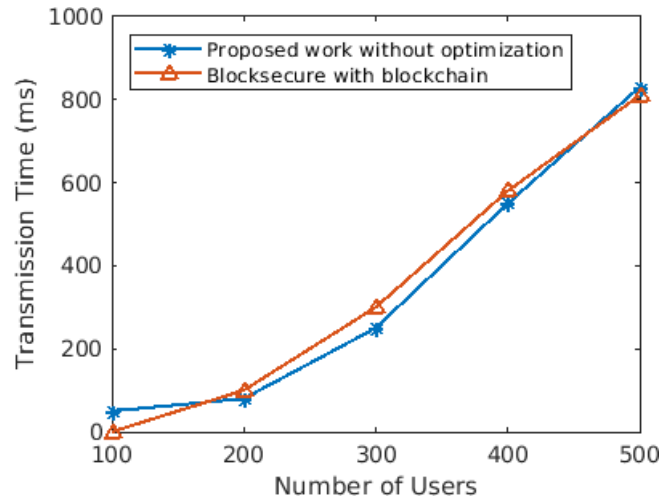


Figure 4.15: Transmission Time Comparison between Proposed Work without Optimization and Block Secure

The block secure architecture has two distinct architectures to store user data in the cloud. There are two types of architectures being discussed: one involves blockchain with numerous data centers, while the other involves multiple data centers with genetic optimization. Based on the information presented in Figures 4.15 and 4.16, the proposed architecture is compared to block secure designs using transmission time as the parameter. Transmission time refers to the duration needed to store and retrieve files from the cloud. The suggested architecture, when not optimized, requires the same amount of transmission time as the blockchain-based solution without optimization. This is achieved by adjusting the number of users, as shown in Figure 4.15. Therefore, the suggested architecture preserves the efficiency of the current work. As depicted in Figure 4.16, the suggested architecture, which incorporates optimization implementation, is being compared to a block secure design with genetic implementation. The suggested architecture offers reduced transmission time for storing and accessing files from the cloud database compared to block secure. The block secure architecture necessitates additional computational time in genetic algorithm, while the proposed architecture requires less analysis. Therefore, the performance of the proposed architecture surpasses that of block secure designs in terms of transmission time.

The performance of the proposed architecture is evaluated by comparing it to the existing literature on secure cloud storage with access control, using runtime as the metric. The performance of the proposed architecture is evaluated by measuring the runtime of access transactions while modifying the amount of attributes of the users.

According to Figure 4.17, the implementation of the suggested architecture for executing access transactions takes less time compared to secure access control. As the number of attributes rises, the execution time of both designs also increases. The implementation of straightforward access controls in the suggested architecture reduces the execution time in comparison to the existing literature.

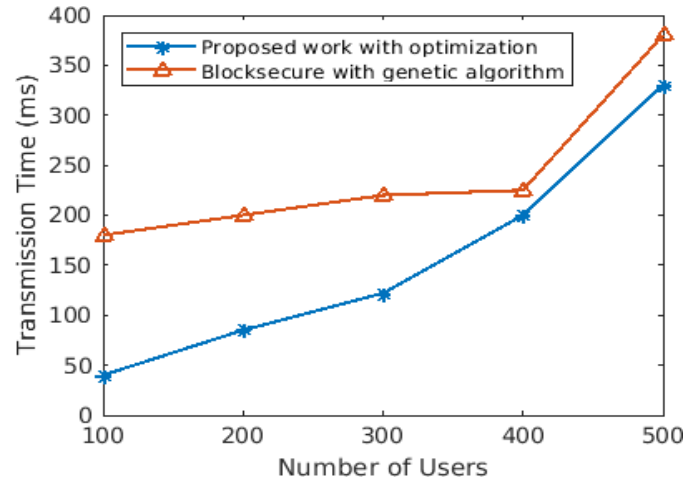


Figure 4.16: Transmission Time Comparison between Proposed Work with Optimization and Block Secure

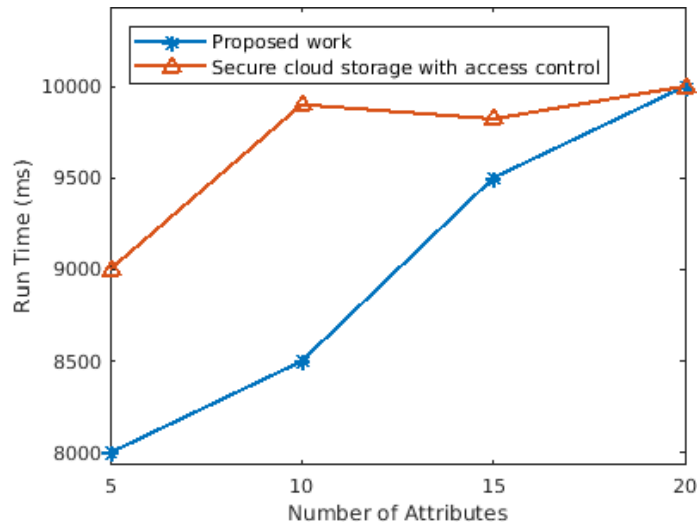


Figure 4.17: Run-Time Comparison of Proposed Architecture with the Secure Access Control

The encoding and decoding time of the suggested design are compared with the existing literature. Based on the information shown in Figures 4.18 and 4.19, the suggested design necessitates comparable amounts of time for both encryption and decryption, when compared to the deduplication

architecture based on blockchain. The decoding time for both architectures is shorter than the encoding time. As illustrated in the pictures, the suggested architecture and literature require an equal amount of time for both operations. Hence, the performance evaluation and comparative analysis demonstrate that the suggested design upholds the efficacy of the current work and offers an enhanced and secure solution.

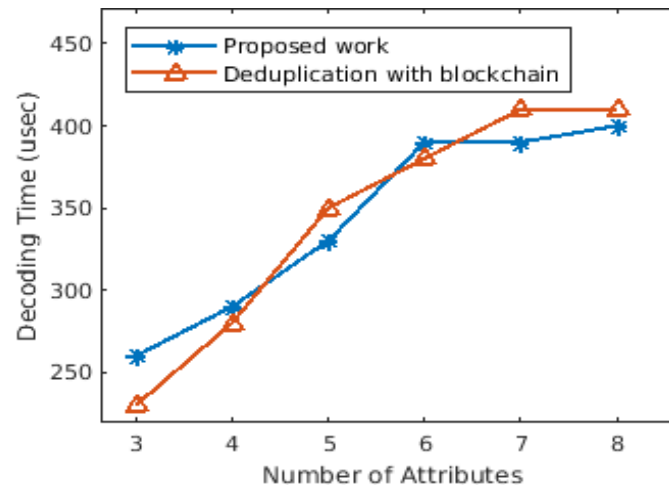


Figure 4.18: Encoding Time Comparison between Proposed Work and Deduplication with Blockchain

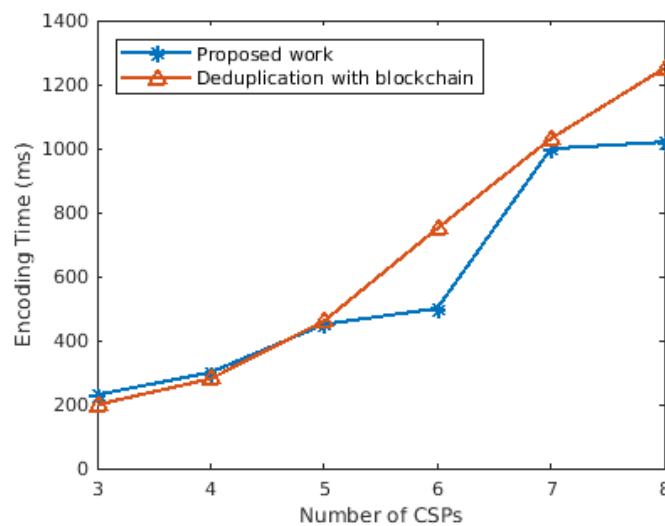


Figure 4.19: Decoding Time Comparison between Proposed Work and Deduplication with Blockchain

## **4.8 Distributive Execution of UCCS Server**

In the past ten years, Cloud storage has emerged, offering enhanced Information security. Ensuring the security of information is a significant concern in cloud computing systems. Clouds are not constrained by boundaries, allowing data to be stored in any data center across a geographically spread network. The behavior of cloud computing gives rise to significant concerns surrounding user authentication, information integrity, and secrecy. Therefore, it is suggested to use an improved, innovative, and robust security framework. To enhance information security and guarantee confidentiality, integrity, and authentication during data storage, access, and peer interactions in data centers. Given the current research situation, we have developed a framework that integrates different security preservation strategies.

Therefore, in order to create an efficient solution, it is necessary to distribute the processing. This research paper suggests using Apache Spark as a distribution framework for machine learning-based cryptographic compression. We utilized VMWare Workstation as our execution platform. Blockchain is utilized to offer an unchangeable collection of files or data to all participating nodes in the security system of cloud storage. A comparative study is being done to compare the performance of non-distributive analysis with distributive analysis.

### **4.8.1 Introduction**

Organizations across many sectors and of all sizes are utilizing cloud computing for a diverse range of purposes, including but not limited to data backup, disaster recovery, virtual desktops, software development and testing, big data analytics, and customer-facing web applications.

Cloud computing facilitates the access of data and cloud applications from remote physical servers, databases, and computers over the internet by client devices. Although there are certain advantages to depending on cloud services, cloud computing also presents its own set of difficulties. The primary obstacle confronting cloud computing is commonly acknowledged to be security. Organizations confront the potential dangers of data breaches, API and interface hacking, compromised passwords, and

authentication concerns when they depend on the cloud. Moreover, there is a dearth of openness concerning the methods and locations in which the cloud provider manages sensitive information that has been entrusted to them. Ensuring security requires meticulous consideration of cloud setups and adherence to organizational policy and practice.

Ensuring cloud security requires distinct protocols compared to traditional IT settings. Several recommended strategies for enhancing cloud security include:

**1. Shared responsibility for security:** In general, the cloud provider is accountable for ensuring the security of the cloud infrastructure, while the user is responsible for safeguarding their data within the cloud. However, it is crucial to have a clear definition of data ownership.

**2. Data encryption:** Data should be encrypted while storage, during transmission, and during active processing. Users must retain complete authority over security keys and hardware security modules.

**3. User identity and access management:** Users want comprehensive comprehension and clear insight of network, device, application, and data accessibility.

**4. Collaborative management:** Effective communication and well-defined procedures between the provider and user are essential for achieving smooth and secure cloud integrations that are also sustainable in the long term.

**5. Security and compliance monitoring:** Start by comprehending the regulatory compliance criteria that are relevant to your company. Then, establish a system for actively monitoring all connected systems and cloud-based services to ensure continuous visibility of all data transfers.

It is important to be aware that popular free cloud storage services such as Google Drive and Onedrive may have restrictions when it comes to following the recommended guidelines.

#### 4.8.2 Proposed Distributive UCCS Server

Our suggested framework will encompass all security protocols, ensuring the secure storage and retrieval of data. We are examining compression methods, encryption schemes, splitting, hashing, and their implementation on VMware. The advent of decentralized currency has been a remarkable recent breakthrough, which has given rise to the emergence of the bitcoin system. Bitcoins are commendable, but their greatness is enhanced by the underlying technology on which they operate. The technology, called blockchain, has other applications beyond decentralized cash. Here, we utilize this principle to leverage the collective capacity of all the systems in the network, resulting in a significant increase in processing power at a minimal expense. This benefits everyone in the network.

The importance of our work is substantial, both currently and in the future. Recently, users have been utilizing cloud storage services such as GoogleDrive and OneDrive. Next, we will utilize storage drives and security services. We must be prepared for them, and our efforts ensure our readiness.

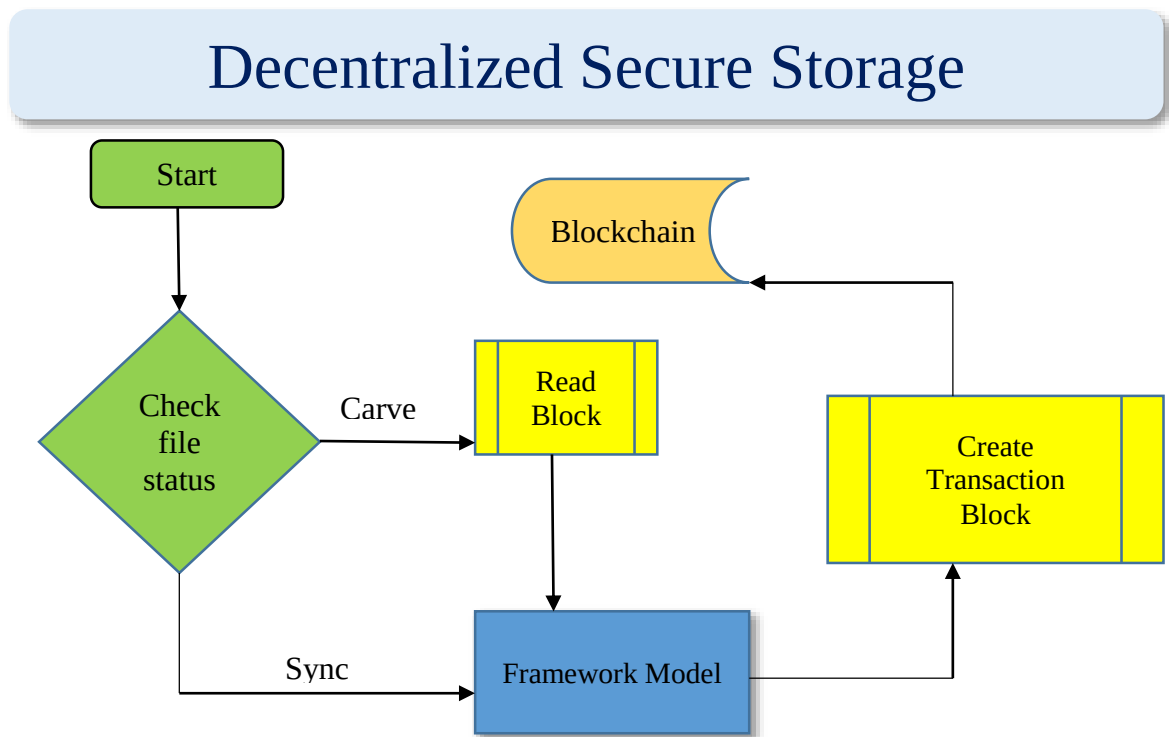


Figure 4.20: Proposed framework for Decentralized Secure Storage.

Let's comprehend the functioning of the framework. Upon the arrival of a file to be tested, it is necessary to verify the file's status. We verify if the user intends to upload or download the material. When the user wants to upload a file, it is processed by our framework to ensure its security and then added to the blockchain. However, if the user desires to download the file, it is necessary to retrieve and transfer the block from the blockchain to the framework in order to obtain the original file.

The system we employ utilizes a range of compression and encryption methods, including Huffman, Coding and RSA. Therefore, a substantial level of computational capacity and data storage is necessary to operate this model. Thus, executing on a solitary node is an inefficient resolution.

