



Article

A Proposed Framework for Secure Data Storage in a Big Data Environment Based on Blockchain and Mobile Agent

Khalil Ahmad Alsulbi ^{1,2,*}, Maher Ali Khemakhem ² , Abdullah Ahamd Basuhail ², Fathy Eassa Eassa ², Kamal Mansur Jambi ² and Khalid Ali Almarhabi ¹ 

¹ Department of Computer Science, College of Computing at Alqunfudah, Umm Al-Qura University, Makkah 21514, Saudi Arabia; kamarhabi@uqu.edu.sa

² Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; makhemakhem@kau.edu.sa (M.A.K.); abasuhail@kau.edu.sa (A.A.B.); feassa@kau.edu.sa (F.E.E.); kjambi@kau.edu.sa (K.M.J.)

* Correspondence: kalsulbi0004@stu.kau.edu.sa

Abstract: The sum of Big Data generated from different sources is increasing significantly with each passing day to extent that it is becoming challenging for traditional storage methods to store this massive amount of data. For this reason, most organizations have resolved to use third-party cloud storage to store data. Cloud storage has advanced in recent times, but it still faces numerous challenges with regard to security and privacy. This paper discusses Big Data security and privacy challenges and the minimum requirements that must be provided by future solutions. The main objective of this paper is to propose a new technical framework to control and manage Big Data security and privacy risks. A design science research methodology is used to carry out this project. The proposed framework takes advantage of Blockchain technology to provide secure storage of Big Data by managing its metadata and policies and eliminating external parties to maintain data security and privacy. Additionally, it uses mobile agent technology to take advantage of the benefits related to system performance in general. We present a prototype implementation for our proposed framework using the Ethereum Blockchain in a real data storage scenario. The empirical results and framework evaluation show that our proposed framework provides an effective solution for secure data storage in a Big Data environment.

Keywords: Big Data; security; privacy; cloud storage; mobile agent; blockchain; Ethereum



Citation: Alsulbi, K.A.; Khemakhem, M.A.; Basuhail, A.A.; Eassa, F.E.; Jambi, K.M.; Almarhabi, K.A. A Proposed Framework for Secure Data Storage in a Big Data Environment Based on Blockchain and Mobile Agent. *Symmetry* **2021**, *13*, 1990. <https://doi.org/10.3390/sym13111990>

Academic Editor: Jeng-Shyang Pan

Received: 17 September 2021

Accepted: 18 October 2021

Published: 21 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The term “Big Data” is used to describe the massive amount of data generated from websites, social media platforms, the Internet of Things (IoT), multimedia archives, and other personalized services. Big Data can also be described as a set of complex or massive data that is difficult to manipulate with traditional database management systems [1]. Big Data are classified into three categories: unstructured, semi-structured, and structured data [2]. The data are often collected and stored in the different platforms listed above. The storage of Big Data has become a major topic of discussion, as people now realize the importance of data in supporting decision making. There has been a significant increase in the amount of Big Data, with an increase from the PETA-BYTE to ZETA-BYTE level in only the past two decades. The EMC and the IDC predicted that the amount of data would be more than 40 ZETA-BYTES in the year 2020 [3,4]. Newer forecasts by the IDC predict that amount will reach 75 ZETA-BYTES by the year 2025 [5]. Big Data gives people and companies advantages, such as being able to identify new trends, define trends, and make informed decisions. Despite these advantages, Big Data has also created issues, such as privacy and security issues, specifically issues to do with data availability, confidentiality, data privacy, integrity, and auditing and monitoring [6]. Because of the huge amount of data and its complex use arising from the aforementioned privacy and security issues,

traditional data management tools cannot process and store Big Data efficiently [7]. Many companies in today's world are using third-party cloud storage to handle their data. Cloud storage that is being leveraged by companies has many advantages including saving costs, quick data reliability and accessibility, the ability to replicate data for easy retrieval in case of disaster or loss, and workload balance [8]. However, cloud storage has some challenges arising from its nature of being owned by a third party, which leads to trust issues between the owner and the third party. Therefore, privacy and security concerns are seen as the main challenges to cloud storage [9]. Issues relating to secure transaction logs and data storage, security best practices for non-relational data stores, secure computations in distributed programming frameworks, and end-point input validation/filtering were listed as the main privacy challenges by the Cloud Storage Alliance Big Data working group (CSA) [10]. Other issues include real-time security monitoring, granular access control, granular audits, cryptographically enforced data-centric security, scalable privacy-preserving data mining and analytics, and data provenance. The contributions of this research can be summarized as follows:

- Designing an architecture that provides secure data storage and sharing using the Blockchain and Mobile Agent for Big Data applications. Proposed architecture related to Blockchain is based on two layers with two private Blockchains; to protect data, a data storage system by leveraging IPFS technology was adopted;
- Proposing a trustability check mechanism for the new joint requests to the Big Data system to ensure the basic level of security of the new device;
- Employing a two-level manager that deals with data fragmentation and storage to ensure data protection and preventing any illegal data retrieval;
- Developing the proposed solution to effectively and securely operate on local storage instead of cloud ones to gain a high level of data privacy;
- Designing a lightweight and generic solution that can be applied to various Big Data environments;
- Realizing a performance evaluation of the proposed framework based on metrics such as throughput and latency for read and write operations but also by making a security analysis and assessment of the developed prototype by comparison with other similar approaches existing in the literature.

