ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 12 N. 1 (2023), e31478
eISSN: 2255-2863
DOI: https://doi.org/10.14201/adcaij.31478

Blockchain Enabled Hadoop Distributed File System Framework for Secure and Reliable Traceability

Manish Kumar Gupta[a], Rajendra Kumar Dwivedi[b]

[a] Department of Information Technology & Computer Application, Madan Mohan Malaviya University of Technology, Gorakhpur, U.P., India, 273010
[b] Department of Information Technology & Computer Application, Madan Mohan Malaviya University of Technology, Gorakhpur, U.P., India, 273010
manish.testing09@gmail.com, rajendra.gkp@gmail.com

KEYWORDS

blockchain; commodity hardware; HDFS; security; vulnerabilities

ABSTRACT

Hadoop Distributed File System (HDFS) is a distributed file system that allows large amounts of data to be stored and processed across multiple servers in a Hadoop cluster. HDFS also provides high throughput for data access. HDFS enables the management of vast amounts of data using commodity hardware. However, security vulnerabilities in HDFS can be manipulated for malicious purposes. This emphasizes the significance of establishing strong security measures to facilitate file sharing within Hadoop and implementing a reliable mechanism for verifying the legitimacy of shared files. The objective of this paper is to enhance the security of HDFS by utilizing a blockchain-based technique. The proposed model uses the Hyperledger Fabric platform at the enterprise level to leverage metadata of files, thereby establishing dependable security and traceability of data within HDFS. The analysis of results indicates that the proposed model incurs a slightly higher overhead compared to HDFS and requires more storage space. However, this is considered an acceptable trade-off for the improved security.

1. Introduction

Nowadays, data is growing rapidly because of the increasing use of digital devices at a very high rate. These heterogeneous types of digital data are generated from various sources such as finance, medicine, agriculture, education, business, and many more fields. Big Data (BD) market will achieve 230 $ billion in 2025 and decrease the expenditures of retail, transportation, media, manufacturing,

and entertainment. BD can be defined as a leading-edge technology to analyze a big quantity of data and confine its key characteristics (J. Gantz et al., 2011). The era of BD opened the door to new opportunities (P. Cato et al. 2015). Nowadays, BD is used by the government to analyze and market for decision-making, and organizations use it to predict user interests and needs. McKinsey Global Institute report states that, with the help of BD, retailers have improved their margins by 60%(Manyika J et al., 2011). According to IBM, 2.5x1018 bytes of data are generated every day and 90% of the data was produced within the previous two years.

HDFS serves as a decentralized file system that facilitates the storage of enormous amounts of data on various computers within a Hadoop cluster. It was originally developed as part of the Apache Hadoop project, an open-source framework for distributed computing and BD processing. HDFS is designed to handle large datasets, typically ranging from terabytes to petabytes in size, and to provide high throughput data access (K. Shvachko et al., 2010). The data is split into blocks and replicated across multiple machines to ensure data availability and fault tolerance. By replicating data, HDFS can continue to function even if some of the nodes in the cluster fail. The two primary components of HDFS are the NameNode (NN) and the DataNode (DN) (Gupta M. K. et al., 2022). The NN stores the metadata of the file system, including the location of each file block and the permissions for each file. The DN component of HDFS is responsible for storing the physical data blocks on the local file system and communicates with the NN to ensure that the data is replicated and available. HDFS supports various operations, including read, write, and append, and provides a simple command-line interface for managing files and directories. It is typically used as a primary storage system for Hadoop, enabling large-scale data processing using tools such as MapReduce, Spark, and Hive (M. Zaharia et al. 2016). The International Organization for Standardization used the breakdown of BD security into four main themes to develop a security standard for BD security. An outline of the major BD security-related concerns is shown in Figure 1.

Organizations are showing keen interest and investing substantially in blockchain (BC) and BD, which were able to solve real-world challenges due to their huge potential. Today most customers use online transactions, which leads to a heavy amount of data. This exponential rise in the amount of data creates a new market for industries; helping them understand customer needs, their pattern of purchasing, and many other aspects.

Figure 1. Security challenges associated with BD

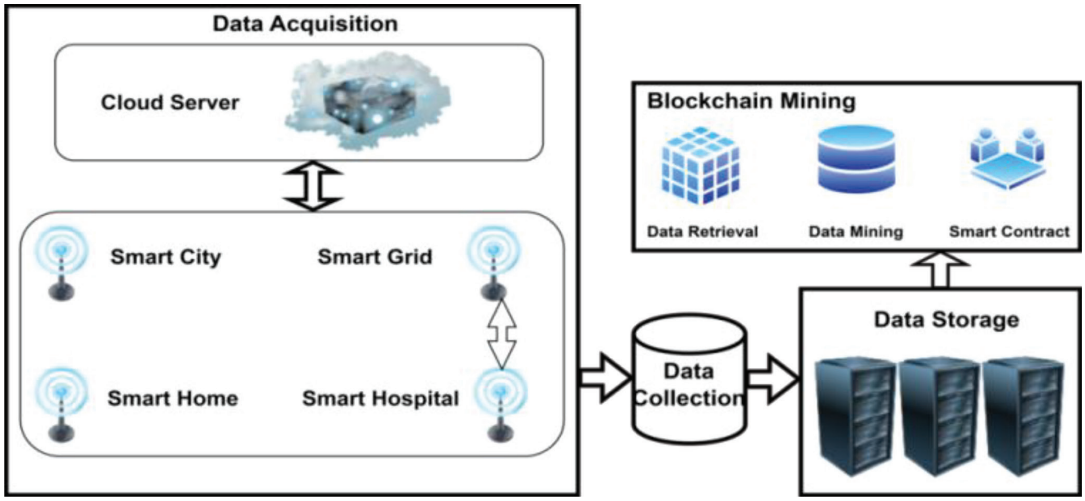

Figure 2. An overview of the integration of BC with BD

Figure 2 depicts the integration of Big Data (BD) with Blockchain (BC) and illustrates the life cycle of BD. The process involves data acquisition, data collection, data storage, and blockchain mining operations. Various sources, such as smart grids, smart cities, smart homes, and smart hospitals, contribute to data collection.