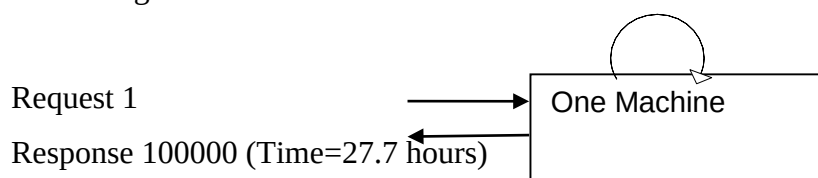
The Amazon AWS Cloud services are utilized to execute the framework in a distributed context. The Apache Spark framework is utilized to distribute the workload among the nodes that are involved. Blockchain stores information about files on separate nodes.

### 4.8.3 Centralized versus Distributive Computing

Utilizing distributed computing can result in significant performance advantages. For illustrative purposes, let us consider an example. Assume that we have a total of 10,000 requests that need to be fulfilled. The average duration for processing each individual request is around 1 second. Therefore, after the first response, the last response will come after  $100000/3600 = 27.7$  hours. However, if we have 1000 nodes working in parallel, the same response is going to come in  $100000/6000 = 1.6$  minutes. This is a great performance benefit.

#### Centralized System

Processing time=1 sec



#### Distributive System

Processing time=1 sec

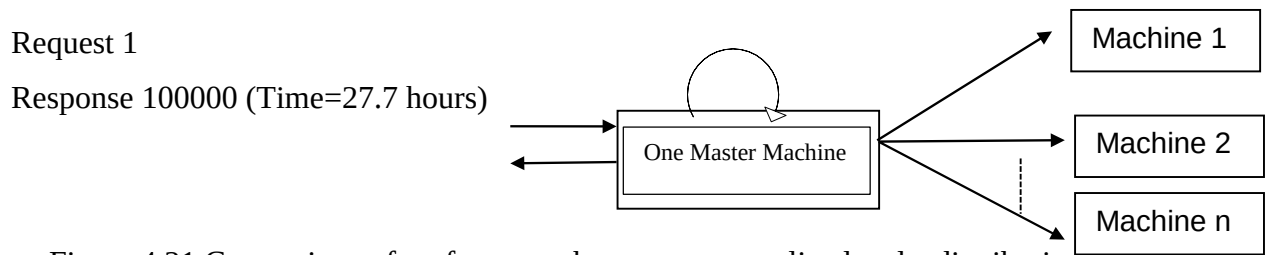


Figure 4.21 Comparison of performance between a centralized and a distributive system

#### 4.8.4 Role of Apache Spark in UCCS Server

Apache Spark is a framework utilized for the purpose of cluster or distributed computing. The framework is open source. As our model will be executed on all interconnected nodes, Spark will enable them to run concurrently and provide fault-tolerance as well. Thus, in the event of a node failure, the system will not experience a breakdown and will continue to function effectively. Spark provides users with high speed, user-friendly interface, and advanced analytics capabilities.

In the field of distributed computing, various technologies have been developed, such as Hadoop, MapReduce, and Storm. Hadoop typically operates in batch processing mode and is not designed to handle real-time data. Apache Spark provides real-time data analysis in a distributed setting. Another benefit of Spark is the provision of a cohesive framework for handling diverse datasets, such as text datasets and graph datasets. Additionally, it is capable of processing data from both real-time and batch sources.

Spark supports application development in multiple languages including as Java, Scala, and Python. In addition to Map Reduce, the system also supports SQL queries, streaming data, machine learning, and graph data processing. DataBricks conducted an experiment in October 2014, where they achieved a new world record by sorting 100 TB of data using Apache Spark on 207 EC2 virtual servers in just 23 minutes. Apache Spark is compatible with several platforms such as Hadoop, Mesos, Kubernetes, standalone environments, and cloud infrastructures. It has the capability to function with a variety of data sources like as HDFS, Cassandra, HBase, and S3.

After doing initial testing on a single node, it is necessary to scale up and test the model on thousands of nodes due to the vast amount of data, amounting to petabytes, that will

be processed. For our testing purposes, we have utilized an AWS M3 RAM medium instance, which has a capacity of 3.75 GB of RAM, a single-core CPU, and a 410 GB SSD. Our data has been saved in HDFS, a system specifically built for extensive storage.

**4.8.4.1 AWS M3-** M3 is an instance of AWS. M3 is a second generation, general purpose EC2 instance type. We are using a medium M3 instance.

**4.8.4.2 M3 medium-** The device has a Random Access Memory (RAM) capacity of 3.75 Gibibytes (GiB) and utilizes a Solid State Drive (SSD) with a cache size of 1X4 Gigabytes (GB). M3 instances are equipped with Intel Xeon E5-2670 processors, which have a high frequency and can be either Sandy Bridge or Ivy Bridge architecture. The M3 medium is a versatile family of instances designed for various purposes. It is equipped with a single virtual central processing unit (vCPU).

**4.8.4.3 Amazon EC2-** EC2 is an elastic cloud computing service provided by Amazon Web Services (AWS) that allows users to rent virtual servers, known as instances, on-demand. This is a web service that provides secure and scalable computing resources on the cloud. Its purpose is to simplify web-scale cloud computing for developers. Service instances can be easily added or deleted in a matter of minutes to adjust the scale of computing resources according to changing needs.

Amazon EC2 offers users a diverse selection of instance types to accommodate their specific requirements. Instance types offer a range of options for CPU, memory, storage, and networking capabilities, allowing users to select the optimal combination of resources. Each instance type offers multiple instance sizes to efficiently adjust your resource allocation based on your workload.

**4.8.4.4 SSD - Solid State Drives (SSDs).** These devices provide excellent random input/output (IO) performance. A solid-state drive (SSD) is utilized as a form of non-volatile memory for the long-term storing of data. SSD utilizes interfaces that are compatible with standard IO and Hard Disk Drives, facilitating straightforward

replacement in numerous applications. Our model utilizes 410 gigabytes of solid-state drive (SSD) storage.

**4.8.4.5 HDFS-** The Hadoop Distributed File System (HDFS). This system is meant to be fault-tolerant and can be deployed on inexpensive hardware. It is particularly well-suited for handling huge datasets. Given the large amount of data in petabytes that our model will process, HDFS is an appropriate choice. It has the capability to expand to accommodate hundreds of nodes within a single cluster. Hadoop is a data warehousing solution that relies on MapReduce for data processing.

Spark is specifically engineered to facilitate the reading and writing of data to and from HDFS, as well as alternative storage systems such as HBase and Amazon's S3.

**4.8.4.5.1 HDFS Architecture-** HDFS utilizes a master-slave architecture. A Hadoop Distributed File System (HDFS) cluster consists of a single NameNode, which serves as the master server responsible for controlling the namespace of the file system. It additionally controls the clients' access to files. Additionally, there are many DataNodes, with one assigned to each node in the cluster, responsible for storing data. The file is divided into one or more segments, known as blocks, which are then stored in DataNodes. The NameNode oversees the operations of opening, shutting, and renaming files and folders. Additionally, it provides information regarding the allocation of blocks to DataNodes. The DataNodes handle the task of reading and writing data from the clients. The DataNodes are also responsible for block generation, deletion, and block replication.

**4.8.4.5.2 Non-Computational Data Locality-** We require co-located computing and data storage within a single node. Within the realm of databases, we encountered the concepts of computational and data locality. Essentially, this implies that we had information stored on one device and the computational tasks were carried out on another device. Typically, data is distributed among numerous workstations while the server handles the processing. This leads to a substantial volume of data transmission. Therefore, classical MapReduce is ineffective in facilitating efficient data sharing. Essentially, Hadoop is employed for the purpose of data storage. It is ineffective for analysis. Spark significantly enhances the velocity of real-time analysis.

**4.8.4.5.3 Data and processing on each machine-** Each machine in a Hadoop cluster contains both the HDFS data and the MapReduce system. There are two benefits to this approach: firstly, it adds redundancy in the system. Consequently, the level of fault-tolerance rises. Furthermore, the coexistence of data and data processing software on the same machine results in an enhanced speed of information retrieval.

Machine1 Data

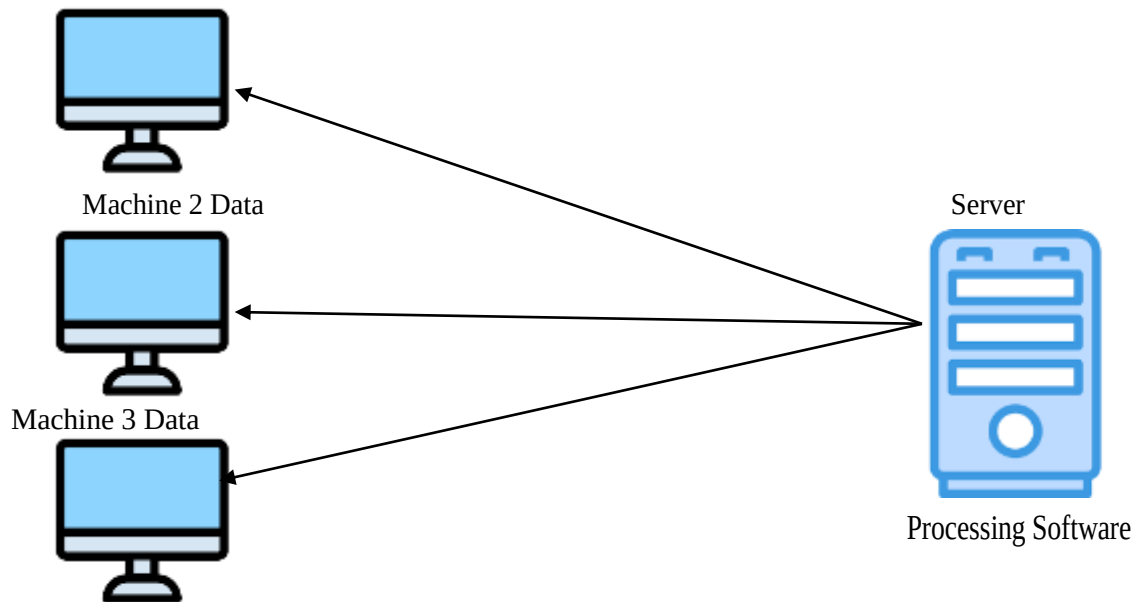


Figure 4.22: Traditional Database Processing Structure

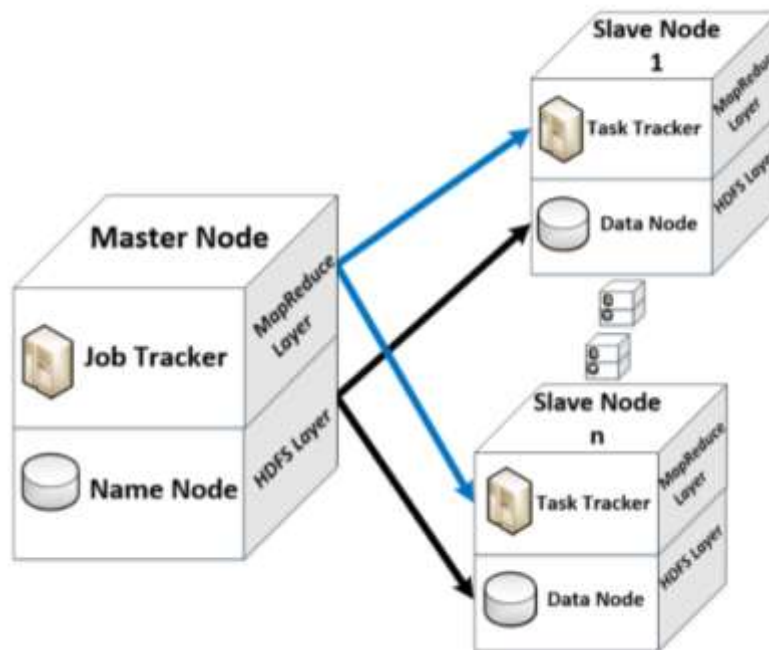


Figure 4.23 Hadoop Cluster

### 4.8.5 Working of the Hadoop Cluster

The Hadoop cluster operates in a modular manner, with the workload being distributed between master and slave nodes. The procedure is outlined in detail as follows:

- 1) Start.
- 2) The client makes a request for a file.
- 3) FileTracker receives the request.
- 4) The NameNode, responsible for storing metadata, keeps track of which DataNodes hold the blocks that comprise a particular file.
- 5) The client directly retrieves the blocks from the particular DataNodes.
- 6) FileTracker efficiently assigns the Map and Reduce jobs to the appropriate TaskTrackers in a manner that takes into account the rack configuration.
- 7) FileTracker detects any jobs that have failed and require rescheduling on an alternative URLTracker.
- 8) The URLTracker creates Java Virtual Machines (JVMs) to execute Map and Reduce jobs and receive updates from the JVMs.
- 9) Cease once the process has finished.

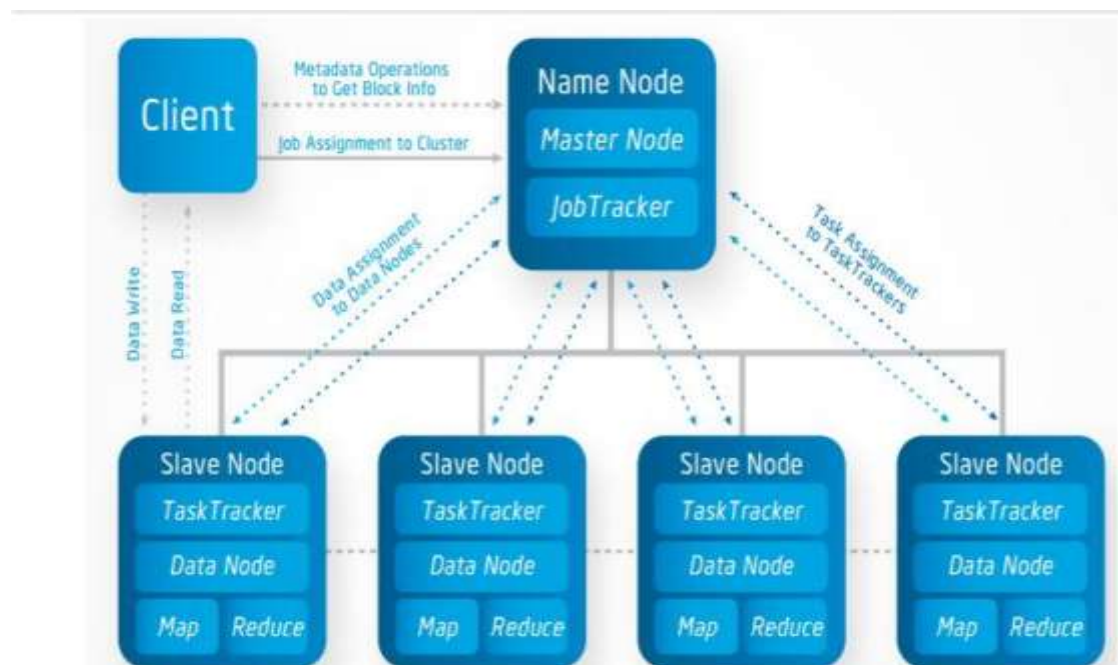


Figure 4.24: Hadoop Master-Slave Architecture

# CHAPTER 5

## FINDINGS AND CONCLUSIONS

Once the framework has been implemented successfully, it will undergo testing using about 20 files. The Excel document containing the comprehensive outcome will be sent, encompassing a dataset comprising 20 files.