The rest of the paper is structured as follows: Section 2 provides background knowledge of the technologies which used in our proposed system and presents a discussion of the available literature concerning privacy and security solutions used with Big Data. Section 3 presents a description of the recommended architecture. Section 4 presents an overview of the experiment and the details of the assessment and evaluation phase and provides a discussion of the simulation results. The last section, Section 5, presents the conclusions of the paper as well as a discussion of future work.

2. Background and Related Work

In this section, relevant background knowledge on Blockchain, Ethereum platform, smart contract, mobile agent, and security and privacy requirements in Big data frameworks are discussed in detail in the following subsections. Then, some related studies are presented.

2.1. Blockchain

Satoshi Nakamoto introduced Blockchain in 2008 [11] to underline the Bitcoin cryptocurrency network. It is a distributed database consisting of a digital ledger of transactions which used to store all committed transactions in a secure and immutable way. A Blockchain is built up of sequential blocks. Each block has a cryptographic hash value of the previous block as shown in Figure 1. Every hash identifies its block uniquely. Due to the many benefits of Blockchain, it can revolutionize many areas such as health care, education, banking, voting, etc. The most important of these benefits are decentralization, efficiency, auditability, traceability, transparency, immutability, and security [12].

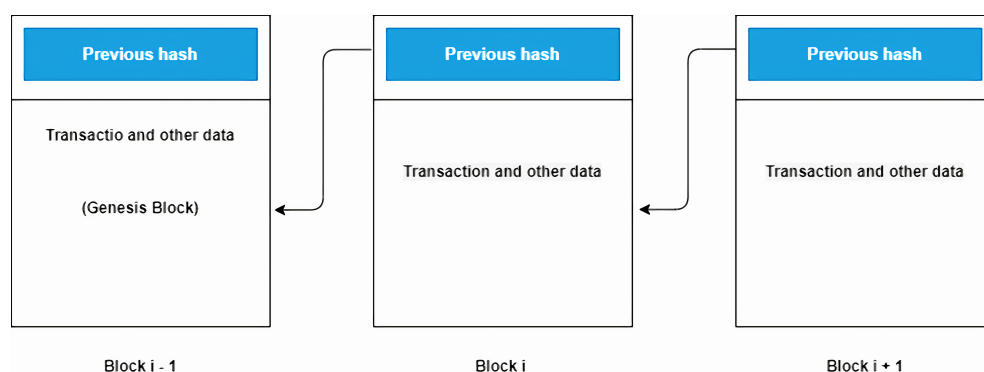


Figure 1. Blockchain structure.

Blockchains are divided into public, private, and consortium Blockchains [13]. In public Blockchains, anyone can participate in the network. They are fully decentralized and not controlled or regulated by any institution [14,15]. Private Blockchains are not open; they are only used within an institution [16]. Consortium Blockchains, which occur between the private Blockchain and the public Blockchain, are usually used when there are multiple user roles [17].

2.2. Ethereum

There have been numerous Blockchain platforms developed, most of them are focused on smart contracts and decentralized apps. These platforms differ in various aspects, including network type (public or permissioned), built-in cryptocurrency support, transaction workflow, performance, cost, and privacy. Ethereum is an open-source distributed Blockchain network introduced in 2015 by the Ethereum Foundation. Ethereum provides a smart contract platform which is the main feature of Ethereum. Ethereum also provides a language known as Solidity which enables programmers to customize their own Blockchain [18].

2.3. Smart Contract

A smart contract is an attractive attribute of Blockchain. It is an executable code deployed on the Blockchain network. A smart contract is triggered by sending a transaction to it. Smart contracts integrate with Blockchain to do a task in real-time with a high degree of security. This integration helps to develop, design, and implement real-world problems in less time and cost without a third party [19]. A smart contract consists of the address, value, state, functions, and state as shown in Figure 2.

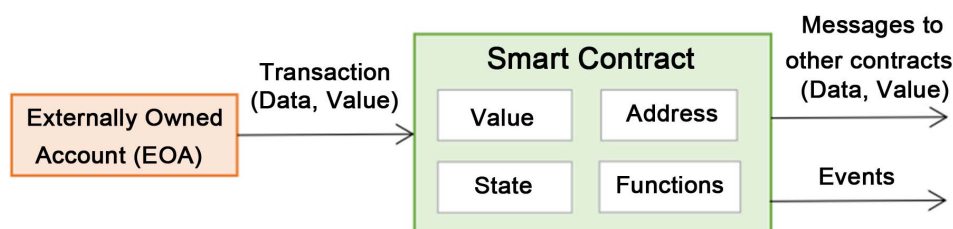


Figure 2. A basic structure of a smart contract [20].

2.4. Mobile Agent

A software agent is a computer program that works on behalf of another entity to achieve goals in a dynamic environment (human or computational) [21]. A mobile agent is one of the types of agent software which is a program that can migrate through the machines of the network to accomplish some tasks on behalf of some user as shown in Figure 3. A mobile agent has numerous advantages, the most prominent of which are as follows:

- They enable the cost-effective and efficient use of communication networks with high latency and low bandwidth.
- They make it possible to utilize portable, low-cost personal communications devices to accomplish sophisticated activities even when they are disconnected from the network.
- They allow true decentralization and asynchronous operations.