1. Data Acquisition: This phase involves gathering data from different sources, which can include the aforementioned smart systems. These sources generate large volumes of data, which are collected for further processing and analysis.
2. Data Collection: Once the data is acquired, it is collected and prepared for storage and analysis. This step involves organizing and structuring the data to make it suitable for storage and subsequent processing.
3. Data Storage: The collected data is stored in data storage devices, which can be traditional storage systems or more advanced solutions, such as distributed storage systems. This step ensures that the data is readily accessible and available for further analysis and processing.
4. Blockchain Mining Operations: In the context of integrating BD with BC, blockchain mining operations are performed on the collected data. Blockchain is a decentralized and distributed ledger technology that offers transparency, immutability, and security. Mining operations involve validating and adding data to the blockchain, ensuring its integrity, and creating an auditable record.

The integration of BD with BC provides several benefits, including enhanced data security, trust, transparency, and decentralized control. By leveraging the capabilities of blockchain technology, organizations can ensure the integrity and reliability of the collected data while enabling secure and transparent data sharing and analysis.

Motivated by matchless security, the decentralized, distributed, transparent, and immutable concept of BC technology, we propose a new technique that combines BC with HDFS. Fig 2 shows the integration of BD with BC.

BC-based security mechanism to secure BD in HDFS is proposed in this paper. The contributions of this paper are three-fold:

i. Clients upload their files on HDFS and after that ES256 (Encryption Standard 256) algorithm is applied to the rest data.
ii. Hyperledger Fabric (HF) retrieves data from HDFS and stores it within Hyperledger. Access to the data within Hyprledger is restricted to authenticated users only.
iii. HDFS uses CHV (Calculate Hash value) algorithm for hash value.
iv. NodeJS extracts hash values from HDFS using APIs and then stores them in the HF. NodeJS periodically maintains metadata and stores them in HF.

This paper contains six sections, Section1 introduced the topic and described the motivation behind the integration of BD and BC. An overview of Hadoop, BC, Smart Contract, and Hyperledgre Fabric is presented in Section 2. Section 3 is related to the literature survey of the proposed work. The main part of this research paper is presented in Section 4 which highlights the proposed work and the approach to data security. In Section 5, the experimental setup, performances, and limitations are discussed. And finally, the conclusion and future work are outlined in Section 6.

2. Background

This section presents an introduction to Hadoop and Blockchain and the motivation for their integration.

2.1. HADOOP

Hadoop is an open-source framework for distributed computing and BD processing that was originally developed by Apache. It provides a platform for storing, processing, and analyzing large and complex data sets using commodity hardware (M. K. Gupta et al., 2022). The core of the Hadoop ecosystem is HDFS, which is designed to store and manage large volumes of data across multiple machines in a cluster. Hadoop also includes a processing engine called MapReduce, which allows developers to write distributed computing jobs that can be run on the cluster (V. Mothukuri et al., 2015). Hadoop is designed to be highly scalable and faulttolerant. By distributing the data and processing it across many machines, Hadoop can continue to function even if some of the nodes in the cluster fail. Additionally, Hadoop provides features such as replication and data locality, which help ensure that data is available and accessible when needed. In addition to HDFS and MapReduce, the Hadoop ecosystem includes a wide variety of other tools and technologies for managing and processing BD. As provided by Lai et al. (2014), other tools in the Hadoop ecosystem include:

1. Hadoop YARN: a resource manager that allows for multiple processing engines, such as Spark or Hive, to share resources on a single cluster.
2. Apache Spark: a fast and flexible processing engine that can be used with Hadoop or standalone.
3. Apache Hive: a data warehousing tool that provides an SQL-like interface to query and analyze large datasets stored in distributed storage systems such as Hadoop.
4. Apache Pig: a high-level data processing tool that simplifies the creation of MapReduce jobs on Hadoop clusters.
5. HBase: a distributed, column-oriented database designed to store and manage large amounts of semi-structured or structured data on top of HDFS.

2.2. Blockchain

In the current era, blockchain (BC) is one of the most outstanding and secure technologies. According to IBM, BC is a decentralized, shared, and distributed ledger for recording transactions and tracking assets (tangible, such as property, house, or vehicle, or intangible, such as digital currency, or intellectual property) in the network. All transactions can be monitored by all participating members in a P2P manner (W. Viriyasitavat et al. 2019;L. Da Xu et al., 2019). Data in a BC is stored as an individual block, just like in a linked list. Each block in the BC contains three fields: data, previous hash, and hash. Data is stored in a P2P model so that all the participating nodes will know the full history of the entire BC Figure 3 depicts the working process of BC. Someone makes a transaction request. The desired transaction will cost the P2P network of computers a lot of money (Nodes). Using well-known techniques, the node of the network verifies the transaction and the user's state. A new block for the ledger is created by combining verified transactions with other transactions. The current BC is then updated with the new block. Table 1 provides a comparative analysis of different BC platforms.

Table 1. A comparative analysis of different BC platforms

	Quorum	**Hyperledger**	**Corda**	**Open chain**	**Multi Chain**	**EoS**
Open Source	✓	✓	✓	✓	✓	✓
Financial Services	✓	X	✓	X	✓	X
Cross Industry	X	✓	X	X	X	✓
Digital Asset Management	X	X	X	✓	X	X
Consensus Algorithms	PoA, BFT, Raft	Kafka, RBFT, Sumeragi, PoET	RBFT, Raft	Pluggable	PBFT	Delegate PoS
Language	Solidity	Java, Golang, NodeJS	Kotlin, Java	JavaScript	C++	C++

2.3. Hyperledger Fabric

HF is an open-source enterprise-grade BC platform developed by the Linux Foundation's Hyperledger project. It is designed to provide a modular and scalable framework for building distributed ledger applications for business use cases. HF uses a permissioned network model, which means that only authorized parties can access the network and its data. This makes it well-suited for enterprise applications that require a higher level of privacy, security, and control than public BC (Elli Androulaki et al. 2018). HF allows for the development of smart contracts, which are self-executing pieces of code that can automate business processes and enforce rules and policies. Smart contracts are written in general-purpose programming languages such as Go, Java, and JavaScript, which makes it easier for developers to create complex applications. HF also supports a pluggable consensus mechanism, which allows users to choose the consensus algorithm that best suits their needs. This flexibility enables HF to support a wide range of use cases, from supply chain management to digital identity to finance and more.