### 5.1 Execution for uploading a sample file on the :

Consider a file named 'Sample.txt'.

#### 5.1.1 Compressing the file

Prioritize compressing the file. Produces the outcome, as the compression yields a file that is approximately 50% smaller in size than the original file.

On the command prompt write:

```
$ python Menu.py
```

Display the UCCS menu. The File menu contains two options: Store and Retrieve. When you choose the store option, a window appears asking if you are a registered user. If the answer is affirmative, then the program will display the login window. Otherwise, it will prompt the registration window. Only registered users have the ability to store or retrieve data. This is the initial stage of the authentication process.

Upon successful login, a filedialog is displayed prompting the user to choose the desired file name for storage. Upon selecting the filename, the compression algorithm will be applied to it. The compression techniques, used is Huffman coding.

Upon compression, the resulting file will be named 'Sample.bin' and will have a size that is about 50% smaller than the original file.

The table below gives the result on applying Huffman Coding algorithm on 20 different files.

| Sr. No. | Name of File      | Size of file (KB) | Size of file after compression (KB) | Compression Ratio | Saving Percentage | Compression Factor |
|---------|-------------------|-------------------|-------------------------------------|-------------------|-------------------|--------------------|
| 1       | Actpendin.txt     | 678               | 412                                 | 60.76696          | 39.23303          | 1.64563            |
| 2       | file_example_AVI  | 1447              | 613                                 | 42.3635           | 57.6364           | 2.3605             |
| 3       | Boc3.txt          | 771               | 461                                 | 59.79247          | 40.20752          | 1.67245            |
| 4       | Sampledocs_xml    | 51005             | 51005                               | 100               | 0                 | 1                  |
| 5       | File_example_JPEG | 2446              | 2444                                | 99.9182           | 0.08176           | 1.0008             |
| 6       | File_example_MP3  | 5196              | 5112                                | 98.3833           | 1.6166            | 1.0164             |
| 7       | Sampledocs_pdf    | 102789            | 102789                              | 100               | 0                 | 1                  |
| 8       | File_example_MP4  | 17422             | 17284                               | 99.2078           | 0.7921            | 1.0079             |
| 9       | NEPEC.txt         | 1009              | 618                                 | 61.24876          | 38.75123          | 1.63268            |
| 10      | File_example_PNG  | 2013              | 2013                                | 100               | 0                 | 1                  |
| 11      | Property.txt      | 2                 | 1                                   | 50                | 50                | 2                  |
| 12      | File_example_DOC  | 1003              | 995                                 | 99.2023           | 0.7976            | 1.0080             |
| 13      | README.txt        | 3                 | 1.20                                | 40                | 60                | 2.5                |
| 14      | File_exmaple_DOCX | 1003              | 1003                                | 100               | 0                 | 1                  |
| 15      | Revocation.txt    | 191               | 107                                 | 56.02094          | 43.97905          | 1.78504            |
| 16      | File_example_RTF  | 1010              | 532                                 | 52.6732           | 42.3267           | 1.8984             |
| 17      | Statsgo.txt       | 63                | 39                                  | 61.90476          | 38.09523          | 1.61538            |
| 18      | Carrier.txt       | 1842              | 914                                 | 41.61997          | 50.38002          | 2.01531            |
| 19      | Zopener.txt       | 3                 | 1.55                                | 51.66666          | 48.33333          | 1.93548            |
| 20      | Inhist.txt        | 4706              | 2840                                | 60.34849          | 39.65150          | 1.65704            |

Table 5.1 Table showing the result of Huffman Coding Algorithm

### 5.1.2 Key Generation and Encryption of the file

The file owner requests the generation of a key. The framework performs the key

creation function and produces the public and master key for the file owner. The produced master key is utilized to encrypt the file. The file is encrypted using the 'RSA Algorithm'. Upon encryption, the file will be transformed into an unreadable format. The file 'Sample.bin' is compressed and subsequently encrypted using the RSA technique. The output is an encrypted file named 'encrypted\_sample.bin'.

Table show the file after compression, encryption and decryption

| <b>Sr. No.</b> | <b>Name of File</b>    | <b>File after Compression</b> | <b>File after Encryption</b>    | <b>File after Decryption</b>    |
|----------------|------------------------|-------------------------------|---------------------------------|---------------------------------|
| 1              | Actpendin.txt          | Actpendin.bin                 | Encrypted_Actpendin.bin         | Decrypted_Actpendin.bin         |
| 2              | file_example_AVI.avi   | File_exmple_AVI.bin           | Encrypted_File_exmple_AVI.bin   | Decrypted_File_exmple_AVI.bin   |
| 3              | Boc3.txt               | Boc3.bin                      | Encrypted_Boc3.bin              | Decrypted_Boc3.bin              |
| 4              | File_example_JPEG.jpg  | File_example_JPEG.bin         | Encrypted_File_example_JPEG.bin | Decrypted_File_example_JPEG.bin |
| 5              | Carrier.txt            | Carrier.bin                   | Encrypted_Carrier.bin           | Decrypted_Carrier.bin           |
| 6              | File_example_MP3.mp3   | File_example_MP3.bin          | Encrypted_File_example_MP3.bin  | Decrypted_File_example_MP3.bin  |
| 7              | Inhist.txt             | Inhist.bin                    | Encrypted_Inhist.bin            | Decrypted_Inhist.bin            |
| 8              | File_example_MP4.mp4   | File_example_MP4.bin          | Encrypted_File_example_MP4.bin  | Decrypted_File_example_MP4.bin  |
| 9              | NEPEC.txt              | NEPEC.bin                     | Encrypted_NEPEC.bin             | Decrypted_NEPEC.bin             |
| 10             | File_example_PNG.png   | File_example_PNG.bin          | Encrypted_File_example_PNG.bin  | Decrypted_File_example_PNG.bin  |
| 11             | Property.txt           | Property.bin                  | Encrypted_Property.bin          | Decrypted_Property.bin          |
| 12             | File_example_DOC.doc   | File_example_DOC.bin          | Encrypted_File_example_DOC.bin  | Decrypted_File_example_DOC.bin  |
| 13             | README.txt             | README.bin                    | Encrypted_README.bin            | Decrypted_README.bin            |
| 14             | File_exmaple_DOCX.docx | File_exmaple_DOCX.bin         | Encrypted_File_exmaple_DOCX.bin | Decrypted_File_exmaple_DOCX.bin |
| 15             | Revocation.txt         | Revocation.bin                | Encrypted_Revocation.bin        | Decrypted_Revocation.bin        |
| 16             | File_example_RTF.rtf   | File_example_RTF.bin          | Encrypted_File_example_RTF.bin  | Decrypted_File_example_RTF.bin  |
| 17             | Statsgo.txt            | Statsgo.bin                   | Encrypted_Statsgo.bin           | Decrypted_Statsgo.bin           |
| 18             | Sampledocs_xml.xls     | Sampledocs_xml.bin            | Encrypted_Sampledocs_xml.bin    | Decrypted_Sampledocs_xml.bin    |
| 19             | Zopener.txt            | Zopener.bin                   | Encrypted_Zopener.bin           | Decrypted_Zopener.bin           |

|    |                    |                    |                              |                              |
|----|--------------------|--------------------|------------------------------|------------------------------|
| 20 | Sampledocs_pdf.pdf | Sampledocs_pdf.bin | Encrypted_Sampledocs_pdf.bin | Decrypted_Sampledocs_pdf.bin |
|----|--------------------|--------------------|------------------------------|------------------------------|

Table 5.2: File name after compression, encryption and decryption

We employ the RSA algorithm, which is a type of cryptographic technique that uses asymmetric key encryption. It employs a pair of keys, namely a public key and a private key. The level of security provided by an encryption algorithm is directly influenced by the size of its key. The greater the size of the key, the more difficult it becomes to decipher the encryption technique. The framework utilizes a key size of between 1024 to 2048 bits for the RSA algorithm.

Prior to encrypting the file, it is necessary to produce both the public and private keys. Subsequently, the file must undergo encryption. The keys are exchanged with the user using the Diffie-Hellman algorithm.

### 5.1.3 Check integrity of file

UCCS creates the metadata of the file, including the file hash, Merkle root, mapping details of the file and owner/user, and stores these in the decentralized network. The user verifies the integrity of the received data by transmitting a request to the data owner. The data owner retrieves the user and file information from the decentralized network, conducts the integrity verification procedure using the Merkle root technique, and then sends the outcome back to the user.

The integrity check function guarantees the integrity of the saved cloud data. Users have the ability to request an integrity check for the file from the data owner. The data owner retrieves the user and file details from the decentralized network and proceeds with the verification process. The individual who possesses the data retrieves the Merkle root that is stored for the requested file, recalculates the Merkle root of the file, and then compares the two to determine if they match. If the results are matched, then the integrity is confirmed; otherwise, it returns false.

### 5.1.4 Adding file in block

According to the framework, compressed and encrypted files are saved on distributed storage, while the meta data of the file, such as file hash, Markel root, and mapping details, are stored on a decentralized network. The file's metadata is saved on a

decentralized network in the form of a block. Data in each block is stored using the standard JSON format. The data for each block is represented in a similar manner:

```
{
  "phash": "Piece_hash",
  "file": "file_name",
  "type": "file_type",
  "size": "file_size",
  "pic": "piece_counter",
  "url": "Tracking server URL"
}
```

To incorporate this functionality in Python, begin by constructing a block class with the previously specified characteristics. Efforts have been made to ensure that each block is distinct, hence preventing any instances of repetition.

After the addition of file details in block the block looks like:

```
{ "length": 2, "chain": [{ "nonce": 0, "index": 0, "hash":
"3ad3ba5f08e0f8d15c81c0ae54bf6e9e9136d3a81d2303649beef37b9e57a3b9",
"transactions": [], "timestamp": 1637433074.778131, "previous_hash": "0" },
{ "nonce": 15, "index": 1, "hash":
"000f3fb6cadacba296685ec6cf0158eef7c8b1b4c9269e65f5615dd7468cc59c",
"transactions": [{ "url": "http://localhost:8080/gui/web/index.html", "filetype":
"text", "pic": 1, "filename": "Sample.txt", "filesize": 716664, "phash":
"63758e5ff7bd467b1793631643d5e1e30c2c683740aeb77c3ec44741bf0c472e15f808
883c0bcf30717480d8d5f07dd3771fa69df77deb6b4fad672c5e33bb8e" } ],
"timestamp": 1637433074.778207, "previous_hash":
"3ad3ba5f08e0f8d15c81c0ae54bf6e9e9136d3a81d2303649beef37b9e57a3b9" } ] }
```

### 5.1.5 Adding block in blockchain

To add a block to the blockchain, begin by creating a class specifically designed for the blockchain. To maintain the unchangeability of the entire blockchain, the current block includes the hash of the prior block. The comprehensive understanding of the data included within each block establishes a system for safeguarding the overall integrity

of the entire chain, albeit to some extent. The `previous_hash` variable was provided in the `Block` class for this reason. To initiate the blockchain, establish the `create_genesis_block` function. This generates a primary block with an index of 0 and a prior hash of 0. Subsequently, append this to the linked list that maintains a record of every block.

```
import time

class Blockchain:

    def __init__(self):
        self.unconfirmed_transactions = []
        self.chain = []
        self.create_genesis_block()

    def create_genesis_block(self):
        genesis_block = Block(0, [], time.time(), "0")
        genesis_block.hash = genesis_block.compute_hash()
        self.chain.append(genesis_block)

    @property
    def last_block(self):
        return self.chain[-1]
```

#### 5.1.6 Seedbox testing

A seedbox is a distant server with a large bandwidth capacity that is utilized for the purpose of uploading and downloading digital files from a peer-to-peer (P2P) network. Seedboxes are specifically outfitted with Bittorrent-specific software, such as `utorrent`, `qBittorrent`, `Deluge`, or other similar programs.

VMWare Workstation serves as a versatile platform for executing many tasks. Begin by installing VMware. It is advisable to utilize VMware Workstation Player 12. Utilize Ubuntu as the Operating System with VMware. Therefore, proceed with the installation of Ubuntu. It is advisable to utilize Ubuntu 18.04.06 LTS. Admin privileges are required to configure the seedbox. To execute this command on the terminal, type the following:

```
abc@ubuntu:~$ sudo -i
```

Install the seedbox with root privileges. It is advisable to utilize Utorrent server. To initiate the seedbox, execute the command.

```
root@ubuntu:~# utserver -settingspath /opt/utorrent-server-alpha-v3_3/ &
```

This provides a visually appealing web user interface that closely resembles the user interface of a personal computer. The default port for operating it will be 8080. The browser can be utilized to gain access to it. To access the website, use any web browser and enter the following URL, replacing "your IP address" with your actual IP address:

<https://localhost/gui>

A dialog box will be displayed, prompting the user to provide a username and password. Next, the User Interface is presented in the following manner:

A seedbox is equipped with one or more torrent clients that have a web user interface (UI) accessible remotely via a secure shell (SSH). The appearance of the Web User Interface is as follows:

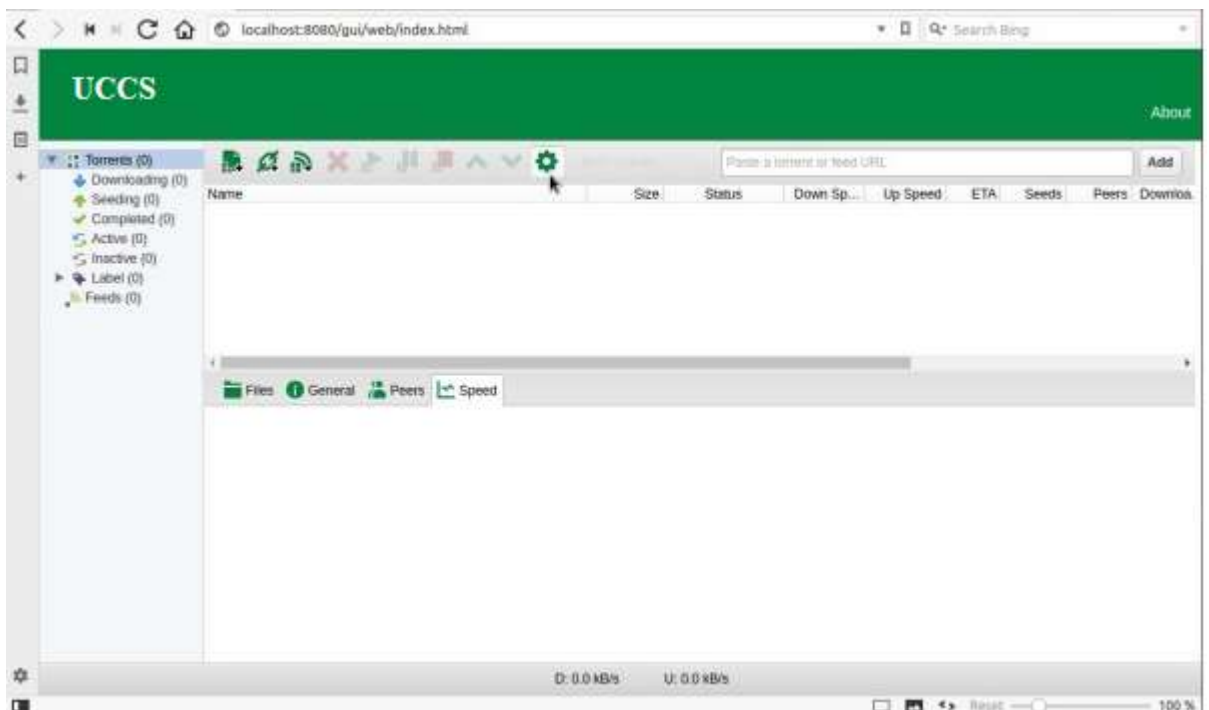


Figure 5.1: Seedbox Web User Interface

Simply input torrent files or Uniform Resource Locators (URLs) into the interface and

the download process will commence once they are loaded. Seedboxes enable the retrieval of files saved on them via HyperText Transfer Protocol (HTTP), File Transfer Protocol (FTP), or SSH FTP (SFTP).