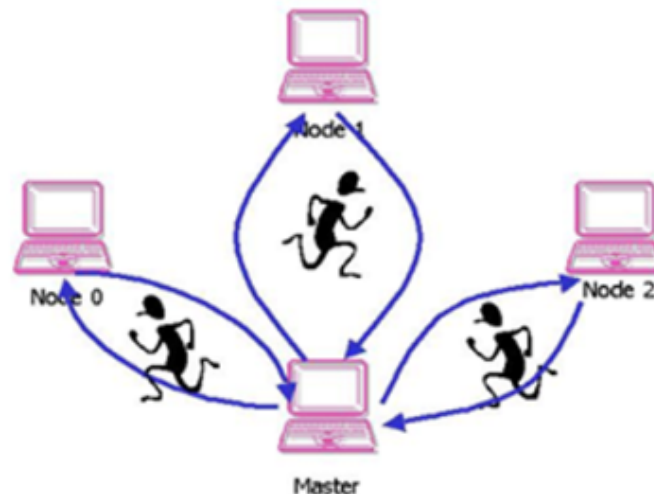


Figure 3. Mobile agent [22].

2.5. Security and Privacy Requirement in Big Data Frameworks

The researchers investigated the most recent trends with regard to privacy and security issues in the Big Data environment [23]. An analysis of the comprehensive and essential requirements used in the development of privacy and security framework was conducted. The requirements explored were as follows:

2.5.1. The Trustability of New Nodes Connected to the Big Data Environment

The initial phase of developing a private and secure framework for a Big Data system starts with authenticating the trustability of new nodes used in the security system. Nodes that are not trusted usually pose a threat to the system, as they may allow invaders and hackers to access data stored in that system. In a typical Big Data system, there are many devices and nodes linked to the system, and it is important to check their trustability before allowing them to connect to ensure the protection of the system. Such systems have security requirements, for example, having active antivirus protection and having the latest update for their operating system for new nodes, and devices and nodes are only allowed to join if they fulfil these requirements.

2.5.2. Access Control Enforcement

One of the main aspects of Big Data with respect to privacy and security is Access Control (AC). It is important for individuals and organizations dealing with Big Data to ensure that they have an access control policy. Access control policies govern the connection of different nodes to the system. Having a crooked access control policy can allow hackers to gain illegal access to the stored data, hence posing privacy and security issues [24]. AC policies help to protect data by providing privacy and security permissions for nodes and devices. There has been a significant research focus on access control policies, but potential issues still need to be addressed.

2.5.3. Integrity and Security of Data during Transfer and Storage

Another aspect of ensuring the privacy and security of Big Data systems is guaranteeing data protection in the transmission and storage phases. This entails ensuring data integrity (DI), and the accuracy and consistency of data. The main causes of data integrity challenges are related to users, software, hardware, and intruders [10]. Sometimes, Big Data are stored in locations that are not known, and in such cases, DI becomes an issue of concern. In order to maintain integrity, data should be kept secure in all phases of transmission and storage.

2.5.4. Policy Protection

Policies in the Big Data environment describe the sets of rules used to control, transfer, and access data as they move from one place to another. Attackers often alter policies in an endeavor to gain illegal access to data. For most data owners, it is difficult to know whether a policy is illegal or legal or if it has been changed during transmission, storage, or processing. It is important to determine a secure way to safeguard policies from being modified, as most previous studies only focused on data protection with little attention given to policy protection.

2.5.5. Data Privacy

Data privacy entails not sharing the personal data of an individual without their explicit approval. Big Data may comprise personal or sensitive data related to individuals, and hence it is important to ensure that these are not shared without approval. Even obtaining approval from an individual is limited to specific and necessary reasons. Therefore, to ensure the privacy of data, datasets should be free of personal identifying attributes, such as names and addresses.

Previous research examining solutions to privacy and security issues in Big Data determined that five requirements need to be satisfied. Shubhi Gupta et al. [25] recommended a solution entailing the use of the Elliptic Curve Cryptography (ECC) algorithm. Here, Big Data are classified into sequential parts on the basis of same or IP-resembling data types and titled alphanumerically. The ECC algorithm is implemented with minimal complexity, because it uses a small key size and still provides a high level of efficiency. This algorithm helps to enhance the security of data access as well as reliability and scalability. Another study by Yiu Chung et al. [26] recommended a solution involving the use of the Secure Pattern-Based Data Sensitivity Framework (PBDSEF) to safeguard the security of sensitive information in healthcare settings. This solution employs machine learning to secure information by defining a set of common characteristics of patient information including the data frequency and different symbol patterns used. The framework also utilizes a Hadoop that performs tasks such as identifying sensitive data and encrypting it. Tests on this framework have revealed that it has a fast response time and achieves a high level of security for sensitive data. The solution could be made better by adding privacy safeguard techniques when transmitting data across the healthcare sector.