2.4. Comparative Analysis of Different Consensus Algorithms Used in the Hyperledger Framework

The Hyperledger community is developing several consensus methods and implementation strategies to guarantee modularity. Table 2 compares the consensus methods employed by the Hyperledger frameworks. Kafka, Redundant Byzantine Fault Tolerance (RBFT), and Sumeragi (BFT- Byzantine Fault Tolerance) used a voting-based consensus approach, and PoET used both a voting and lottery-based strategy.

Table 2. A comparative analysis of different consensus methods employed by the Hyperledger frameworks

	Kafka in HF Ordering Service	**RBFT in Hyperledger Indy**	**Sumeragi in Hyperledger Iroha (BFT)**	**PoET in Hyperledger Sawtooth**
Voting Based Approach	✓	✓	✓	✓
Lottery Based Approach	X	X	X	✓
Crash Fault Tolerance	✓	X	X	X
Byzantine Fault Tolerance	X	✓	✓	✓
Scalable	✓	✓	✓	✓
Real Time	✓	✓	✓	✓
Distributed	✓	✓	✓	✓
Needs of Node to handle Fault	--	3f +1	2f +1	--

3. Related Works

Xinhua Dong et al. (2015) focused on integrating blockchain technology into Hadoop Distributed File System (HDFS) to enhance the security and traceability of data provenance. Su et al. (2021) discussed a resource allocation mechanism for mobile social big data (BD) that considers security concerns. The authors proposed a matching-coalitional game solution for efficient resource allocation. J. Wu et al. (2018) presented a security situational awareness framework for smart grids using big data analysis. The paper discussed the use of big data analytics to enhance the security of smart grid systems. C. H. Liu et al. (2019) explored the application of blockchain technology and deep reinforcement learning in enabling secure data collection and sharing in the context of the Industrial Internet of Things (IIoT). G. Liu et al. (2022) introduced B4SDC, a blockchain system designed for secure data collection in Mobile Ad Hoc Networks (MANETs). The paper highlighted the importance of data security in MANETs and presented a blockchain-based solution. X. Xu et al. (2020) focused on blockchain-enabled computation offloading for the Internet of Things (IoT) in the mobile edge computing (MEC) environment. The paper proposed a solution called BeCome to enhance the efficiency and security of computation offloading. U. U. Uchibeke et al. (2018) discussed a blockchain-based access control ecosystem for enhancing the security of big data. The paper presented a framework to control and secure access to big data using blockchain technology. G. S. Aujla et al. (2018) focused on securing and auditing big data in the cloud environment. The authors proposed a solution called SecSVA for secure storage, verification, and auditing of big data. Z. Zhou et al. (2022) discussed the potential benefits of using blockchain for securing sensitive data in transportation systems and provided insights into the

application of blockchain for enhanced security and privacy in the era of 6G networks. P. Sharma et al. (2021) focused on leveraging blockchain technology to enhance the security of medical big data. It explored the potential of blockchain in ensuring data integrity, privacy, and confidentiality in healthcare settings. P. Sarosh et al. (2021) addressed the challenges related to data security, privacy, and confidentiality in the context of healthcare.A. Azzaoui et al. (2022) presented a blockchain-based architecture for secure storage and management of medical big data. The paper explored the integration of quantum computing and blockchain technology in the context of medical data security. T. Mohanraj et al. (2022) proposed a hybrid encryption algorithm for enhancing the security of big data stored in the Hadoop Distributed File System (HDFS). M. K. Yousif et al. (2023) discussed the use of the NTRUEncrypt method for information security in big data. The paper explored the application of this encryption method to protect the confidentiality and integrity of big data. S. Guan et al. (2023) presented a secure storage solution for big data in a cloud computing environment using Hadoop. The paper focused on ensuring the security and privacy of big data in cloud-based storage systems.

Table 3. Features and challenges of existing work

Author [citation]	**Methodology**	**Features**	**Challenges**
V. Mothukuri et al.	Integration of blockchain in HDFS.	• Blockchain integration for secure provenance traceability. • Enhancing data security in HDFS.	• Scalability of the blockchain system. • Overhead of blockchain operations.
Z. Su et al.	Matching-coalitional game solution.	• Security-aware resource allocation for mobile social BD. • Efficient allocation of resources.	• Complex resource allocation scenarios.
J. Wu et al.	BD analysis for smart grid security.	• BD analysis-based security situational awareness for smart grids.	• Privacy concerns with big data analysis. • Integration with existing smart grid systems.
C. H. Liu et al.	Blockchain and deep reinforcement learning for IoT.	• Blockchain-enabled data collection and sharing for industrial IoT. • Integration of deep reinforcement learning.	• Scalability of the blockchain system. • Efficient training of deep reinforcement learning models.
G. Liu et al.	Blockchain system for security data collection in MANETs.	• Secure data collection in Mobile Ad Hoc Networks (MANETs) using blockchain.	• Resource constraints in MANETs.
X. Xu et al.	Blockchain-enabled computation offloading forIoT in MEC.	• Blockchain-enabled computation offloading for IoT in the mobile edge computing (MEC) environment.	• Overhead of blockchain operations in MEC.