A seedbox conceals the IP address. When attempting to identify the individual viewing a specific torrent file, only the seedbox IP address will be seen, not the user's.

Seedboxes possess a high level of security. All operations performed with a seedbox are conducted over a remote connection. You are not engaging in immediate downloading or uploading of any data to your system.

### **5.1.7 Running on VMWare**

VMware Workstation is primarily intended for the purpose of creating and overseeing many virtual machines (VMs) concurrently. It signifies that the application must be installed on the host operating system that is running on a real computer.

VMware Workstation is compatible with regular x86-based hardware that has 64-bit Intel or AMD processors. It may be installed on host operating systems that are either 64-bit Windows or Linux.

VMware Workstation facilitates the deployment of several applications and operating system workloads on a single server for companies. Consequently, it facilitates more efficient allocation of resources.

By utilizing VMware, each application may be executed in its own operating system on a single physical server, effectively optimizing the available CPU capacity of the server. VMware enables greater energy and space optimization by allowing the execution of a larger number of applications with a reduced number of physical servers.

The host VM refers to the physical machine that supplies the guest VM with computer hardware resources, including processing power, memory, disk, and network I/O.

Virtual machine software enables the execution of applications and operating systems, facilitates data storage, facilitates network connectivity, and performs various computer tasks. It necessitates regular maintenance, including upgrades and system monitoring.

Due to the aforementioned advantages of VMware, it is advisable to utilize VMware

Workstation Player12 with the Ubuntu 18.04.06 LTS operating system. Additionally, it is advised to use the Python programming language within the virtual machine framework. All components of the framework are operational inside the specified environment.

## **5.2 Execution for downloading sample file from framework:**

When executing the framework with an example file, it is readily apparent how the framework operates.

### **5.2.1 Reading Block**

Firstly, login to the framework and send request to data owner. Within the framework, blockchain functions as a ledger for storing file details in the form of blocks.

On blockchains, entries are organized into batches called "blocks." These blocks are subsequently linked together on decentralized ledgers, hence the term "blockchain".

The data owner verifies the user's data with the blockchain user metadata. If it match data owner send key generation request to UCCS. UCCS saves user details in the proposed architecture and generates access policies corresponding to its attribute list using the CP-ABE algorithm UCCS generates public and master key for user and with the help of Deffi-Helman algorithm the keys are shared with user. An Ethereum block explorer is utilized to retrieve information about the contents of a specific block. Upon input, it will display all the particulars. The user sends the access request to the corresponding data owner along with the file ID. The data owner fetches the user and file mapped information and corresponding access policy from the blockchain network. After checking the access details, access may be granted or denied to the user. After the access verification phase, the encrypted data fetch from the cloud storageand sent to the requested user.

### **5.2.2 Reading file**

This process is commonly referred to as file downloading. The file's specifications are recorded within a block of the blockchain. The blockchain is replicated across all nodes in the network. In order to access the file, you must first locate the node that has the block containing the file details. Once the block containing file details is obtained, it

will provide comprehensive information on the file, including its location, the number of parts it is separated into, the hash of each part, and the file's size, among other details.

The file's details are used to generate a hash for each component of the file in a block. Each section of the file is then compared to the corresponding part in the block. If all parts match, they are combined to form the whole file, making it easier to read.

### **5.2.3 Decrypt the file**

Upon obtaining the entire file, it was found to be unreadable. The file is encrypted, which is the cause behind that. The Ultra compression cloud storage (UCCS) employed the RSA asymmetric encryption technique for the purpose of encryption. RSA, also known as Rivest-Shamir-Adleman, is an asymmetric encryption method that employs distinct public and private keys for encryption and decryption operations. The RSA algorithm is a form of encryption that uses public key cryptography and is often regarded as the most secure method of encrypting data. RSA encryption involves the use of a public key to encrypt sensitive information, which can then only be decrypted using the corresponding private key.

UCCS assigns a public and master key pair to the data owner, while a public and secret key pair is assigned to the user. Now utilize the secret key to decipher the file.

### **5.2.4 Decompress the file**

The decrypted file remains unreadable due to its compressed format. To get the original file, using the specific decompression algorithm.

## **5.3 Static Analysis**

Firstly let's start with static analysis. In the training phase various algorithms will be tested and again the set of data splitter into 80% training and 20% testing. The graph below shows the result obtained from that. So It was found that Random Forest is the winner for the framework based on static analysis.

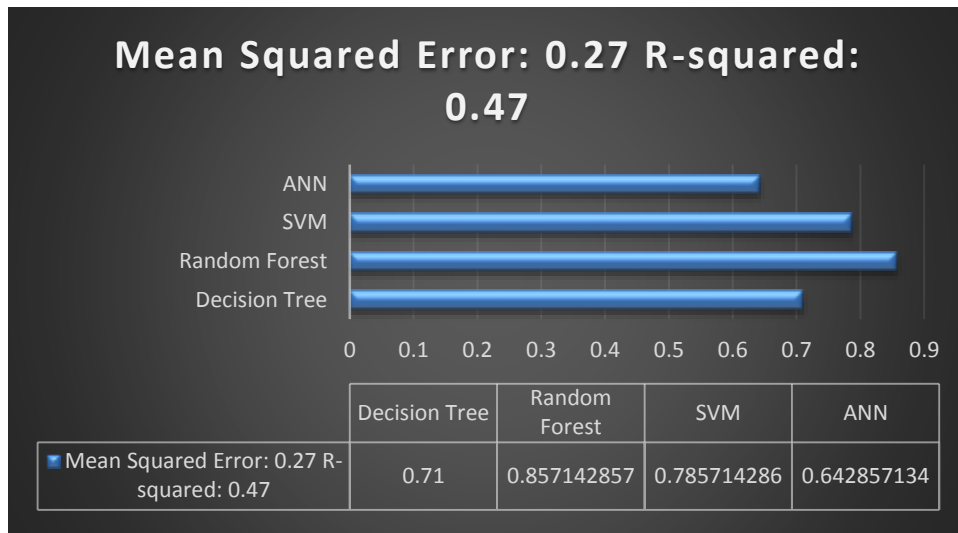


Figure 5.2 Comparison of various machine learning algorithms showing Random Forest is the best

## 5.4 Final Results

A total of 20 samples have been collected for the final testing. The framework offers several advantages over the typical frameworks currently available in the market:

- The model has achieved a 80% accuracy rate in securing the file.
- The framework is cost-free in comparison to traditional cloud storage software and secure authentication systems, which have a large cost.
- The framework is both non-proprietary and decentralized. Consequently, it is devoid of any bias. It is also devoid of unscrupulous companies seeking to make profit illegally.
- The framework guarantees the secrecy, integrity, and authenticity of the file.
- It is a highly efficient framework in comparison to other frameworks in the system. The complexity of the system is  $\log_n$  with respect to the number of nodes. The workload is evenly dispersed and allocated across all the nodes. Therefore, regardless of the complexity of the sample analysis, it is always rapid and does not require a significant amount of resources.
- The framework is advantageous for everyone. Both customers and companies.

#### **5.4.1 Results obtained from the Framework**

Upon completion, this method guarantees the protection of user data and files. By employing compression techniques, it is possible to minimize the amount of storage needed. It utilizes the asymmetric encryption algorithm for security purposes. The hashing technique, specifically the Markel root, is utilized to ensure integrity. In conclusion, it can be stated that the system is characterized by its affordability, scalability, flexibility, and, notably, it's enhanced security.

#### **5.4.2 Comparison with the existing Centralized systems**

Current frameworks frequently employ centralized technologies. These technologies are permissible but ineffective. The implementation of the framework involves the use of decentralised technologies such as Apache Spark on the Amazon AWS Cloud.

The graph demonstrates that the distribution is effectively achieved among all four nodes, and the application utilizes all of them for distributed execution. Four machines with IP addresses have been used here.

172.31.29.19

172.31.23.172

172.31.28.58

172.31.25.122

The four nodes were evaluated collectively under a load for a duration of one hour. The time is represented on the X-axis, while the distribution of load is represented on the Y-axis.

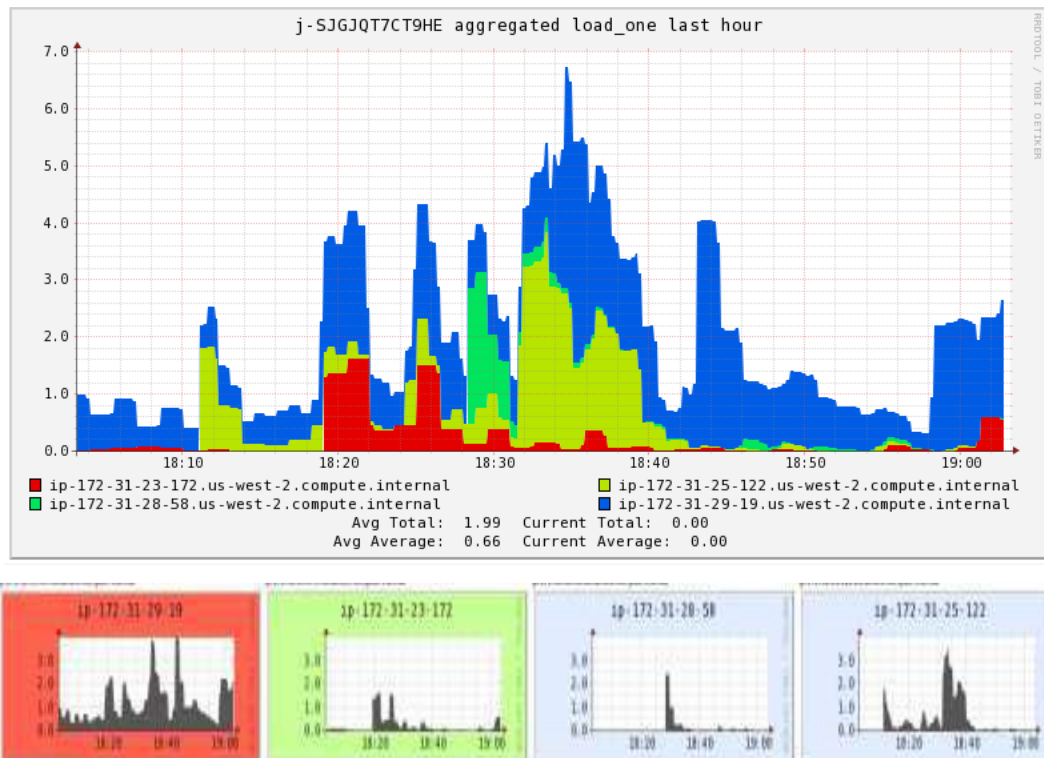


Figure 5.3: Distribution of load on 4 nodes aggregated for 1 hour.

When an instance of AWS is launched, it is assigned a hostname. The Amazon EC2 private DNS name is formatted as follows:

ip-173-31-23-172.us-west-2. compute.internal.

ip-173-31-23-172 is IPV4 address which is the internal domain.

The service provided is called "Compute" and it is available in the region known as "west". This name is intended for a Linux instance.

#### 5.4.2.1 Working of the Distributed Framework

- 1) Generate a prototype Amazon EMR cluster with the AWS Management Console.
- 2) Upon accessing the AWS Management Console, choose the Elastic MapReduce Service. Amazon Elastic Map Reduce (EMR) provides a managed Hadoop framework. Therefore, it is convenient, efficient, and economical to verify large amounts of data across flexible and expandable Amazon EC2 instances. It also allows for the execution of other distributed frameworks such as Apache Spark.
- 3) Next, choose the desired number of nodes to execute our model. It sequentially picks

nodes in increasing order, starting with one, then two, then three, and finally four, to assess the effectiveness of distribution.

4) It also chooses the necessary technologies for processing, such as 'Apache Spark' and 'Hadoop'. Spark is a framework used for processing data in a distributed manner, while Hadoop is used for processing data within a single node.

5) Subsequently, an Apache Zeppelin Notebook is made accessible. In this application program is written and executed in a distributed manner utilizing Hadoop and Apache Spark. Zeppelin is an online notebook designed for the purpose of interactive data exploration. This tool is utilized for the process of collecting and importing data, as well as for the tasks of identifying, analyzing, visualizing, and collaborating on the data.

#### **5.4.2.2 Blockchain for Records**

Blockchain is a decentralized data format that distributes trust among multiple nodes rather than relying on a single entity. It can be described as a distributed linked list that is stored on each computer (miner) in the network. It is an extensive network of transactions. It can be described as a distributed ledger. The network experiences constant growth as further transactions are incorporated. A blockchain is a permanent and unchangeable record of all the transactions that have taken place inside a network, which is stored by all the nodes in the network. Every block includes the address of the subsequent block.

Furthermore, it encompasses a detailed record of the transaction, encompassing the precise timestamp and all pertinent transactional details.

The blockchain is capable of being verified. The transaction is encoded as a 25-character hash. This hash is an alphanumeric sequence. In order for a transaction to be put to the blockchain, it must be validated by all the miners in the network. Every miner must vote on the legitimacy of every transaction. Prior to voting, individuals must furnish evidence of having successfully solved this arbitrary mathematical problem. This serves as evidence of their computational effort. The algorithm used for this purpose is known as the 'Proof of Work' algorithm.

Blockchain technology is founded on the amalgamation of two long-standing concepts: peer-to-peer technology and cryptography.

**P2P technology:** Torrents employ peer-to-peer (P2P) technology to disseminate information among nodes. Nodes inside the blockchain network engage in peer-to-peer (p2p) communication and locally store data pertaining to the blockchain. Both blockchain and torrents work without the need for a central server.