Zeyad A. Al Odat et al. [27] also recommended a solution that relies on splitting Big Data into small chunks and allotting them into different cloud locations. This solution comprises two steps, with the first being the distribution and retrieval of the data. The second step involves the use of a secure hash algorithm to verify data after they are retrieved from the cloud. This solution has had good results in terms of its high speed and high level of security.

Yinghui Zhang et al. [28] recommended that in order to achieve Big Data security, cloud storage should be used. For this solution, the privacy of users' data is guaranteed by the cloud system. However, this solution has a leakage rate of 1/3, which is greater than that of any other solutions discussed above. The results also show that this system is mostly suitable for data stored in the cloud alone.

Kai Fan et al. [29] also proposed a solution for Big Data security based on key hierarchical management. For this scheme, there are three levels of keys: lower, middle, and upper

keys. The approach uses keys to guarantee security, with the upper key encrypting the middle key and, in turn, the middle key encrypting the lower key. This solution works by transferring data to the cloud after encrypting it on the side of the client. However, this system has some weaknesses with there being a need to enhance key distribution and exchange, reduce the time used in encryption, and enhance encryption and decryption behavior on the side of the server.

Gagangeet Singh Aujla et al. [15] suggested a novel schema that offers secure storage of Big Data in the cloud. This solution has different entities, namely, a data service provider (DSP), client, trusted party auditor (TPA), cloud service provider (CSP), and Kerberos server. The DSP provides data that are segmented in blocks and encrypted with an attribute-based encryption (ABE) algorithm. These encrypted data are then sent to the CSP and stored at a public tag, while their associated tag is stored in a private cloud. This system helps to maintain data integrity and can withstand different types of attacks.

Muqaddas Naz et al. [30] presented a Blockchain-based secure data sharing and delivery of digital assets framework to provide quality of data and their authenticity and a stable business platform for the owner. This framework used decentralized storage IPFS storage to address bloating problem at the owner's end. To ensure data authenticity a review-based system has been used in this framework to able the customers to add their comments about the data. Different smart contracts are developed such as owner smart contracts and access smart contracts. This solution has good results in terms of the least computational time for encrypting the shares.

Conghui Zhang et al. [31] proposed a new schema to solve some problems in Hadoop such as single point failure and scalability. The proposed schema uses Blockchain to prevent tampering with metadata. The main idea of this schema is to distribute the Namenode server using Blockchain to protect metadata and manage access tasks of users. The experiments show that this mechanism can restrict malicious access and meta-data anti-tamper.

Shubham Desai et al. [32] proposed a new approach of secure IPFS cloud storage system using Blockchain. The idea of this approach is to store the data in the cloud, while the information identifying the file, will be stored on the Blockchain. The data will be encrypted before sending to storage using AES256 encryption and the data owner provide the encryption key. The evaluation results demonstrate the proposed approach can provide a good solution for developing a data-sharing platform based on cloud storage.

Jian Chen et al. [33] suggested a Big Data management system based on Blockchain technology. The idea of the proposed model is to divide data into core data and non-core data. The core data and hash result of the non-core data will be stored in Blockchain while the non-core stored in the central database. The new model can solve the data redundancy problem and insufficient storage space. Blockchain is used to improve the security of big data storage because of its characteristics such as decentralization and non-tamper ability. However, there are some drawbacks of Blockchain, the most important one is limited storage space. So it cannot store large amounts of data.

Jung et al. [34] developed a data management system that uses Blockchain to store the transactions that contain data item name, and IP address, port number, and signature of the data generator. To locate the target data item, transactions can be searched by name and verify ownership. Blockchain improves security as the results show that the proposed system outperforms the previous storage systems.

Unlike previous studies [15,25–34], this study focuses on not only access control and data storage but also develops an effective data management strategy and comprehensive solution for data security and privacy in a Big Data environment with a smart contract and Blockchain. It is worth noting that the use of a Blockchain to solve security and privacy problems in a Big Data environment has previously been demonstrated in various practical scenarios, such as [30–34].

However, research on the development of a comprehensive solution to address the issues facing Big Data security and privacy is still lacking. This study aims to fill this gap

and provide a comprehensive solution that addresses Big Data security and privacy issues. The main differences between our study and previous studies are as follows:

- We propose a mechanism that checks the basic level of security for any new device that needs to connect to the system. Hence, we ensure the trustability of the device;
- We propose a two-level manager that deals with data fragmentation and storage. Therefore, data passes through two levels of fragmentation which makes data retrieval complicated for intruders and malicious requests, as detailed in Section 3.1.2. Furthermore, the two-level manager prevents and direct connection between users and data as discussed in Section 3.1.2;
- We implement a private Blockchain with a smart contract in each level of the manager to store the meta-data and policies related to each level. Consequently, data tampering and malicious access are prevented;
- We develop our system to be deployed in local storage to ensure a high level of data privacy, unlike most existing studies that rely on cloud storage.

Considering the various studies explored in this section, each study focused on a single issue and hence could not offer a comprehensive solution to address the issues facing Big Data security and privacy. Therefore, the solutions offered by the studies warrant further study. The current study aims to integrate these solutions to obtain a comprehensive one.