(continued)

Table 3. Features and challenges of existing work (continued)

Author [citation]	Methodology	Features	Challenges
U. U. Uchibeke et al.	Blockchain-based access control ecosystem for BD security.	• Blockchain-based access control for securing and controlling access to big data.	• Integration with existing access control systems.
G. S. Aujla et al.	Secure storage, verification, and auditing of BD in the cloud environment.	• Secure storage, verification, and auditing of big data in the cloud.	• Ensuring integrity and authenticity of stored data.
A. Azzaoui et al.	Blockchain-based Architecture for medical big data security.	• Integration of blockchain and quantum computing for secure storage and management of medical big data.	• Implementation challenges of quantum computing in a practical system.
T. Mohanraj et al.	Hybrid encryption algorithm for securing big data in HDFS.	• Hybrid encryption algorithm for enhancing the security of big data stored in HDFS.	• Key management and distribution for hybrid encryption.
M. K. Yousif et al .	NTRUEncrypt method for information security in big data.	• Information security for big data using the NTRUEncrypt method.	• Performance trade-offs between security and computational efficiency.
S. Guan et al.	Hadoop-based secure storage solution for big data in the cloud computing environment.	• Secure storage solution for big data in cloud computing using Hadoop.	• Scalability and performance considerations in a cloud environment.
P. Sharma et al.	Blockchain-enabled Hadoop.	• Ensures secure and reliable traceability.	• Interoperability between technologies. • Data fragmentation and synchronization.
Z. Zhou et al.	BC-based DFS.	• Provides secure and reliable traceability.	• Integration complexity. • Scalability and performance.
P. Sarosh et al.	Integrates blockchain with big data.	• Enhances data security and integrity. • Provides auditable and transparent data.	• Complex implementation and management. • Scalability and performance.

4. Proposed Model

The proposed approach for improving the security of data stored in HDFS by storing hash values in the BC is discussed in this section.

4.1. Blockchain Platform

A BC platform may be categorized into three groups: public, private, and permissioned. Public BC platforms allow any user to join the network, whereas only permitted users are allowed on private BC platforms. The Fabric and Quorum platforms are examples of private BC platforms. Private BC platforms are more secure than public ones, as only trusted users are allowed.

4.2. Architecture

The architecture model of the proposed system is shown in Figure 3. There are three components in our research work: HDFS, the BC network, and the scripting language for communication (NodeJS). HDFS cluster containsa single Name Node and multiple Data Nodes. Hyperledger Fabric is used to store hash value and NodeJS acts as a communication channel between the HDFS cluster and the BC network.When an HDFS client uploads the files on the Data Node (available in the HDFS cluster) the file transfer is secured by ES256 encryption. After that, the data node replicates the data in another data node based on the replication factor. Hyperledger Fabric is used to store metadata and edit log files, which include hash values and access times, but are not limited. CHV Algorithms have been used to find hash values. NodeJS extracts metadata files from HDFS using APIs and then stores them in the HF. The user can query HF to gain insight into the history of a particular file. The proposed model utilizes the BC to store the metadata.

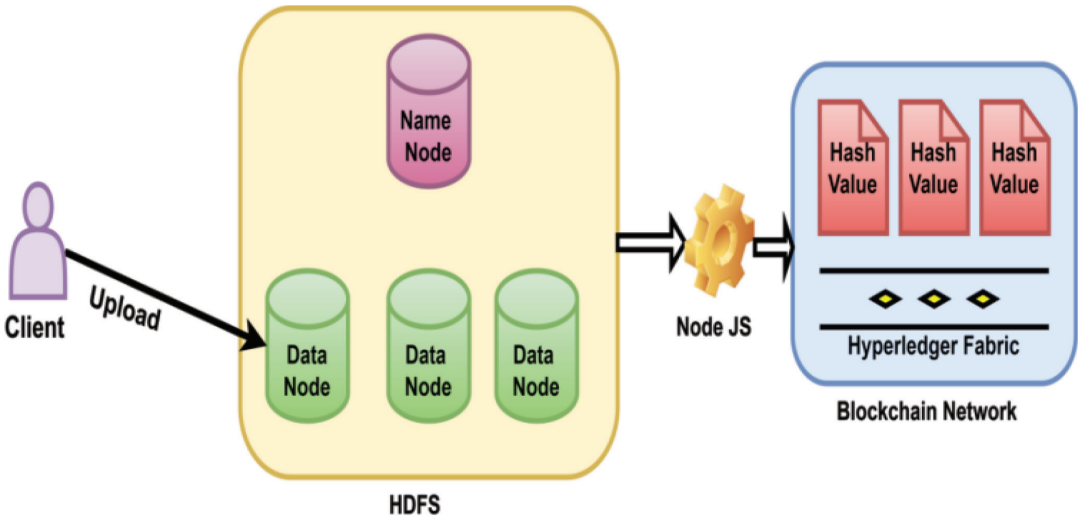

Figure 3. Architecture model

4.2.1 User

A user is classified as an administrator or an employee who has been granted access to HDFS. When adding, modifying, or deleting a file, the user is not required to know of the HF existence. All file operations are carried out solely using Hadoop's API.

4.2.2. API

The API is utilized to execute operations and retrieve information. However, if security measures are disabled on a local server, such as deactivating Kerberos, API actions are executable by any individual on the network without necessitating authentication.

4.2.3. Authenticity

When uploading a file to HDFS, the user must verify the file's authenticity by validating whether the checksum in HDFS corresponds to the checksum on their local machine.

4.3. Flowchart

Hyperledger Fabric and Hadoop frameworks have been used to implement this model.