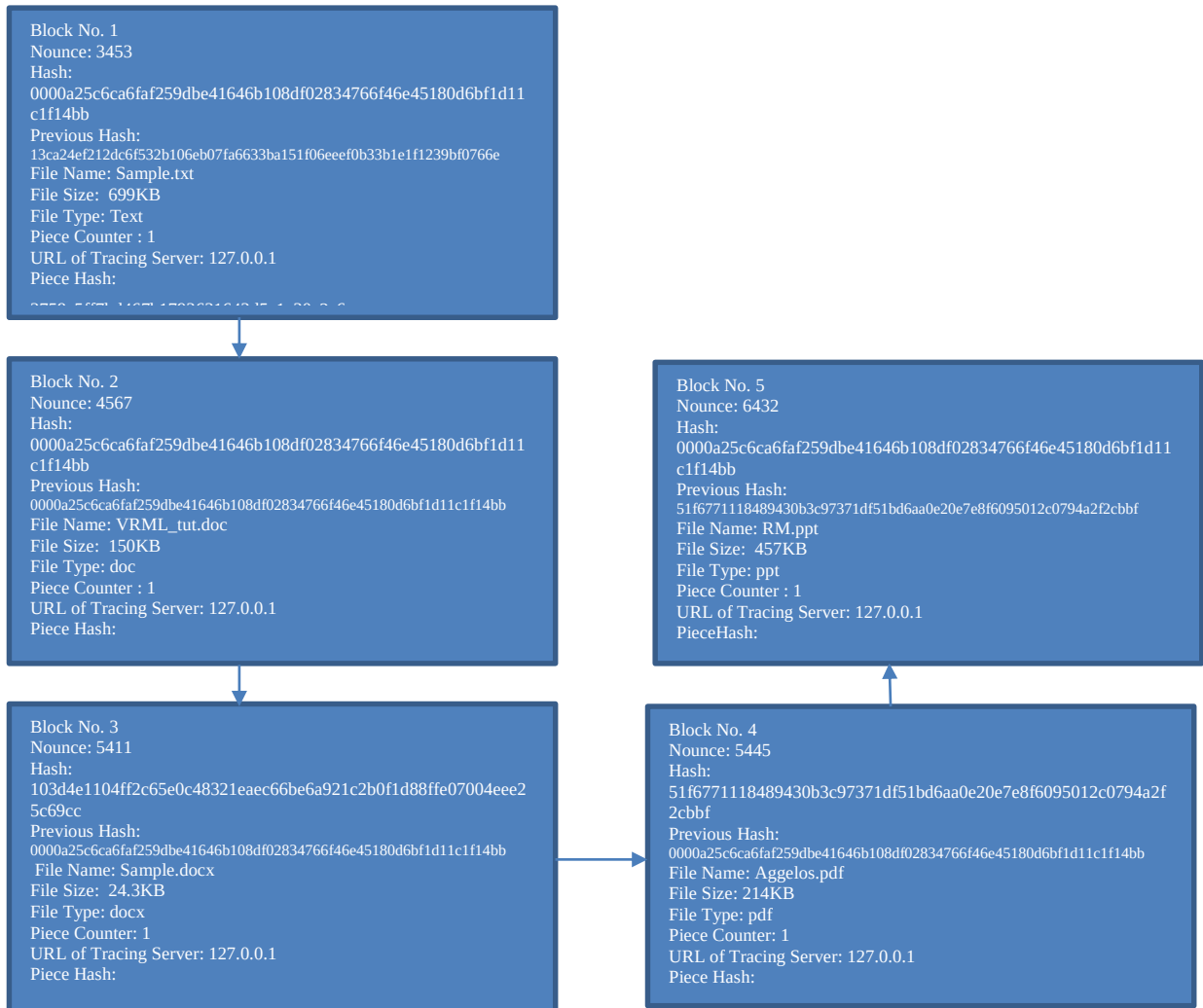


Figure 5.4: A part Blockchain of five blocks generated by Implemented model.

**Cryptography:** Ensures the attainment of both transparency and privacy. Cryptographic hash functions secure individual transactions. The immutability of the blockchain is ensured by the cryptographic linkage between blocks and transactions. Miners meticulously verify each transaction occurring on the network, rendering any form of cheating unfeasible.

1 **Nonce:** A nonce is a 32-bit (4-byte) field used in a blockchain network. It is utilized

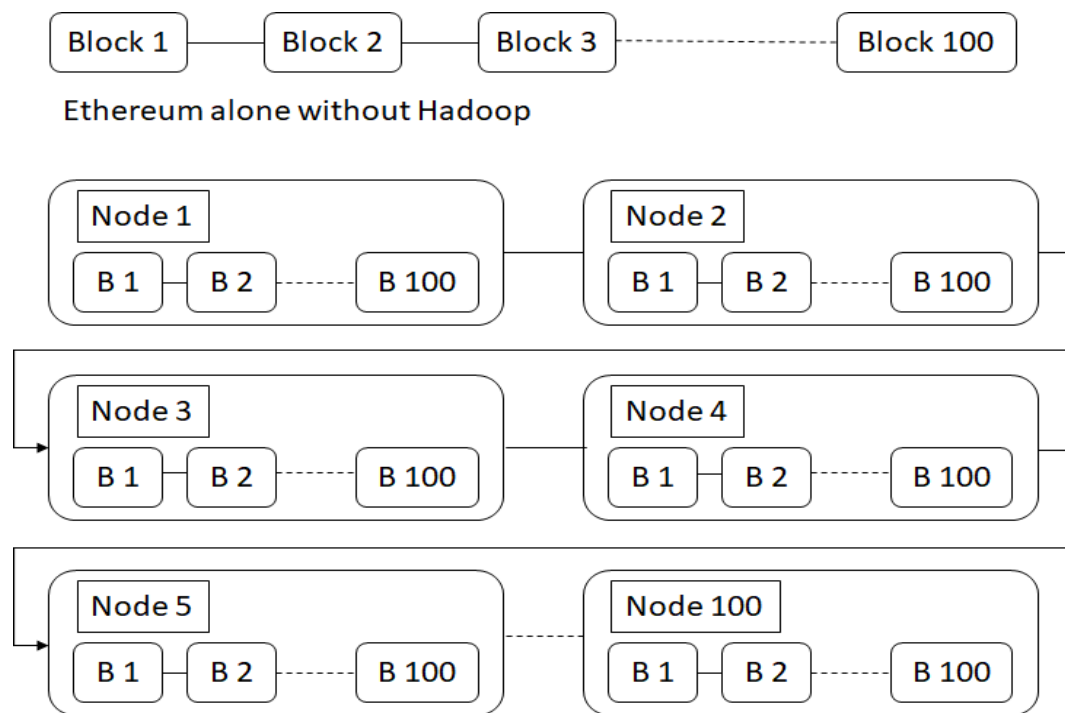
to determine the level of difficulty in mining a new block in order for the transaction to complete.

- 2 **Hash:** refers to an alphanumeric value that is computed for the data included within each block. This represents the unique identifier of this data in a digital format. The trustworthiness of the Blockchain is derived from the utilization of hashes. If any alterations are made to the contents, its hash value will be altered as well, resulting in rejection by other blocks in the chain.
- 3 **The previous hash:** is utilized to ensure the integrity of the chain. This includes the cryptographic hash of the preceding block.
- 4 **File Name:** This refers to the specific name assigned to a file that is contained in the transaction data.
- 5 **File Size:** This refers to the dimensions of a file.
- 6 **File Type:** refers to the specific format of a file, such as .txt, .doc, .xls, and others.
- 7 **Piece Counter:** This counter keeps track of the number of pieces into which the file is divided.
- 8 **The tracking server:** provides the URL of the node within the network where the file is located.
- 9 **Hash of pieces in File:** It is utilized to ensure the file's integrity. This text includes the cryptographic hash of a specific portion of a file.

#### 5.4.2.3 Using Ethereum for maintaining Blockchains

Originally, Bitcoins served as a decentralized digital payment system without any central regulatory authority. It functions as a decentralized digital money that is exchanged directly between users without the need for intermediaries. Ethereum serves as both an infrastructure and a payment system. It offers Smart Contracts and also

Crowdsourcing. Being designed in a Turing Complete language, it offers developers enhanced capability. A Turing complete language allows for the execution of any computable task given sufficient time and computational resources. The database in both Bitcoin and Ethereum is replicated across all nodes. Now, this augments the storage demand for each node involved in the detection technique. Consequently, it constructs the Ethereum blockchain on our Hadoop Cluster in a methodical and organized way.



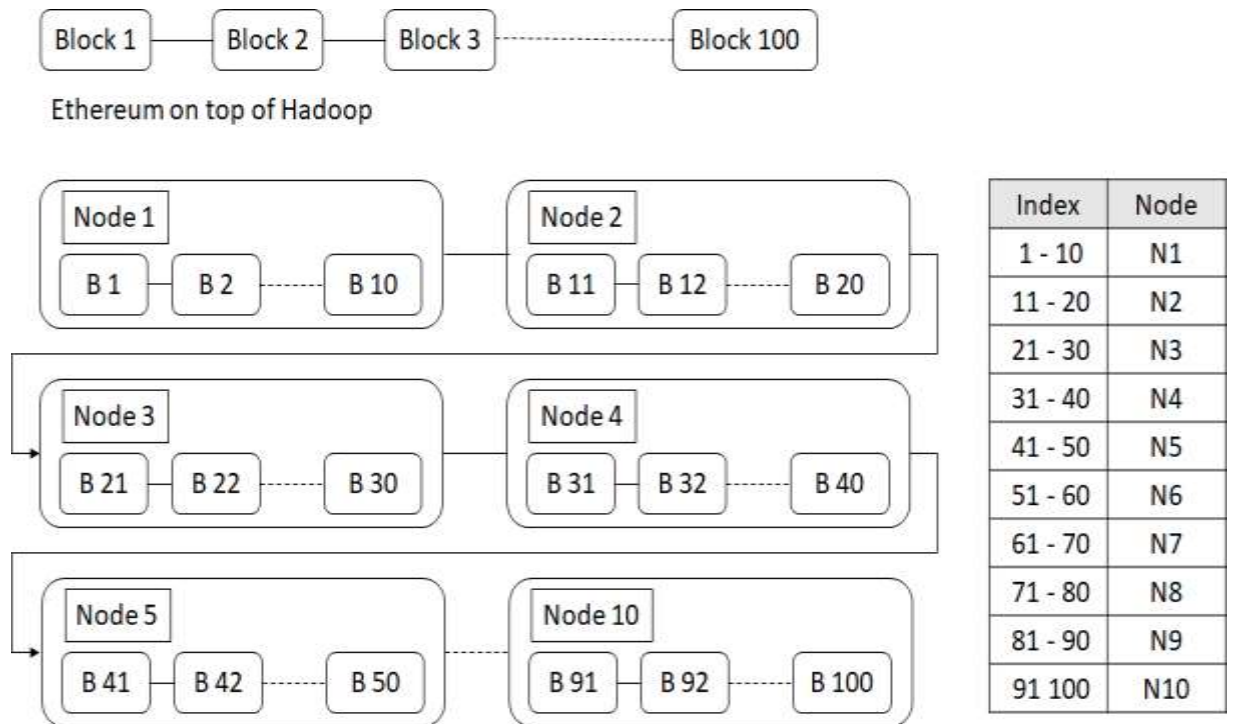


Figure 5.5: Comparison of Ethereum Blockchain database architecture without Indexing and with Indexing on top of Hadoop.

### 5.4.3 Summary of work done

This section provides a concise overview of the research conducted to address the difficulties and problems associated with the centralized cloud storage system by utilizing blockchain technology.

To tackle the privacy and security issues of the cloud storage system, this research proposes a new blockchain architecture that enhances the security aspects.

**Furthermore,** the suggested decentralized architecture, based on blockchain technology, offers a potential solution by removing the necessity for trusted intermediaries and addressing the problem of single points of failure. This is achieved through the implementation of a peer-to-peer network that grants equal validation privileges to all participants and guarantees transparency.

**Furthermore,** in order to enhance the security of the cloud storage system, the suggested design employs multiple cryptographic algorithms to establish a key generation mechanism, assure confidentiality, implement a data access control

mechanism, and guarantee the integrity of the cloud data. The blockchain network securely records and stores all transactions as hash values using cryptography. In addition, blockchain transactions are authenticated by the network participants to ensure that the user's connection with the cloud storage system stays private. The blockchain architecture handles the key-related and user access policy information in a decentralized manner, resulting in a more resilient environment. The blockchain network stores metadata of user data and records all access and validation activities.

**Finally**, in order to maximize resource efficiency and limit the time it takes to complete tasks, the suggested design implements a Honeybee optimization algorithm. This technique is used to optimize the usage of resources in the cloud storage system and reduce both execution time and response time.

#### **5.4.4 Contribution of the Research**

This research work outlines the methodology used to tackle the difficulties encountered in cloud storage systems. A suggested blockchain-based distributed architecture for cloud storage systems aims to offer the following five features to both organizations and individual end-users:

- Reduction in storage space using Huffman Coding algorithm.
- Distributed key generation process using CP-ABE algorithm
- Maintain the integrity of the stored data using the Markle root concept
- Optimization of the cloud storage system.
- Provide access control mechanism.

The study introduces a decentralized blockchain-based cloud architecture that utilizes the Huffman coding and CP-ABE algorithm. This is achieved by leveraging smart contract capability to enable a fully distributed system for managing keys, access policies, and the hash of the cloud data. Blockchain technology utilizes a decentralized approach to store information, ensuring that it remains unaltered once it is stored as a block in the blockchain structure.

**Furthermore**, this research endeavor guarantees the authenticity and reliability of cloud data storage by implementing the Markle root concept. The suggested design

partitions the file into numerous pieces. The metadata of divided file segments, represented as hashes, are arranged in a Merkle Tree structure to derive the Merkle root for the user's data. The Markle tree's root is used to verify the integrity of the cloud data.

**Furthermore**, the suggested study applies the optimization technique to the cloud storage system with the aim of reducing both the transaction response time and the execution time. Efficiently harnessing cloud resources minimizes the storage system by employing various procedures, such as data uploading, data validation, and result retrieval. These actions necessitate a certain amount of time for execution and the generation of a response, which is contingent upon the specific requirements of the application. Hence, minimizing the duration needed to execute these tasks enhances the efficiency of the cloud system.

**Lastly**, the proposed architecture was further improved by implementing the access policy in a hierarchical structure, resulting in a more refined access control mechanism at the system level.

## 5.5 Limitations of the Study

This study has several constraints.

- The framework must be able to function in a distributed context in order to optimize resource utilization. Within a centralized setting, the framework may experience reduced speed and inefficiency.
- The system relies on Smart contract blockchain and cloud technologies, creating a dependence on ancillary technologies. The smart contract blockchain is interchangeable with alternative blockchains, and the Amazon cloud can be substituted with alternative cloud platforms. However, the total effectiveness of the framework is partially reliant on these supplementary technologies as well. As they make progress, the framework also improves.