3. Materials and Methods

3.1. System Design and Architecture

We propose a new security framework for Big Data based on the Blockchain and mobile agents, which works in a multi-cluster environment. It is distributed in every main node in every cluster, as shown in Figure 4. We have a number of clusters, C1, C2, C3, and Cn, and every cluster has a number of storage nodes, SN1, SN2, and SNn. The main idea of our proposed system is to store the meta-data and policies and every transactional piece of data in the Blockchain. The actual data are distributed in storage nodes in the interplanetary file system (IPFS). Table 1 shows the notations used in this section which will help the reader to understand the figures.

Table 1. Notations.

Symbols	Meaning
M	Manager
HLM	High-Level Manager
LLM	Low-Level Manager
HFM	High Fragmentation Module
IPFS	Interplanetary File Storage
MD	Meta-Data
HLBC	High-Level Blockchain
LLBC	Low-Level Blockchain
HPF	High Policy File
LPF	Low Policy File
CSMA	Check Security Mobile Agent
LFM	Low Fragmentation Module

We also suggest the use of mobile agents to achieve advantages such as the economical and efficient use of communication channels that may have high latency and low bandwidth.

Our proposed framework consists of three layers, a user interface layer, Blockchain layer, and storage layer, as shown in Figure 5.

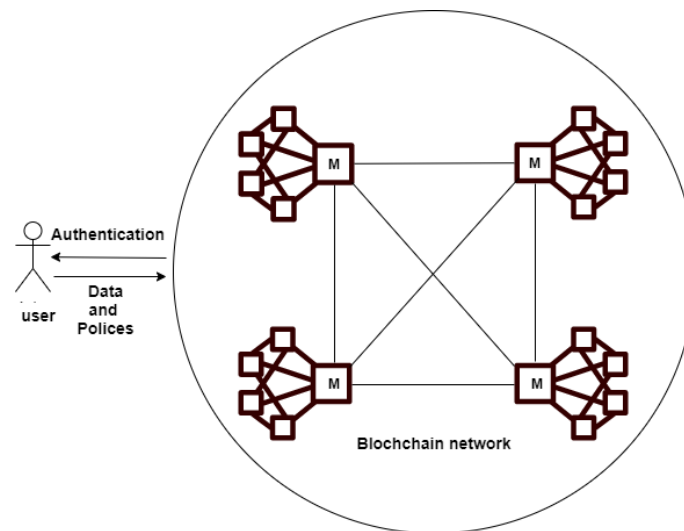


Figure 4. High-level architecture of the proposed framework.

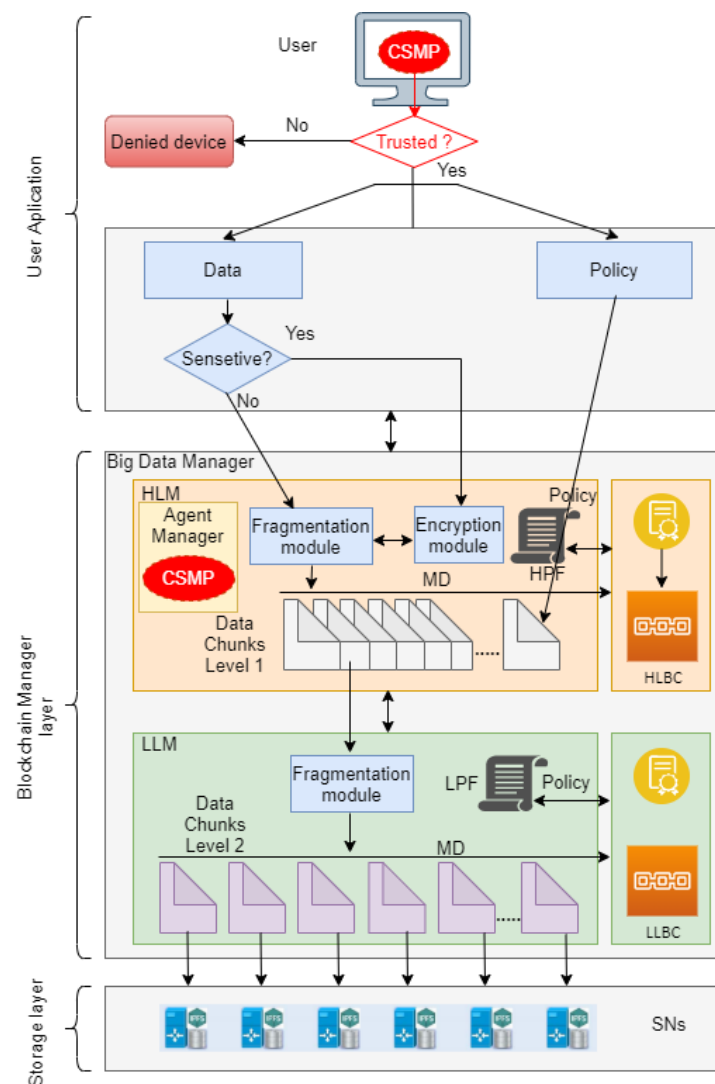


Figure 5. The design of the proposed framework architecture.

3.1.1. User Application Layer

The first layer involves user applications (data producers). It contains an interface to allow users to create an account and upload their data and policies.