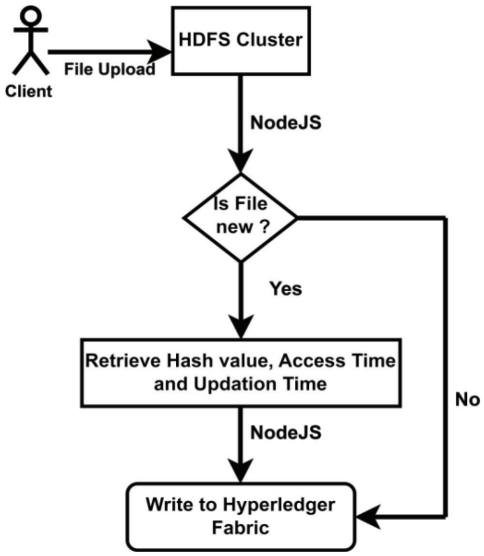

Figure 4. Flow chart

4.4. Algorithms

Algorithm 1: Data Transmission and Security

Initialize: HDFS Cluster, HF, NodeJS
Input: Users data
Output: Hash value
Begin
Step 1: The user uploadsa file on the data node (HDFS cluster).
Step 2: Encrypt data using **Algorithm 2: ES256**
Step 3: Encrypted data is now stored in Data Node, andreplicated according to the replication factor.

Step 4: Computation of hash value of encrypted data using **Algorithm 3: CHV**
Step 4: The hashvalue stored inHF.
Step 5: NodeJS periodically maintain the hash value.
Step 6: Users can generate a query to HF to read the changes.
End

Algorithm 2: Encryption Standard (ES256)

Input: Plain data, Keys (K_0, K_1……K_n)
Output: Encrypted data
Begin
Step 1: Create round keys by using KSA (Key Schedule Algorithm)
Step 2: Start the Encryption process
i. Computation of substitution byte using S-Box
ii. Shift row by using **Algorithm 4: ShR**
iii. Mix columns by matrix multiplication
iv. Perform XOR operation with round key and output of iii.
End

Algorithm 3: Calculate Hash Value (CHV)

Input: Encrypted data
Output: Hash Value
Begin
Step 1: Append 1 at the last position and pad 0 until the multiple of 512.
Step 2: Initialize the hash value with the fractional part of the square root of the first 8 prime numbers.
Step 3: Initialized round constant.
Step 4: Create message schedule(w) by using **Algorithm 5: MeSch**
Step 5: Perform compression operation by using **Algorithm 6.**
Step 6: Modify the final value by adding the current hash value to the compressed block.
Step 7: Concatenate the final hash.
End

Algorithm 4: Shift Rows (ShR)

Input: Rows of S-Box
Output: Shift rows
Begin
Step 1: No change in the first row of S-Box**.**
Step 2: Shift the second row by one bit to the left.
Step 3: Shift the third row by two bits to the left.
Step 4: Shift the fourth row by three bits to the left.
End

Algorithm 5: Message Schedule (MeSch)

Initialize:⊕ ←xor
Input: Data in 32 bit format
Output: Message Schedule
Begin
Step 1: Array[w]←Plain text (entry will be 32-bit format).
Step 2: w[0….63]
Step 3: perform i to iii for each integer ranging from 16 to 63
i. X0=(w[i-15] rr 7) ⊕ (w[i-15] rr18) ⊕ (w[i-15] >> 3)
ii. X1=(w[i-2]rr17) ⊕ (w[i-2] rr19) ⊕ (w[i-2] >> 10)
iii. w[i]=w[i-16] + X0 + w[i-7] +X1
End

Algorithm 6: Compression

Initialize:⊕ ←xor
rr← Right Rotate
&&← and
! ← Not
Input: Hash value
Output: Compressed hash value
Begin
Step 1: Initialize variables and set them equal to the current hash value.
α=hsv0, β=hsv1, μ=hsv2, £=hsv3, €=hsv4, ¥=hsv5, $=hsv6, ¢=hsv7
Step 2: for each integer ranging from 0 to 63
i. X1= (e rr 6) ⊕ (e rr 11) ⊕ (e rr 25)
ii. ch= (e && f) ⊕ ((!e) && g)
iii. tmp1= h+ X1 +ch+ k[i] +w[i]
iv. X0 = (a rr 2) ⊕ (a rr13) ⊕ (a rr 22)
v. Mj= (a && b) ⊕ (a && c) ⊕ (b && c)
vi. tmp2 = X0 + mj
vii. ¢=$, $=¥, ¥=€, €=£ + tmp1, £= μ, μ = β, β = α, α = tmp1+ tmp2
End

Algorithm 7: HDFS implementation

Input: Data
Output: Hash value (Stored in HF)
Begin
Step 1: User uploads data into Data Node
Step 2: Start Processing
i. Hadoop WebHDFS REST API ← NodeJS. (To support Asynchronous function)
ii. WebHDFS-npm API ← NodeJS (To support various library functions)
iii. WebHDFS REST API LIST DIRECTORY ← File upload
iv. If the file is new

Step 3: Get the hash value
i. GETFILECHECKSUM ← Hash Value
Step 4: Get access time, and updation time and write it inHF
i. GETFILESTATUS ← Access time and updating time
ii. Write Hash value, Access time, and Updating time into HF
iii. CORNJOB ←NodeJS check change periodically.
End

5. Performance Evaluation

This section outlines the experimental setup, performance in terms of processing speed and data storage, and lastly the proposed model's limitations.

5.1. Experimental Setup

A machine with the Ubuntu operating system on Oracle VM Virtual Box was used. The setup was a single-node cluster with one NN and one DN. Hardware specifications were an Intel i5-12400CPU@2.5 GHz, 8GB RAM, 64-bit OS, X64-based processor, and 1TB of HDD. The software specifications were HF, Docker, Docker Compose, Hadoop, and NodeJS. Different types of files (.csv, .txt, .dat) ranging from 4 GB to 10 GB were used.Ubuntu is chosen as the operating system for this specific setup due to several reasons that align with the requirements and tools used in the environment, such as Compatibility, Virtualization Support, Single-Node Cluster, Resource Efficiency, Software Package Availability, Stability and Security etc