## 5.6 Conclusions

- The aforementioned architecture provides a secure means of storing files. This is a systematic approach that allows for accurate identification of most files and the mitigation of associated hazards. The framework effectively stored files of all types with precision. During the initial phase, the file will be compressed in order to decrease its size and conserve storage space. Subsequently, the compressed file can be encrypted in order to enhance the security of the file. The encrypted file is subsequently divided and subjected to a hashing process to verify the file's integrity. Finally, the file details will be placed in a block of the blockchain.
- We are utilizing RSA-based CP-ABE, which is considered the most secure method of encryption. Cracking RSA encryption is a formidable challenge. Brute force assault is ineffective due to the vast number of potential keys that need to be tested. Furthermore, this process requires a significant amount of time. A dictionary attack is not effective with the RSA technique because the keys used are numeric and do not contain any characters.
- Deciphering the frequency analysis of the characters is challenging due to the fact that a single encrypted block can represent many characters. There are no explicit mathematical techniques to exploit the RSA cryptography.
- It has been noted that by utilizing a dynamically produced blockchain, it is possible to demonstrate the ability to secure any file using this code. This file can be utilized to upload or download on a cloud platform and gain all the associated security rights. Subsequently, several files might be uploaded or downloaded in the cloud environment.

## 5.7 Further Research Potential

There is ample opportunity for additional investigation in this area. There could potentially exist more optimal methods for establishing safe cloud storage. These methods are likely to be more secure and have a significantly lower risk of being vulnerable. One can choose for the Neural Network technique due to the affordability of GPUs. This strategy is highly effective in a distributed context. Nevertheless, this

approach yielded a lower score compared to the other algorithms. Consequently, this method has remained unaffected. In the future, as this algorithm undergoes more development, it may get a higher score and be utilized more effectively. As all the systems included in the blockchain willingly contribute their CPU services, the efficiency of the blockchain and the overall system will rise as more blocks are added. Cloud technology facilitates the execution of dispersed services. As the efficiency and cost-effectiveness of Cloud services increase, the system continues to develop. The suggested architecture will be expanded to include a blockchain-based method for securely deleting cloud data while utilizing cloud services.

## Bibliography

- 1 Abimbola (January 2021), 'Cloud Computing Concept and Roots', <https://www.researchgate.net/publication/348958633>
- 2 Abdullah, & Kabir (2021), 'A Multilayer Security Framework for Cloud Computing in Internet of Things (IoT) Domain', *Journal of Computer and Communications*, pp. 31-42.
- 3 Adhanadi, Novamizanti, & Budiman (April 2020), 'DWT-SMM-based audio steganography with RSA encryption and compressive sampling', *TELKOMNIKA Telecommunication, Computing, Electronics and Control* Vol. 18, No. 2, pp. 1095~1104
- 4 Agrawal & Vyas (21 October 2020), 'Encoding Techniques for Image Compression: A Survey', <https://www.researchgate.net/publication/344786311>
- 5 Ageed, & Zeebaree (03/01/2024), 'Distributed Systems Meet Cloud Computing: A Review of Convergence and Integration', *International Journal of INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING*, pp. 469-490
- 6 Ahmed (2020), 'Virtualization Mechanism and Tools: A Comprehensive Survey', *International Journal of Computer Science Engineering*, Volume 9, No. 4 pp. 346-355.
- 7 Almutairi, Alghanmi, & Monowar (2021), 'Survey of Centralized and Decentralized Access Control Models in Cloud Computing', (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, Vol. 12, No. 2, pp. 339-346.
- 8 Alonso, Echevarria, Casola, Torre, Huarte, Osaba & Lobo (2023), 'Understanding the challenges and novel architectural models of multi-cloud native applications – a systematic literature review', *Journal of Cloud Computing: Advances, Systems and Applications*, <https://doi.org/10.1186/s13677-022-00367-6>
- 9 Anwar, Anaya, Butt, Butt & Saad. (2020), 'Generation Analysis of Blockchain Technology: Bitcoin and Ethereum', *Information Engineering and Electronic Business*, pp. 30-39.

- 10 Arman, Rehnuma & Rahman (2020), 'Design and Implementation of a Modified AES Cryptography with Fast Key Generation Technique', IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)
- 11 Arslan & Goker (12 March 2022), 'Compress-Store on Blockchain: A Decentralized Data Processing and Immutable Storage for Multimedia Streaming', arXiv:1905.10458v2
- 12 Atwal, & Kumar (26 April 2021), 'A Comparative Analysis of Different Encryption Algorithms:RSA, AES, DSS for Data Security', <https://doi.org/10.20944/preprints202104.0673.v1>
- 13 Austria, (8-1-2020), 'Analysis of Blockchain-Based Storage Systems', UNLV Theses, Dissertations, Professional Papers, and Capstones. 3984. <http://dx.doi.org/10.34917/22085740>
- 14 Awad (2020), 'A Framework for Improving Information Security Using Cloud Computing', International Journal of Advanced Research in Engineering and Technology, Volume 11, Issue 6 pp. 264-280.
- 15 Azeroual, & Fabre (9 March 2021), 'Processing Big Data with Apache Hadoop in the Current Challenging Era of COVID-19', Big Data Cogn. Comput., 5, 12. <https://doi.org/10.3390/bdcc5010012>
- 16 Azmi & Hussain (November 22, 2020), 'A Bibliometric Analysis of Dropbox on Scopus Publication', Webology, Volume 18, Special Issue on Information Retrieval and Web Search, pp.54-78.
- 17 Ba, Hu, Chen, Hao, Li, & Yan (8 November 2021), 'A Blockchain-Based CP-ABE Scheme with Partially Hidden Access Structures', Hindawi Security and Communication Networks Volume 2021, Article ID 4132597, 16 pages <https://doi.org/10.1155/2021/4132597>
- 18 Bajaj, Jatain, Chaudhary, & Nagpal (2021), 'Cloud Storage Architecture: Issues, Challenges and Opportunities', International Journal of Innovative Research in Computer Science & Technology, Volume-9, and Issue-3 pp.70-73.

- 19 Bangar, Pallavi & Shetty (2022), 'A Review Paper on Cloud Computing Architecture, Types, Advantages and Disadvantages', International Journal of Advanced Research in Science, Communication and Technology, Volume 2, Issue 2, pp. 14-22.
- 20 Basavarajegowda & Sundaram (2022), "Enhanced CP-ABE with RSA for Secure and Revocable Data Transmission of Big Data in Cloud", International Journal of Intelligent Engineering and Systems, Vol.15, No.2, 2022, DOI: 10.22266/ijies2022.0430.05, Pages 47-56
- 21 Begum, Huq & Kumar (2020), 'Sandbox security model for Hadoop file system', Journal of Big Data, <https://doi.org/10.1186/s40537-020-00356-z>
- 22 Berawi, Sari, Addiani, & Madyaningrum (November 2021), 'Developing a Blockchain-based Data Storage System Model to Improve Government Agencies' Organizational Performance', International Journal of Technology 12(5) pp.1038-1047.
- 23 Bhuyan, Sontakke, Bisht, Singh & Adtani (2021), 'Research Paper on Security Issues in Cloud Computing', International Journal of Creative Research Thoughts', Volume 9, Issue 4 pp. 1490-1496.
- 24 Biagetti, Crippa, Falaschetti, Mansour & Turchetti (2021), 'Energy and Performance Analysis of Lossless Compression Algorithms for Wireless EMG Sensors', Sensors 2021, 21, 5160. <https://doi.org/10.3390/s21155160>
- 25 Bindu , Ramani & Bindu (September 2020), 'Optimized Resource Scheduling using the Meta Heuristic Algorithm in Cloud Computing', IAENG International Journal of Computer Science, Volume 47, Issue 3,
- 26 Chang & Cui (2021), 'Distributed Storage Strategy and Visual Analysis for Economic Big Data', Journal of Mathematics, Volume 2021.
- 27 Cossio (n. d.), 'Ethereum, IPFS and neural compression to decentralize and protect patient data in computational pathology',
- 28 Dam & Neumaie (29 Jun 2022), 'Challenges of mapping Vulnerabilities and Exposures to Open-Source Packages', arXiv: 2206.14527v1 [cs.SE]

- 29 Dimov & Kirov, (May 27–28, 2022), ‘Data Performance Evaluation of Cloud Storage Providers’, Information Systems & Grid Technologies: Fifteenth International Conference ISGT’2022, pp. 63-73
- 30 Doan, Psaras, Ott & Bajpai (23 December 2022), ‘Toward Decentralized Cloud Storage With IPFS: Opportunities, Challenges, and Future Considerations’, IEEE Computer Society, Digital Object Identifier 10.1109/MIC.2022.3209804, pp. 7-15
- 31 Elsaid, Abbas & Meinel (01 July 2020), ‘Live Migration Timing Optimization for VMware Environments using Machine Learning Techniques’, <https://www.researchgate.net/publication/342462911>
- 32 Fatima, Rehman, Fatima, Khan & Ali (29 July 2022), ‘Comparative Analysis of Aes and Rsa Algorithms for Data Security in Cloud Computing’, Eng. Proc. 2022, 20, 14. <https://doi.org/10.3390/engproc2022020014>
- 33 Gad, Mosa, Abualigah & Abohany (5 March 2022), ‘Emerging Trends in Blockchain Technology and Applications: A Review and Outlook’, Journal of King Saud University – Computer and Information Sciences, pp. 6719-6742.
- 34 Ganesh J, Priyadharshini R, Swetha K & Vaishnavi G (2022), ‘A Secure Decentralized Data Storage Framework Using Blockchain Technology’, International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 2, Issue 1, June 2022, pp. 330-335.
- 35 Ghani, Badshah, Jan, Alshdadi, & Daud (14 Apr 2020), ‘Cloud Storage Architecture: Research Challenges and Opportunities’, Research Pedia Info Open Access Journal, arXiv:2004.06809v1 [cs.NI], VOLUME 1
- 36 Gokulraj S., P.Ananthi, R.Baby, & E.Janani (n.d.) ‘SECURE FILE STORAGE USING HYBRID CRYPTOGRAPHY’
- 37 Gupta, & N V S Reddy (2021), ‘Enhancement of Security of Diffie-Hellman Key Exchange Protocol using RSA Cryptography’, AICECS 2021, Journal of Physics: Conference Series 2161 (2022) 012014 IOP Publishing doi:10.1088/1742-6596/2161/1/012014.

- 38 Guru & Ambhaikar (June 2020), 'Development of "RSA" Encryption Algorithm for Secure Data Transmission', Research Journal of Computer and Information Technology Sciences ISSN 2320 – 6527 Vol. 8(1), pp.9-12,
- 39 Harjito, Tyas, Suryani, & Wardani (2022), 'Comparative Analysis of RSA and NTRU Algorithms and Implementation in the Cloud', (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 13, No. 3, pp. 157-164.
- 40 He, Huang, Shi, Hu, & Fan (2021), 'Enabling Decentralized and Dynamic Data Integrity Verification for Secure Cloud Storage via T-Merkle Hash Tree Based Blockchain' Mobile Information Systems, Volume 2021, Article ID 9977744, 17 pages, <https://doi.org/10.1155/2021/9977744>
- 41 Hu, Fang, Yang & Zang (2020), 'Research on Cloud Architecture of Enterprise Distributed Business Information System Based on SOA', Journal of Physics: Conference Series.
- 42 Huang, Wang & Wang (2020), 'Research and Application of Smart Contract Based on Ethereum Blockchain', Journal of Physics: Conference Series.
- 43 Jayasankar, Thirumal & Ponnurangam (2021), 'A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications' Journal of King Saud University - Computer and Information Sciences, Volume 33, Issue 2, February 2021, Pages 119-140
- 44 Jha, Dhingra, Mittal, & Vardan (2022), 'Secured Document Storing Using Blockchain', IJRTI Volume 7, Issue 5, ISSN: 2456-3315, pp. 403-410
- 45 Kapil, Mishra & Gupta (2020), 'A Performance Perspective of Live Migration of Virtual Machine in Cloud Data Center with Future Direction', International Journal of Wireless and Microwave Technology pp. 48-56.
- 46 Kaur & Bansal (JANUARY 2020), 'Cloud Data Security Using Group Multi-Keyword Top K Similarity Search Using Asymmetric Encryption', INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 9, ISSUE 01, pp. 1708-1716

- 47 Kaur, Saxena & Mahajan (2021), 'A Comprehensive Study of Decentralized Cloud Storage Platforms-A Review', International Journal of Creative Research Thoughts, Volume 9, Issue 3 pp. 401-406.
- 48 Khalid, Ehsan, Al-ani, Iqbal, Hussain, Ullah & Nayab (6 February 2023), 'A Comprehensive Survey on Blockchain-Based Decentralized Storage Networks', IEEE Access, VOLUME 11, 2023, pp. 10995-11015.
- 49 Khan, An & Imran (2020), 'A Blockchain Ethereum Technology-Enabled Digital Content: Development of Trading and Sharing Economy Data', IEEE Access, Volume 8 pp. 217045-217056.
- 50 Khan, Aljoaey, Tabassum, Farzamnia, Sharma & Tung (2022), 'Proposed Model for Secured Data Storage in Decentralized Cloud by Blockchain Ethereum', Electronics 2022, 11, 3686. <https://doi.org/10.3390/electronics11223686>  
<https://www.mdpi.com/journal/electronics>
- 51 Kondor, Bulatovic, Stéger, Csabai & Vattay (23 August 2021), 'The Rich Still Get Richer: Empirical Comparison of Preferential Attachment via Linking Statistics in Bitcoin and Ethereum', ORIGINAL RESEARCH article Volume 4 - 2021 | <https://doi.org/10.3389/fbloc.2021.668510>
- 52 Koteswari & Kopparthi (2022), 'Analysis of Data Security and Privacy Protection in Cloud Computing Services', International Journal for Research in Applied Science & Engineering Technology, Volume 10 Issue 3 pp. 1928-1932.
- 53 Krishna, & Manikandan (2020), 'A Study on Cryptographic Techniques', International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Volume 6, Issue 4, pp. 321-327.
- 54 Kritika, Himani, Singh, Upadhyay, Agrawal & Gupta (2022), 'Centralize storage system with encryption vs decentralize storage system using blockchain', International Conference on Computational and Intelligent Data Science (ICCIDS-2022),
- 55 Kumar & Prakash (2020), 'Secure File Storage on Cloud Using Hybrid Cryptography Algorithm', International Journal of Creative Research Thoughts, Volume 8, Issue 7, pp. 334-340

- 56 Kumar, Athavale & Kartikey (2021), 'Security Issues in Cloud Computing: A holistic view', International Journal of Internet of Things and Web Services, Volume 6, pp. 18-29.
- 57 Lal & Rasheed (2020), 'A Review on Data Compression Techniques', International Journal of Advance Research and Innovative Ideas in Education, Volume-6 Issue-1, pp. 590-597.
- 58 Lau, Li, Macneil, McEwen, Shukla, & Xu (n. d.), 'COMPRESSING THE TRANSACTION DATA OF BLOCKCHAIN', Maths Industry workshop.
- 59 Lee, Kim, Shah, Ahn & Yoon (18 June 2021), 'Performance Evaluations of Distributed File Systems for Scientific Big Data in FUSE Environment', Electronics 2021, 10, 1471. <https://doi.org/10.3390/electronics10121471>
- 60 Li, Xu, Zhang, Guo & Cheng (n. d.), 'SoK: Decentralized Storage Network', School of Computer Science and Technology, Shandong University, Qingdao 266237, China
- 61 Lai, Yang, Liu, Li & Li (11 December 2023), 'Web3: Exploring Decentralized Technologies and Applications for the Future of Empowerment and Ownership', Blockchains 2023, 1, pp.111–131. <https://doi.org/10.3390/blockchains1020008>  
<https://www.mdpi.com/journal/blockchains>
- 62 Lima, & Campos (2022), 'Information Storage and Retrieval System: An analysis of impact of variables and measures aimed at the organization and retrieval of information centered on user', RDBCI: Rev. Dig. Bibliotec e Ci. Info. / RDBCI: Dig. J. of Lib. and Info. Sci.| Campinas, SP | v.20| e022012.
- 63 Liu, Patil, Afanasyev, Feltus, Shannigrahi, Zhang & Yu (2021), 'Designing Hydra with Centralized versus Decentralized Control: A Comparative Study', IWCI '21, December 7, 2021, Virtual Event, Germany, pp.4-10.
- 64 Loay, Sousi, Yehya & Joudi (November 2020), 'AES Encryption: Study & Evaluation', <https://www.researchgate.net/publication/346446212>
- 65 Macko & Hennessey, (2022), 'Survey of Distributed File System Design Choices', ACM Transactions on Storage, Vol. 18, No. 1, Article 4.