3.1.2. Blockchain Manager Layer

The Blockchain layer (Blockchain network) is the core of the proposed system. It contains the proposed manager, which manages all of the components. The proposed manager consists of two layers with two private Blockchains which are high-level and low-level managers. The two levels are connected to each other, as described below:

1. High-level manager (HLM): The high-level manager deals with users requests. Therefore, this layer receives any new joint request and verifies the trustability of the requesting device using a mobile agent. Moreover, the HLM keeps meta-data of all clusters to easily and securely manages the clusters connections. Therefore, any untrusted or malicious cluster will not be able to join the Big Data system or communicate with other clusters. In addition, the high-level manager fragments received data into multiple chunks to be assigned to multiple clusters. Besides, HLM encrypts received sensitive data before the fragmentation. A high-level manager consists of six components that work cooperatively as demonstrated below:
 - Mobile agent named *check security requirements agent*. This agent is generated by the agent manager in high-level manager to migrate to any new node that wants to connect to the big data system as shown in Figure 6. This mobile agent checks to see if new nodes are trusted by detecting whether they meet security policy requirements, such as updated antivirus software. In general, this agent will determine if the node can connect to a Big Data system or not.
 - A high fragmentation module, which divides the data into chunks to distribute them on clusters.
 - A high-level policy file, which contains information about clusters in the system and a set of rules established by the data owner that determines who can access their data.
 - A high-level Blockchain, which stores the high-level meta-data and high-level policy and all transactions performed in the high-level manager.
 - Smart contract named *HighMasterNode*. This smart contract performs many functions, the most important of which are:
 - Creating a high-level policy file and storing it in high-level Blockchain.
 - Storing meta-data that were created after data fragmentation by the high-level manager in the high-level Blockchain.
 - Encryption module: this module encrypts any received sensitive data using Advanced Encryption Standard.
2. Low-level manager (LLM): The low-level manager keeps the meta-data of storage nodes of the corresponding cluster. Thus, communication between storage nodes is handled through the LLM. As a result, untrusted or malicious nodes are not able to join the cluster. Additionally, the LLM fragments received data from the HLM into multiple smaller chunks and stores them in multiple storage nodes. The low-level manager is made up of four components as described below:
 - A low fragmentation module, which divides the chunk into smaller chunks and distributes them among storage nodes in the cluster.
 - A low-level policy file, which contains information about the nodes in the clusters and includes rules that specify whether data can be transferred from one node to another.
 - A low-level Blockchain, which stores the low-level meta-data and low policy file and all transactions performed in the low-level manager.
 - Smart contract named *LowMasterNode*. This smart contract performs many functions, the most important of which are:

- Creating low-level policy file and storing it in low-level Blockchain.
- Storing meta data that was created after data fragmentation by the low-level manager in low-level Blockchain.

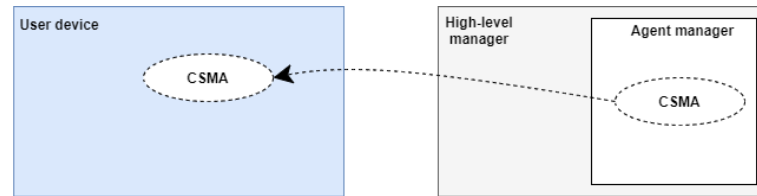


Figure 6. Check security requirements agent.

3.1.3. Storage Layer

The last layer is the storage layer, which consists of a distributed storage node that is used to store actual data.

3.2. Workflow of the Proposed Solution

The system uses the following steps, as shown in Figure 7:

1. Step 1: Initialization (executed by the user device):
The user can create an account on the metamask after it has been ensured that their device meets the security requirements. These requirements can include having an active antivirus program, having the latest operating system update, and so on. These requirements can be checked through the check security requirements agent. The user's account address is fetched in the application through web3.js from the metamask. The user can upload data files and policies to the high-level manager. Sensitive data are encrypted using symmetric-key encryption and transfer, while the transfer of normal data is done without encryption.
2. Step 2: Uploading data to the Blockchain manager (executed by the high-level manager):
The high-level manager receives the data and divides it using the high fragmentation module into equal n chunks. Then, every chunk is sent to a specific cluster after meta-data of this chunk are created. The meta-data and policy are stored in the high-level Blockchain manager through a smart contract. The high-level manager receives the policies and stores them in the high policy file.
3. Step 3: Uploading data to storage nodes (executed by the low-level manager):
The low-level manager receives the chunk of data sent to this cluster and divides it using the low fragmentation module into equal n chunks. Every chunk is stored in the IPFS in a corresponding storage node in this cluster after its meta-data have been created. The meta-data are stored in the low-level Blockchain manager through a smart contract.

Note that, in this context, the meta-data and policy are stored in the private Blockchain, and any modifications to data files in the IPFS or transfers can be detected by the manager.

Algorithm 1 gives the basic flow of system processes and Algorithm 2 gives the processes of data fragmentation.