5.2. Experimental Results

The comparison between HDFS and the proposed model was performed with an input data size of 7 GB. The experiment was conducted with and without BC-related services. The results showed that the execution time of Wordcount was not significantly impacted by the additional services in the proposed model, indicating that there is little performance overhead in the proposed model. The result from this evaluation is shown in Figure 5 & Figure 6. Additionally, the memory utilization of Wordcount in HDFS was measured to be 6.96 GB, while in the proposed model it was measured to be 7.66 GB, as shown in Figure 7. This suggests that there is a slight increase in memory utilization in the proposed model, but it is considered an acceptable trade-off for the added security provided by the BC-related services. Overall, the results indicate that the proposed model can provide security enhancements without significantly impacting the performance of MapReduce jobs.The NodeJS client's performance was tested under various loads. Six files of different sizes, spanning from 1 megabyte to 110 gigabytes, were uploaded. The NodeJS client calculated the time it took to retrieve file metadata from HDFS and send this metadata to the client. The performance was compared and shown in Figure 8. Essentially, this test measures how well the NodeJS client can handle different loads on the proposed model and whether the Wordcount job has an impact on its performance. In the case of "without load", no job was running, while the Wordcount job was running in the case of "with load". When comparing the two cases, it was observed that HDFS takes extra time to produce the hash value in the case of "with load", particularly for larger files. In the meantime, HDFS was trying to access the file from the suggested model, but the increased workload on the disk caused delays and slower file opening or retrieval.