- 66 Manikyam, & Dev (February, 2021), 'A Framework for Leveraging Image Security in Cloud with Simultaneous Compression and Encryption Using Compressive Sensing', *Revue d'Intelligence Artificielle* Vol. 35, No. 1, pp. 85-91
- 67 Mbongue, Kwadjo & Bobda (12 Mar 2021), 'Performance Exploration of Virtualization Systems', arXiv:2103.07092v1 [cs.DC]
- 68 Mensah, Tchao, Sikora, Agbemenu, Mensah, Ahmed, Welte & Keelson (11 August 2022), 'An Overview of Technologies for Improving Storage Efficiency in Blockchain-Based IIoT Applications', *Electronics* 2022, 11, 2513. <https://doi.org/10.3390/electronics11162513>
- 69 Meng & Sun (12 October 2022), 'Research on Decentralized Storage Based on a Blockchain', *Sustainability* 2022, 14, 13060. <https://doi.org/10.3390/su142013060>
- 70 Mihus, (2022), 'Evolution of Practical Use of Blockchain Technologies By Companies', *Economics, Finance and Management Review*, Issue 1 (9).
- 71 Mohamad, Din, & Ahmad (February 2021), 'Research trends review on RSA scheme of asymmetric Cryptography techniques', *Bulletin of Electrical Engineering and Informatics* Vol. 10, No. 1, pp. 487~492
- 72 Naureen (2021), 'Big Data Analytics with Hadoop', *International Journal of Engineering Research & Technology (IJERT)* ISSN: 2278-0181 Published by, [www.ijert.org](http://www.ijert.org) ICRADL - 2021 Conference Proceedings, Special Issue – 2021, Volume 9, Issue 5, pp. 33-37.
- 73 Nevpurkar, Bandgar, Deshmukh, Thombre, Sadafule, & Bhat, (May 2020), 'Decentralized File Storing and Sharing System using Blockchain and IPFS', *International Research Journal of Engineering and Technology (IRJET)* e-ISSN: 2395-0056 Volume: 07 Issue: 05 | May 2020 [www.irjet.net](http://www.irjet.net) p-ISSN: 2395-0072, pp. 560-563
- 74 Nygaard, Meling & Olsen (March 27-31, 2023), 'Cost-effective Data Upkeep in Decentralized Storage Systems', *ACM* ISBN 978-1-4503-9517-5/23/03. <https://doi.org/10.1145/3555776.3577728> pp. 165-173.
- 75 Nygaard & Meling (10 Apr 2023), 'SNIPS: Succinct Proof of Storage for Efficient Data Synchronization in Decentralized Storage Systems', arXiv:2304.04891v1 [cs.DC]

- 76 Palacios, Gordón & Aranda, (2020), 'A critical review on blockchain assessment initiatives: A technology evolution viewpoint', Journal of Software: Evolution and Process published by John Wiley & Sons Ltd.
- 77 Pandit (06 September 2021), 'Hadoop Ecosystem: Technology Study, Architecture and Analysis', : <https://www.researchgate.net/publication/354378675>
- 78 Qabasiyu, & Maiwada (JAN – 2022), 'Use of VMware virtualization technology to deploy private cloud computing infrastructure as a service on business organizations', INTERNATIONAL JOURNAL OF RESEARCH CULTURE SOCIETY ISSN: 2456-6683 Monthly Peer-Reviewed, Refereed, Indexed Journal [Impact Factor: 5.743] Volume - 06, Issue - 01, pp. 59-66.
- 79 Qureshi, (2021), 'A Review Paper on Cloud Computer Architecture', Samvakti Journal of Research in Information Technology, Volume 2 pp. 16-23.
- 80 Qureshi, Qureshi, Tahir, Anwar, Hussain, Uddin & Chen (28 March 2022), 'Encryption Techniques for Smart Systems Data Security Offloaded to the Cloud', Symmetry 2022, 14, 695. <https://doi.org/10.3390/sym14040695>
- 81 Rajeh (February 16, 2022), 'Hadoop Distributed File System Security Challenges and Examination of Unauthorized Access Issue', Journal of Information Security, 2022, 13, <https://www.scirp.org/journal/jis> ISSN Online: 2153-1242, ISSN Print: 2153-1234, pp. 23-42
- 82 Rashmi M R, Goenka, Tenginkai, & Singh (2023), 'CLOUD STORAGE AND RETRIEVAL USING BLOCKCHAIN', International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 10 Issue: 06 | Jun 2023 [www.irjet.net](http://www.irjet.net) p-ISSN: 2395-0072, pp. 317-321
- 83 Reddy, & Sein, (2021), 'A New Framework Approach Enhances Security to Efficient Remote Collaboration in TPA Scheme for Cloud Storage', International Journal for Research in Applied Science & Engineering, Volume 9 Issue 12 pp. 681-685.
- 84 Rossiman, Rashid, Ramzuna, & Almunawar (July-September 2021), 'The Perception and Usage of Google Drive among Higher Education Institution Students in Brunei

- Darussalam', International Journal of Asian Business and Information Management Volume 12, Issue 3, pp. 222-241
- 85 Sachi (May 2020), 'Maintenance of Transparency in Cloud Computing Through Different Approaches', International Journal of Advanced Research in Science, Communication and Technology (IJARSCT) Volume 5, Issue 5, pp. 121-123.
  - 86 Said, Janjua, Hassan, Muzammal, Saleem, Thaipsisutikul, Tuarob & Nawaz (10 December 2021), 'Detailed analysis of Ethereum network on transaction behavior, community structure and link prediction', PeerJ Comput. Sci. 7:e815 <http://doi.org/10.7717/peerj-cs.815>
  - 87 Saif, Gharib, & Mousa (2021), 'A MATHEMATICAL PROPOSED MODEL FOR PUBLIC KEY ENCRYPTION ALGORITHMS IN CYBERSECURITY', Advances in Mathematics: Scientific Journal 10 (2021), no.9, pp. 3063–3092 ISSN: 1857-8365 (printed); 1857-8438 (electronic) <https://doi.org/10.37418/amsj.10.9.1>
  - 88 Semunigus, & Pattanaik, (2020), 'Analysis for Lossless Data Compression Algorithms for Low Bandwidth Networks 'Journal of Physics: Conference Series, doi:10.1088/1742-6596/1964/4/042046
  - 89 Sharma & Batra, (2020), 'Performance analysis of compression algorithms for information security: A Review', EAI Endorsed Transactions on scalable information system, Review article: EAI.EU
  - 90 Sharma & Kaur (3, March 2023), 'A Survey of Distributed Data Storage in the Cloud for Multitenant Application', International Journal of Performability Engineering, vol. 19, pp. 184-192.
  - 91 Sharma, & Sinha, (2021), 'An Efficient Approach on Data Security with Cloud Computing Environment: A Comprehensive Research', Turkish Journal of Computer and Mathematics Education, Volume.12 No.14, pp. 1372-1382.
  - 92 Sheth, Bhosale & Kadam (2021), 'Research Paper on Cloud Computing', 'Emerging Advancement and Challenges in Science, Technology and Management' pp. 87-92.
  - 93 Shukla, Gupta, Rajput, Goswami, & Sharma (2022), 'A Comparative Analysis of Lossless Compression Algorithms on Uniformly Quantized Audio Signals', Image, Graphics and Signal Processing, 2022, 6, 59-69,

- 94 Singh, Bharti & Hooda, (2020), 'An Analysis Of Common Vulnerabilities And Exposures In View Of Mitre Att&Ck', Asian Journal of Convergence in Technology, Volume VIII and Issue II, pp. 15-17.
- 95 Singh, Monga, Tanwar, Hong, Sharma & He (13 February 2023), 'Adoption of Blockchain Technology in Healthcare: Challenges, Solutions, and Comparisons', Applied Science 2023, 13, 2380. <https://doi.org/10.3390/app13042380>
- 96 Song, Han, Eom, & Son (28 November 2022), 'IPFSz: An Efficient Data Compression Scheme in InterPlanetary File System', IEEE Access, Volume 10, pp. 122601- 122611
- 97 Srividhya & Deepthi (2022), 'Data Security and Privacy Protection for Cloud Storage', International Journal of Advanced Research in Science, Communication and Technology, Volume 2, Issue 7, pp. 875-881.
- 98 Sun, Li, Xu & Huang (26 Apr 2022). 'The SecurityWar in File Systems: An Empirical Study from A Vulnerability-Centric Perspective', arXiv:2204.12590v1 [cs.CR]
- 98 Thapliyal & Dimri (Mar 30, 2022), 'Load Balancing in Cloud Computing Based on Honey Bee Foraging Behavior and Load Balance Min-Min Scheduling Algorithm', International Journal of Electrical and Electronics Research (IJEER), Volume 10, Issue 1, pp.01-06, e-ISSN: 2347-470X
- 99 Tushar, Sharma & Mishra, (2021), 'Cryptographic Algorithm For Enhancing Data Security: A Theoretical Approach, International Journal of Engineering Research & Technology, Vol. 10 Issue 03, pp. 274-277.
- 100 Taherdoost, (2023), 'Smart Contracts in Blockchain Technology: A Critical Review', Information 2023,
- 101 Thajel & Hashim (2020), 'A new technique for data compression', IOP Conf. Series: Materials Science and Engineering.
- 102 Venkatakrishna & Reddy (2020), 'Time Efficient Distributed File Storage and Sharing Using P2p Networks', Journal of Interdisciplinary Cycle Research, Volume 12, Issue 4 pp. 1055-1061.
- 103 Wadi, Shidore, Surangalikar, Talewar, & Savalajkar (2022), 'P2P Ride-Sharing using Blockchain Technology', International Journal of Engineering Research & Technology, Vol. 11 Issue 09, pp. 78-82.

- 104 Wang, Gu, and Ma, (2020), 'A Multimode Network Steganography for Covert Wireless Communication Based on BitTorrent', Hindawi Security and Communication Networks Volume 2020.
- 105 Wang, (2022), 'Application Research of Data Encryption Technology in Computer Network Information Security', Hindawi Security and Communication Networks Volume 2022.
- 106 Wu, Yang & Lu (2020), 'DESIGN OF BLOCKCHAIN LEDGER COMPRESSION ALGORITHM', Computer Science & Information Technology (CS & IT), DOI: 10.5121/csit.2020.100107 pp. 71-80
- 107 Xing, Shu, Kang, & Zhao (2022), 'Peertrap: An Unstructured P2P Botnet Detection Framework Based on SAW Community Discovery', Hindawi Wireless Communications and Mobile Computing Volume 2022,
- 108 Xiao, (2022), 'Research and Application of Cloud Platform-Oriented Intelligent Information Management System', Hindawi Wireless Communications and Mobile Computing Volume 2022,
- 109 Yang, Hu & Qin (n.d.), 'Compression Performance Analysis of Different File Formats'
- 110 Ye & Park (11 May 2021), 'Reliable Vehicle Data Storage Using Blockchain and IPFS', Electronics 2021, 10, 1130. <https://doi.org/10.3390/electronics10101130>.

# Annexure

## Questionnaire

### ORGANIZATIONS AVAILING CLOUD SERVICES

This survey aims to gain insights about the utilization of cloud services in various enterprises, considering different aspects. Please respond to all the questions provided in the survey.

#### Section I. Demographic Question

1. Email ID

.....

2. Name

.....

3. Company Name

.....