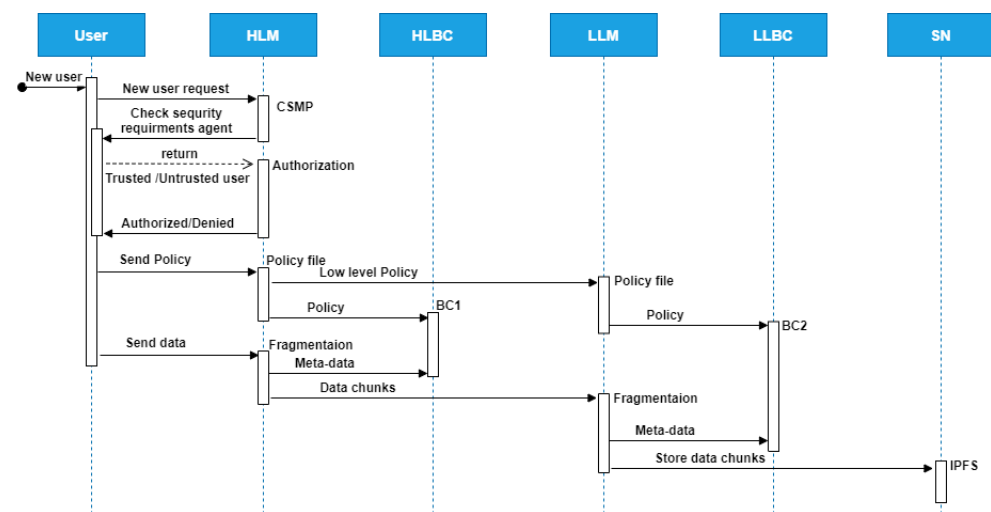


Figure 7. Logical flow execution of the system.

Algorithm 1 General workflow

```

1: User send a request to selected HLM node to join the system
2: the HLM node send a mobile agent to user device
3: if user device meet security requirements then
4:   Register user in the system
5:   User upload the data file
6:   if Data is sensitive then
7:     encrypt(data)
8:   end if
9:   Determine (data policies)
10:  Send(data) and (data policies) to HLM node
11:  Fragmentation (data) into (data chunks) in HLM node
12:  Create (meta data)
13:  Send(data chunk) to LLM node
14:  Update Blockchain1(HLBC) to store meta data
15:  Fragmentation (data chunk) into (data chunks) in LLM node
16:  Create (meta data)
17:  Send (data chunk) to storage node(IPFS)
18:  Update Blockchain2(LLBC) to store meta data
19: else
20:   Return false
21: end if
  
```

Algorithm 2 Data fragmentation

```

1: Upload file
2: Read file from Blockchain
3: if encryption == true then
4:   Encrypt file
5: else
6:   go to step 8
7: end if
8: Let  $N$  = number of nodes
9: Split(file) to  $N$  chunks
10: for  $i = 0 \dots N$  do
11:   Send chunks  $i$  to Node  $i$ 
12: end for
  
```

4. Results and Discussion

4.1. Implementation Details

To validate the solution, the proposed system was implemented and tested. The implementation detail of our proposed framework is described in this section. The implemented prototype has two parts. The first part is the client interface (Front-End), and the second part is the security manager based on the Blockchain and mobile agents (Back-End). For the development of the security manager, a Blockchain platform was required. Various Blockchain platforms exist, such as Hyperledger Fabric, Ethereum, IOTA, Ripple, IBM Blockchain, and OpenChain.

We developed our framework based on the private Ethereum platform. The Big Data management system involves a lot of personal and sensitive data. Therefore, private Blockchain was used as the basic Blockchain architecture of our proposed system. Ethereum has some benefits such as supporting Decentralized Applications, providing a friendly environment for users, and using Solidity language, which is a complete scripting language, to implement smart contracts. Ethereum has a better number of validated transactions in one second than Bitcoin [35]. The important part of the prototype system is the smart contract. We developed it using Solidity language. The front-end web GUI was developed using Metamask and Javascript to create user-interactive forms.

The system runs on the Ubuntu 20.04 (64-bit) operating system with Intel(R) Core(TM) i7-1035G1 CPU @ 2.8 GHz, and 16 GB RAM. It uses the IPFS to simulate storage nodes. Below, we describe the primary tools used to develop our system:

Ganache

Ganache is a type of software that provides virtual accounts to execute smart contracts based on the Blockchain. It is a Blockchain-based simulator used to deploy smart contracts and execute several tests and commands. Ganache stores a unique address for each account and performs mining [36].

Metamask

Metamask is one of the most secure ways to connect to Blockchain-based applications. The user uses the Metamask interface to connect to the Ethereum Wallet [37].

Node.js

Node.js is a back-end JavaScript run-time environment and cross-platform that executes JavaScript code [38].

Truffle Framework

The Truffle framework is an Ethereum-like Blockchain simulation platform used for building, testing, and deploying applications. It is used to compile and deploy smart contracts in this system [39].

IPFS

The IPFS is the Interplanetary File System, a protocol used to store and share Big Data in peer-to-peer networks in a distributed file system. We used the IPFS to store data files to allow flexible and reliable decentralized data storage in a corresponding storage node.

4.2. Performance Evaluation

In this section, we evaluate the performance of the proposed framework to alleviate concerns associated with this new technology.