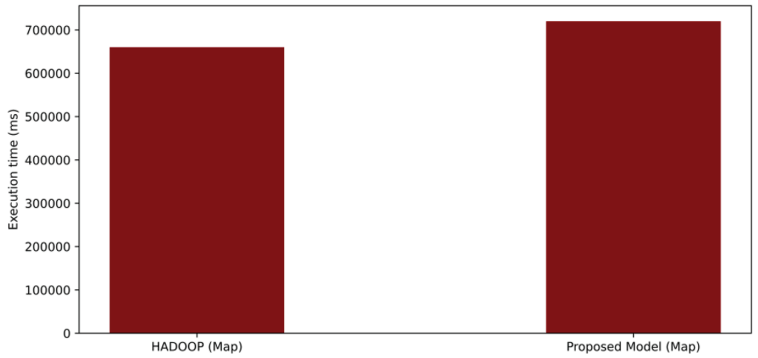

Figure 5. Processing time of map phase in HADOOP and proposed model

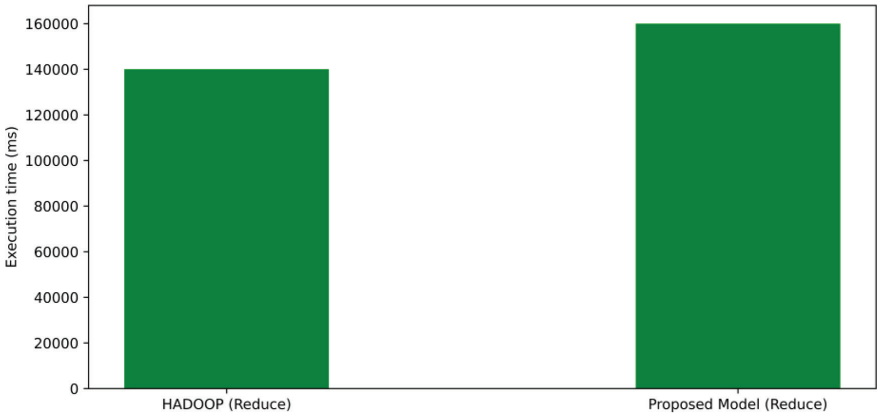

Figure 6. Processing time of reduce phase in HADOOP and proposed model

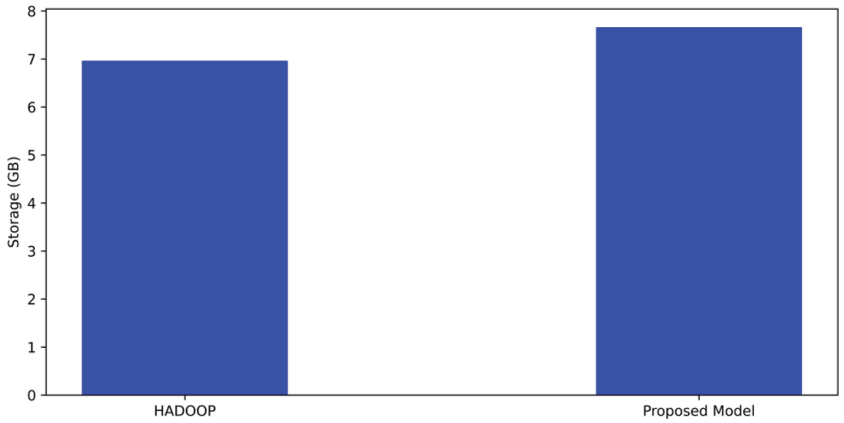

Figure 7. Memory utilization in HADOOP and the proposed model

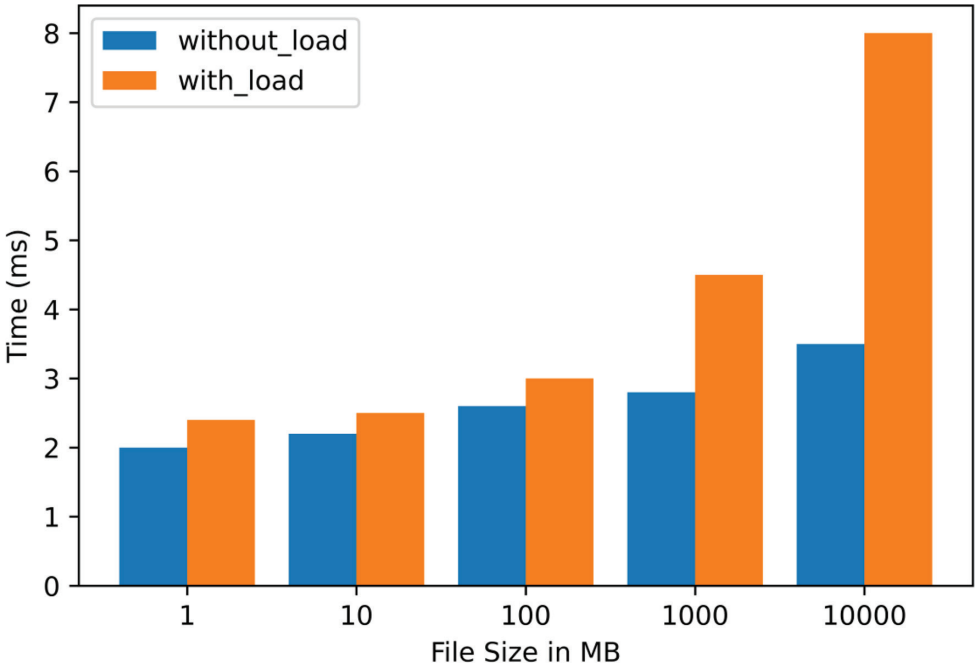

Figure 8. Client execution time

5.3 Comparative Analysis

Based on the information provided in Table 4, it appears that the proposed model generally outperforms the other models in the mentioned parameters. The proposed model demonstrates better performance in terms of storage overhead, communication cost, computation cost, encryption time, and decryption time compared to the other models listed in the table. It is important to note that the performance of a model depends on the specific requirements and context of the application. While the proposed model shows favorable results in this comparative analysis, further evaluation and consideration of other factors may be necessary to make a comprehensive assessment of its overall performance and suitability for a particular use case. Nevertheless, the information presented in Table 4 suggests that the proposed model exhibits better performance in the mentioned parameters compared to the other models listed.

Table 4. Comparative analysis of the performance of the proposed model and other models

Citation	Storage Overhead	Communication Cost	Computation Cost	Encryption Time	Decryption Time
V. Mothukuri et al. [12]	M	L	L	M	M
Z. Su et al. [20]	L	M	M	L	L
J. Wu et al. [21]	L	M	M	L	L

(continued)

Table 4. Comparative analysis of the performance of the proposed model and other models (continued)

Citation	Storage Overhead	Communication Cost	Computation Cost	Encryption Time	Decryption Time
C. H. Liu et al. [22]	H	M	H	H	H
G. Liu et al. [23]	M	M	L	M	M
X. Xu et al. [24]	H	M	M	H	H
U. U. Uchibeke et al. [25]	M	M	L	M	M
G. S. Aujla et al. [31]	M	M	M	M	M
Azzaoui et al. [32]	H	H	H	H	H
T. Mohanraj et al. [33]	L	L	L	L	L
M. K. Yousif et al. [34]	M	M	M	M	M
S. Guan et al. [35]	M	M	M	M	M
Proposed Model	**L**	**L**	**L**	**L**	**L**

L- Low, M- Moderate, H- High.

5.4. Security Implication

The proposed model aimed to enhance the security of HDFS by leveraging BC technology to log critical metadata. This helps prevent hacks and attacks that could compromise the files in the HDFS cluster. Any changes made to files are permanently recorded in the BC, providing a trustworthy and unchangeable record of file activity. This makes it easier for administrators to investigate any unauthorized changes or activity in the HDFS cluster. With the use of a NodeJS client, file changes can be easily monitored and recorded in the BC, providing a reliable way to track file authenticity. Overall, the proposed model is a low-cost and transparent solution for recording file changes and ensuring the security of HDFS.

5.5 Discussion

In the implementation of this model, BC is used to store the hash value of files. Firstly, data is encrypted by ES256 algorithms and then stored in HDFS. If the file is new, then the hash valueis calculated, and after that, this hash value is stored in the BC. The proposed model is tested only on a cluster and performances are calculated in terms of execution time. It may be possible that the proposed work would not have produced the same results when working on a real scenario-based project. The proposed work used HF to store hash value, and it is a well-known fact that the BC concept is immutable. So, when data increased, the length of the BC also increased, and it may be possible that it would be too large and would have decrease the performance as well.

6. Conclusion and Future Work

Integrating the BC concept with HDFS improves security. Traditional HDFS is more optimized for file processing but lacks security. So, to improve security, a new model has been proposed to integrate the BC concept with HDFS. In future research, the proposed model can be extended to real-world projects and multi-node clusters will be worked on as well.

By integrating BC with secondary name node metadata (fs image and edit log files) further improvement is possible. BC store metadata periodically and update their chain by combining fs image and editing log files. In addition to the above, due to the exponential growth of the data, BC may face scalability issues. Therefore, it is an open area of research, with much room for improvement.

7. Reference

Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al., 2018. Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains. *Proceedings of the Thirteenth EuroSys Conference*, Porto, April 2018, 1-15. https://doi.org/10.1145/3190508.319053

Apache hive, 2013. URL https://hive.apache.org/.

Aujla, G. S.; Chaudhary, R.; Kumar, N.; Das, A. K.; Rodrigues, J. J. P. C., 2018. SecSVA: Secure Storage, Verification, and Auditing of BD in the Cloud Environment. *Imminent Communication Technologies for Smart Communities*, pp. 78-85. https://doi.org/10.1109/MCOM.2018.1700379

Azzaoui, A. E. L.; Sharma, P. K.; Park, J. H., 2022. Blockchain-based delegated Quantum Cloud architecture for medical big data security. *Journal of Network and Computer Applications, 198*, 103304. https://doi.org/10.1016/j.jnca.2021.103304

BD Working Group; Cloud Security Alliance (CSA). Expanded Top Ten BD Security and Privacy, 2013, April. Available online:https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Expanded_Top_Ten_Big_Data_Security_and_Privacy_Challenges.pdf (accessed on 9 December 2015).

Cachin, C., 2016. Architecture of the Hyperledger Blockchain Fabric. Workshop on Distributed Cryptocurrencies and Consensus Ledgers. github.com/hyperledger/fabric

Cato, P.; Gölzer, P.; Demmelhuber, W., 2015. An investigation into the implementation factors affecting the success of BD systems. In *2015 11th International Conference on Innovations in Information Technology (IIT)*, pp. 134-139. https://doi.org/10.1109/INNOVATIONS.2015.7381528

Dong, X.; Li, R.; He, H.; Zhou, W., Xue, Z.; Wu, H., 2015. Secure sensitive data sharing on a big data platform. *Tsinghua Science and Technology, 20*(1), 72-80. https://doi.org/10.1109/TST.2015.7040516

Gantz, J.; Reinsel, D., 2011. *Extracting value from chaos- IDC view, 1142*, 1-12.