4. Location (Mark Only One)

- ☐ New Delhi
- ☐ Bangalore
- ☐ Pune
- ☐ Navi Mumbai
- ☐ Mumbai
- ☐ Indore
- ☐ Kolkatta
- ☐ Channai
- ☐ Other

5. Type of Organization (Check all that apply)

- ☐ Telecommunication
- ☐ Technology Services
- ☐ Financial / Banking
- ☐ Education
- ☐ Healthcare
- ☐ Government
- ☐ Retail
- ☐ Transportation
- ☐ Other

6. Which level do you hold in your organization (Make only one oval?)
- ☐ Executive Level
  - ☐ Senior Level
  - ☐ Middle Level
  - ☐ Lower Level
7. What is your degree of responsibility for your organization's cloud services? (Mark only one oval)
- ☐ Primary Responsibility
  - ☐ Secondary Responsibility
  - ☐ Sometime Responsible, whenever needed
  - ☐ Not Responsible

## **Section II: Technical Questions**

8. What are the technical reasons for availing Cloud Services (Check all that apply)
- ☐ Data Backup
  - ☐ Data Compression
  - ☐ Load balancing
  - ☐ On Demand Service
  - ☐ Shared Infrastructure
  - ☐ Fault Tolerance
  - ☐ Clustering
9. What is the required bandwidth for data upload on cloud? (Mark only one oval)
- ☐ 25 Mbps
  - ☐ 50 Mbps
  - ☐ 100 Mbps
  - ☐ 200 Mbps
10. Which data transmission mode is used by your organization? (Mark only one oval)
- ☐ Simplex
  - ☐ Half Duplex
  - ☐ Full Duplex
11. Which authentication method is used while availing cloud service?
- ☐ Multifactor Authentication
  - ☐ Two Factor Authentication
  - ☐ Three Factor Authentication
  - ☐ OTP
  - ☐ Password

12. Which access control policy is used while availing cloud service?

- ☐ Mandatory Access Control (MAC)
- ☐ Discretionary Access Control (DAC)
- ☐ Role Base Access Control (RBAC)
- ☐ Rule Based Access Control (RAC)
- ☐ Attribute Based Access Control (ABAC)

### **Section III: Cloud Service-Related Questions**

13. Which type of cloud service were most common in your organization over past 3 years (Check all that apply)

- ☐ Storage as Service
- ☐ Platform as Service
- ☐ Security as Service
- ☐ Software as Service
- ☐ Infrastructure as Service
- ☐ Database as Service
- ☐ Testing as Service
- ☐ None

14. What percentage of IT budget is allocated to cloud service in your organization (Mark only one oval)

- ☐ >10%
- ☐ 10 – 20%
- ☐ 20 - 30%
- ☐ 30 – 40%
- ☐ 40 - 50%
- ☐ > 50%

15. What is the level of dependency of cloud services? (Mark only one oval)

- ☐ Fully?
- ☐ Partial
- ☐ Occasional
- ☐ No concern

16. What is the size of data uploaded on cloud? (Mark only one oval)

- ☐ 1 TB
- ☐ 2 TB
- ☐ 3 TB
- ☐ 4 TB
- ☐ 5 TB

17. What is the frequency of data upload on cloud? (Mark only one oval)

- ☐ Daily
- ☐ Alternate Day
- ☐ Twice a Week
- ☐ Weekly
- ☐ Fortnightly

18. How many users in your organization use cloud service. (Mark only one oval)

- ☐ 10
- ☐ 20
- ☐ 50
- ☐ 100
- ☐ More than 100

19. Which Cloud deployment model is used? (Mark only one oval)

- ☐ Public Cloud?
- ☐ Private Cloud
- ☐ Community Cloud
- ☐ Hybrid Cloud
- ☐ Multi Cloud

20. What is the average response by CSP after threat detection (Mark only one oval)

- ☐ 1 Hour
- ☐ 1 Day
- ☐ > 1 Day
- ☐ 1 Week
- ☐ > 1 Week

#### **Section IV: Security Related Questions**

21. What are the most common reason for availing cloud service (Check all that apply)

- ☐ Security Issue
- ☐ Privacy Issue
- ☐ Availability
- ☐ Accessibility
- ☐ Cost Saving
- ☐ Other

22. What level of security gain with the use of cloud services (Mark only one oval)

- ☐ High
- ☐ Critical
- ☐ Personal
- ☐ Low

23. Which backup drive is used? (Mark only one oval)

- ☐ Acronise
- ☐ BackBlaze
- ☐ CrashPlan
- ☐ DropBox
- ☐ IDrive
- ☐ OneDrive

24. Frequency of data backup on cloud

- ☐ Daily
- ☐ Alternate Day
- ☐ Twice a Week
- ☐ Weekly
- ☐ Fortnightly

25. Which data encryption standard is used by CSP for your organization (Mark only one oval)

- ☐ AES?
- ☐ DES
- ☐ RSA
- ☐ ECC
- ☐ TSL

26. Did the employee in your organization store sensitive data on cloud?

- ☐ YES
- ☐ No

27. What level of security gain by the use of cloud services (Mark only one oval)

- ☐ Confidentiality
- ☐ Integrity
- ☐ Non-Repudiation
- ☐ Availability
- ☐ Authenticity
- ☐ Reliability

28. What are the threats need to face during the use of cloud service (Check all that apply)

- ☐ Data Breach
- ☐ Data Leakage and Loss
- ☐ Insecure API's
- ☐ Account Hijacking
- ☐ Denial of Service
- ☐ Abuse of Cloud Planning
- ☐ Inadequate Cloud Planning
- ☐ Cloud Related Malware
- ☐ Closser of Cloud Service
- ☐ Natural Disaster
- ☐ Hardware Failure
- ☐ Shared Technology Vulnerabilities

29. What is the most important factor hindering use of cloud services? (Check all that apply)

- ☐ Resource Crunch
- ☐ Insufficient Training
- ☐ Lack of Information
- ☐ New Automated Services
- ☐ Other

## **List of Publications**

1. Prof. Swati D. Ghule & Dr. Anup Girdhar, March 2017, "ROLE OF DATA COMPRESSION IN CYBER SECURITY", Cyber Times International Journal of Technology & Management, Vol. 10, Issue 1, Impact Factor 7.075.
2. Prof. Swati D. Ghule & Dr. Anup Girdhar, February 2018, "Cloud Repository as Storage Service with Security of Data", Research Dimension(International), Page 1-5, Impact Factor 5.1899.
3. Prof. Swati D. Ghule & Dr. Anup Girdhar, March 2018, "Importance of Hash value in Security of cloud storage", Cyber Times International Journal of Technology & Management, Vol. 11 Issue 1, Impact Factor 7.233.
4. Prof. Swati D. Ghule & Dr. Anup Girdhar, March 2019, "Comparative Study of Compression Tools like TAR, RAR, ZIP", IOSR Journal of Engineering (IOSRJEN), Vol. 1 Pg. 29-33, Impact Factor 1.645.
5. Prof. Swati D. Ghule & Dr. Anup Girdhar, March 2020, "Enhancing Torrent by supporting Block Chain Management", An International Multidisciplinary Quarterly Bilingual Peer Reviewed Referred Research Journal, Vol. 7 Issue 25, Pg. 224-229.
6. Prof. Swati D. Ghule & Dr. Anup Girdhar, February 2020, "Performance Analysis of ZIP File", Journal CLIO an annual interdisciplinary journal of history 2020, Vol. 6 Issue 6, Impact Factor 5.6.
7. Prof. Swati D. Ghule & Dr. Anup Girdhar, March 2021, "Cloud as A Centralized Repository for Education Institutes", International Journal of Future Generation Communication and Networking, Vol. 14, Issue 1, Pg. 1684-1691.
8. Prof. Swati D. Ghule & Dr. Anup Girdhar, March 2021, "Evaluation of Huffman Algorithm for Compression Standards", Bioscience Biotechnology Research Communications An Open Access International Journal, Vol. 14, Issue 5, Pg. 44-46.
9. Prof. Swati D. Ghule & Dr. Anup Girdhar, July 2024, "Compression Algorithm used in creation of PDF file", Journal of the School of Language Literature and Culture Studies, Series 23, Book 2.
10. Prof. Swati D. Ghule & Dr. Anup Girdhar, November 2024, "Method proposed for Avoidance of Data Replication in Distributed Storage System", Education Administration:Theory and Practice, Volume 30, Issue 11.

---

# *A CRYPTOGRAPHIC FRAMEWORK FOR ULTRA COMPRESSION TECHNIQUES USED FOR SECURED CLOUD STORAGE*

A Thesis

SUBMITTED TO THE  
TILAK MAHARASHTRA  
VIDYAPEETH, PUNE

FOR THE DEGREE OF  
**DOCTOR OF PHILOSOPHY**

In Computer Science

Under the Board of Modern Sciences &  
Professional Skills



BY

Mrs. Swati D. Ghule  
(25315085004)

UNDER THE GUIDANCE OF

**Dr. ANUP GIRDHAR**

DEPARTMENT OF COMPUTER  
SCIENCE

2025

---

# 80\_Recommendations

## 6.1 Summary of work done

This section provides a concise overview of the research conducted to address the difficulties and problems associated with the centralized cloud storage system by utilizing blockchain technology.

To tackle the privacy and security issues of the cloud storage system, this research proposes a new blockchain architecture that enhances the security aspects.

**Furthermore**, the suggested decentralized architecture, based on blockchain technology, offers a potential solution by removing the necessity for trusted intermediaries and addressing the problem of single points of failure. This is achieved through the implementation of a peer-to-peer network that grants equal validation privileges to all participants and guarantees transparency.

**Furthermore**, in order to enhance the security of the cloud storage system, the suggested design employs multiple cryptographic algorithms to establish a key generation mechanism, assure confidentiality, implement a data access control mechanism, and guarantee the integrity of the cloud data. The blockchain network securely records and stores all transactions as hash values using cryptography. In addition, blockchain transactions are authenticated by the network participants to ensure that the user's connection with the cloud storage system stays private. The blockchain architecture handles the key-related and user access policy information in a decentralized manner, resulting in a more resilient environment. The blockchain network stores metadata of user data and records all access and validation activities.

**Finally**, in order to maximize resource efficiency and limit the time it takes to complete tasks, the suggested design implements a Honeybee optimization algorithm. This technique is used to optimize the usage of resources in the cloud storage system and reduce both execution time and response time.

## 6.2 Contribution of the Research

This research work outlines the methodology used to tackle the difficulties encountered in cloud storage systems. A suggested blockchain-based distributed architecture for cloud storage systems aims to offer the following five features to both organizations and individual end-users:

- Reduction in storage space using Huffman Coding algorithm.
- Distributed key generation process using CP-ABE algorithm
- Maintain the integrity of the stored data using the Markle root concept
- Optimization of the cloud storage system.
- Provide access control mechanism.

The study introduces a decentralized blockchain-based cloud architecture that utilizes the Huffman coding and CP-ABE algorithm. This is achieved by leveraging smart contract capability to enable a fully distributed system for managing keys, access policies, and the hash of the cloud data. Blockchain technology utilizes a decentralized approach to store information, ensuring that it remains unaltered once it is stored as a block in the blockchain structure.

**Furthermore**, this research endeavor guarantees the authenticity and reliability of cloud data storage by implementing the Markle root concept. The suggested design partitions the file into numerous pieces. The metadata of divided file segments, represented as hashes, are arranged in a Merkle Tree structure to derive the Merkle root for the user's data. The Markle tree's root is used to verify the integrity of the cloud data.

**Furthermore**, the suggested study applies the optimization technique to the cloud storage system with the aim of reducing both the transaction response time and the execution time. Efficiently harnessing cloud resources minimizes the storage system by employing various procedures, such as data uploading, data validation, and result retrieval. These actions necessitate a certain amount of time for execution and the generation of a response, which is contingent upon the specific requirements of the application. Hence, minimizing the duration needed to execute these tasks enhances the efficiency of the cloud system.

**Lastly**, the proposed architecture was further improved by implementing the access policy in a hierarchical structure, resulting in a more refined access control mechanism at the system level.

### 6.3 Limitations of the Study

- This study has several constraints.
- The framework must be able to function in a distributed context in order to optimize resource utilization. Within a centralized setting, the framework may experience reduced speed and inefficiency.
- The system relies on Smart contract blockchain and cloud technologies, creating a dependence on ancillary technologies. The smart contract blockchain is interchangeable with alternative blockchains, and the Amazon cloud can be substituted with alternative cloud platforms. However, the total effectiveness of the framework is partially reliant on these supplementary technologies as well. As they make progress, the framework also improves.

### 6.4 Conclusions

- The aforementioned architecture provides a secure means of storing files. This is a systematic approach that allows for accurate identification of most files and the mitigation of associated hazards. The framework effectively stored files of all types with precision. During the initial phase, the file will be compressed in order to decrease its size and conserve storage space. Subsequently, the compressed file can be encrypted in order to enhance the security of the file. The encrypted file is subsequently divided and subjected to a hashing process to verify the file's integrity. Finally, the file details will be placed in a block of the blockchain.
- We are utilizing RSA-based CP-ABE, which is considered the most secure method of encryption. Cracking RSA encryption is a formidable challenge. Brute force assault is ineffective due to the vast number of potential keys that need to be tested. Furthermore, this process requires a significant amount of time. A dictionary attack is not effective with the RSA technique because the keys used are numeric and do not contain any characters.
- Deciphering the frequency analysis of the characters is challenging due to the fact that a single encrypted block can represent many characters. There are no explicit mathematical techniques to exploit the RSA cryptography.
- It has been noted that by utilizing a dynamically produced blockchain, it is possible to demonstrate the ability to secure any file using this code. This file can be utilized to upload or download on a cloud platform and gain all the associated security rights. Subsequently, several files might be uploaded or downloaded in the cloud environment.

## 6.5 Further Research Potential

There is ample opportunity for additional investigation in this area. There could potentially exist more optimal methods for establishing safe cloud storage. These methods are likely to be more secure and have a significantly lower risk of being vulnerable. One can choose for the Neural Network technique due to the affordability of GPUs. This strategy is highly effective in a distributed context. Nevertheless, this approach yielded a lower score compared to the other algorithms. Consequently, this method has remained unaffected. In the future, as this algorithm undergoes more development, it may get a higher score and be utilized more effectively. As all the systems included in the blockchain willingly contribute their CPU services, the efficiency of the blockchain and the overall system will rise as more blocks are added. Cloud technology facilitates the execution of dispersed services. As the efficiency and cost-effectiveness of Cloud services increase, the system continues to develop. The suggested architecture will be expanded to include a blockchain-based method for securely deleting cloud data while utilizing cloud services.

## 6.6 Findings

- The model has achieved a 80% accuracy rate in securing the file.
- The framework is cost-free in comparison to traditional cloud storage software and secure authentication systems, which have a large cost.
- The framework is both non-proprietary and decentralized. Consequently, it is devoid of any bias. It is also devoid of unscrupulous companies seeking to make profit illegally.
- The framework guarantees the secrecy, integrity, and authenticity of the file.
- It is a highly efficient framework in comparison to other frameworks in the system. The complexity of the system is logn with respect to the number of nodes. The workload is evenly dispersed and allocated across all the nodes. Therefore, regardless of the complexity of the sample analysis, it is always rapid and does not require a significant amount of resources.
- The framework is advantageous for everyone. Both customers and companies.

## **6.7 Suggestions:**

1. How one can store, share, and retrieve data in a trustless, peer-to-peer systems can be compared with the proposed framework.
2. The performance of proposed framework can be compared with traditional centralized cloud storage providers (e.g., Amazon AWS, Google Cloud, etc.).
3. With the help of proposed framework one can build business models where users can lease out their extra storage and get paid in crypto currency.
4. Investigate how blockchain's distributed ledger can improve data availability by replicating data across multiple nodes (decentralized storage network).
5. Study latency issues, transaction speed, and overall performance of blockchain for storing large files.
6. Examine barriers to widespread adoption, such as technical complexity, user trust, and the availability of user-friendly interfaces.

**Plagiarism Report**

**Ph.D.**

**Date- 03/09/2024**

**Ghule Swati**

**Department – Computer Science**

**PRN.- 25315008504**

**Title – “ A Cryptographic Framework For Ultra Compression Techniques  
Used For Secured Cloud Storage ”**

|                                                          |            |
|----------------------------------------------------------|------------|
| <b>Chapter 1<br/>INTRODUCTION</b>                        | <b>14%</b> |
| <b>Chapter 2<br/>REVIEW OF LITERATURE</b>                | <b>12%</b> |
| <b>Chapter 3<br/>RESEARCH METHODOLOGY</b>                | <b>10%</b> |
| <b>Chapter 4<br/>ANALYSIS &amp; INTERPRETATION</b>       | <b>11%</b> |
| <b>Chapter 5<br/>CONCLUSION/FINDINGS/RECOMMENDATIONS</b> | <b>8%</b>  |



**Head of Ph.D. Department**