Guan, S.; Zhang, C.; Wang, Y.; Liu, W., 2023. Hadoop-based secure storage solution for big data in cloud computing environment. *Digital Communications and Networks*. https://doi.org/10.1016/j.dcan.2023.01.014

Gupta, M. K.; Pandey, S. K.; Gupta, A, 2022. HADOOP- An Open Source Framework for BD. *In 2022, 3rd International Conference on Intelligent Engineering and Management (ICIEM).* https://doi.org/10.1109/ICIEM54221.2022.9853179

Jindal, A.; Kumar, N.; Singh, M., 2020. A unified framework for BD acquisition, storage, and analytics for demand response management in smart cities. *Future Generation Computer Systems, 108*, pp. 921-934. https://doi.org/10.1016/j.future.2018.02.039

Khalid Yousif, M.; Dallalbashi, Z. E.; Kareem, S. W., 2023. Information security for big data using the NTRUEncrypt method. *Measurement: Sensors, 27*, 100738. https://doi.org/10.1016/j.measen.2023.100738

Lai, W., et al., 2014. Towards a framework for large-scale multimedia data storage and processing on Hadoop platform. *The Journal of Supercomputing, 68*(1), 1-20. https://doi.org/10.1007/s11227-013-1050-4

Liu, C. H.; Lin, Q.; Wen, S. 2019. Blockchain-enabled data collection and sharing for industrial IoT with deep reinforcement learning. *IEEE Transactions on Industrial Informatics, 15*(6), 3516-3526. https://doi.org/10.1109/TII.2018.2890203

Liu, G.; Dong, H.; Yan, Z.; Zhou, X.; Shimizu, S., 2022. B4SDC: A Blockchain System for Security Data Collection in MANETs. *IEEE Transactions on Big Data, 8*(3), pp. 739-752. https://doi.org/10.1109/TBDATA.2020.2981438

Mohanraj, T.; Santosh, R. 2022. Hybrid Encryption Algorithm for Big Data Security in the Hadoop Distributed File System. *Computer Assisted Methods in Engineering and Science, 29*(1-2), 33-48. https://doi.org/10.24423/cames.375

Mothukuri, V.; Cheerla, S. S.; Parizi, R. M.; Zhang, Q. "BlockHDFS: Blockchain-integrated Hadoop distributed file system for secure provenance traceability. *Blockchain: Research and Applications, 2*(1). https://doi.org/10.1016/j.bcra.2021.100032

Peters, G. W.; Panayi, E. 2016. Understanding Modern Banking Ledgers through Blockchain Technologies: Future of Transaction Processing and Smart Contracts on the Internet of Money. In *Banking Beyond Banks and Money* (pp. 239-278). Springer International Publishing. https://doi.org/10.1007/978-3-319-42448-4_13

Sarosh, P.; Parah, S. A.; Bhat, G. M.; Muhammad, K., 2021. A Security Management Framework for Big Data in Smart Healthcare. *Big Data Research, 25*, 100225. https://doi.org/10.1016/j.bdr.2021.100225

Sharma, P.; Borah, M. D.; Namasudra, S., 2021. Improving the security of medical big data by using Blockchain technology. *Computers & Electrical Engineering, 96*, 107529. https://doi.org/10.1016/j.compeleceng.2021.107529

Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R., 2010. The hadoop distributed file system. I*EEE 26th Sym- Posium on Mass Storage Systems and Technologies (MSST)*; 3–7 May 2010; Incline Village, NV, USA, IEEE, Piscataway, NJ, USA, 2010, pp. 1-10. https://doi.org/10.1109/MSST.2010.5496972

Su, Z.; Xu, Q., 2021. Security-aware resource allocation for mobile social BD: A matching-coalitional game solution. *IEEE Transactions on BD, 7*, 632-642. https://doi.org/10.1109/TBDATA.2017.2700318

Uchibeke, U. U.; Kassani, S. H.; Schneider, K. A.; Deters, R., 2018. Blockchain Access Control Ecosystem for Big Data Security. *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 1373-1378.

Viriyasitavat, W.; Hoonsopon, D., 2019. Blockchain characteristics and consensus in modern business processes. *Journal of Industrial Information Integration, 13*, 32-39. https://doi.org/10.1016/j.jii.2018.07.004

Wu, J.; Ota, K.; Dong, M.; Li, J.; Wang, H., 2018. BD analysis-based security situational awareness for smart grid. *IEEE Transactions on BD, 4*(3), 408-417. https://doi.org/10.1109/TBDATA.2016.2616146

Xu, L. D.; Viriyasitavat, W., 2019. Application of blockchain in collaborative internet-of-things services. *IEEE Transactions on Computational Social Systems, 6*(6), 1295-1305. https://doi.org/10.1109/TCSS.2019.2913165

Xu, X.; Zhang, X.; Gao, H.; Xue, Y.; Qi, L.; Dou, W., 2020. BeCome: Blockchain-enabled computation offloading for IoT in mobile edge computing. *IEEE Transactions on Industrial Informatics, 16*(6), 4187-4195. https://doi.org/10.1109/TII.2019.2936869

Zaharia, M.; Xin, R. S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M. J. et al., 2016. *Apache spark: a unified engine for BD processing, Commun. ACM, 59*(11), 56-65. https://doi.org/10.1145/2934664

Zhou, Z.; Wang,M.; Huang, J.; Lin, S.; Lv, Z., 2022. Blockchain in Big Data Security for Intelligent Transportation With 6G. *IEEE Transactions on Intelligent Transportation Systems, 23*(7), 9736-9746. https://doi.org/10.1109/TITS.2021.3107011