4.2.1. Performance Evaluation Settings

Testing environment

We conducted experiments using the following configurations to test the performance of the proposed framework:

We used four virtual machines (VMs) running on Google Cloud to provide the

infrastructure for the private Blockchain. Each VM had a 2 GHz and 4-core Intel i7 CPU. Additionally, we used Hyperledger Besu v2.21.2, an open-source Ethereum client that provides a permissioned private Blockchain to build the private Blockchain [40]. In terms of a Peer-to-Peer network, we used one validator node, three peer nodes, and the Clique Consensus Protocol. We use Hyperledger Caliper v0.3.1 which is a benchmark tool used to evaluate the performance of the Blockchain system [41]. The Caliper tool provides a performance report containing several performance indicators, such as resource utilization, transaction latency, transaction throughput, read throughput, read latency, etc. The programming language is JavaScript and solidity.

Experimental settings

For the write operation, we used the following settings:

The number of test rounds was 8 rounds with 500 transactions per round. The type of control rate was fixed-rate. Finally, the used send rates were 10 tps, 50 tps, 90 tps, 130 tps, 170 tps, 210 tps, 250 tps, and 290 tps. The same setting was used for read operations.

4.2.2. Performance Evaluation Metrics

This section illustrates the main metrics we utilized to evaluate our proposed system. Throughput and latency are two common performance measures used to assess the performance of a Blockchain network [42]. The throughput can be divided into two categories based on the type of operations being dealt with. Read throughput refers to the number of read operations completed within a specific period of time, and this metric is expressed as reads per second (RPS). Write (Transaction) throughput refers to the rate at which valid transactions are committed by the Blockchain system in a specific period of time. This rate is expressed as transactions per second (TPS).

$$\text{Read throughput} = \text{Total read operations} / \text{total time in seconds} \quad (1)$$

$$\text{Transaction throughput} = \text{Total valid transactions} / \text{total time in seconds} \quad (2)$$

Latency also can be divided into two subcategories based on the type of operation. Read latency refers to the time from when the read request is submitted to when a reply is received. Transaction latency refers to the time taken for a transaction's effect to be usable across the network.

$$\text{Read latency} = \text{Time when response received} - \text{submit time} \quad (3)$$

$$\text{Transaction latency} = (\text{Confirmation time} * \text{network threshold}) - \text{submit time} \quad (4)$$

4.3. Performance Assessment

We conducted a performance evaluation using Hyperledger Caliper v0.3.1, which is an open-source benchmarking tool used for evaluating the performance of Blockchain applications in order to understand how our proposed framework would perform in real-case scenarios [42]. Figure 8 shows the experimental results in terms of the throughput and latency for the write operation.

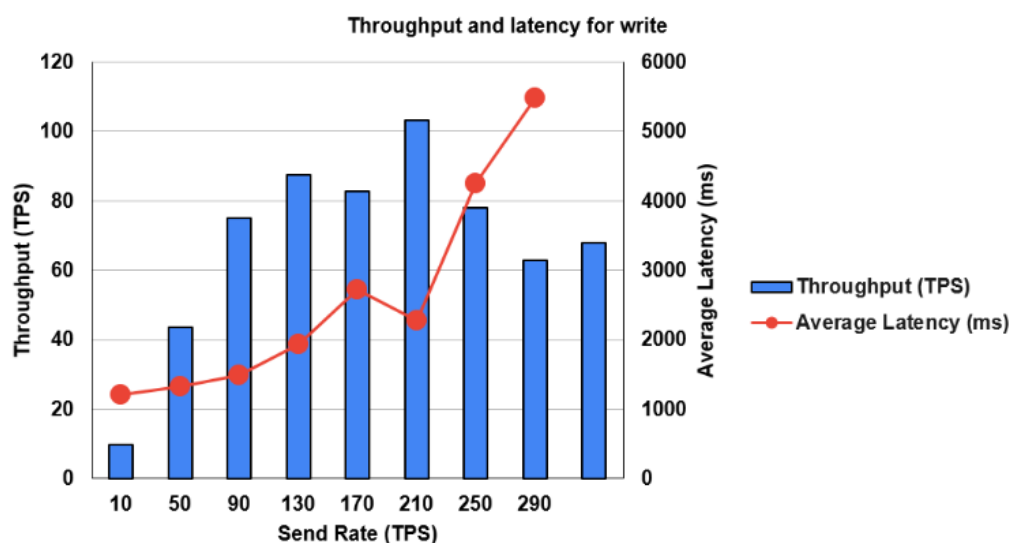


Figure 8. Throughput and latency for the write operation.

We evaluated the system's performance over different transaction send rates (10–290 tps). The transaction throughput grew exponentially as the send rate increased, until the send rate reached about 130 tps. When the send rate was increased from 130 to 170 tps, the transaction throughput growth slowed. Then, the transaction throughput increased again until the send rate reached 210. Finally, the transaction throughput began to decrease gradually. The average write operations throughput was 67.85 tps, as shown in Figure 8. Additionally, with the increase in the send rate, the transaction latency increased until the send rate was 170 tps. Then, the transaction latency decreased when the send rate was from 170 to 210 tps. The transaction latency began to increase gradually after the send rate increased above 210 tps. The average latency in the write operations was 2.58 s. Figure 9 shows the experimental results in terms of the throughput and latency for the read operation.

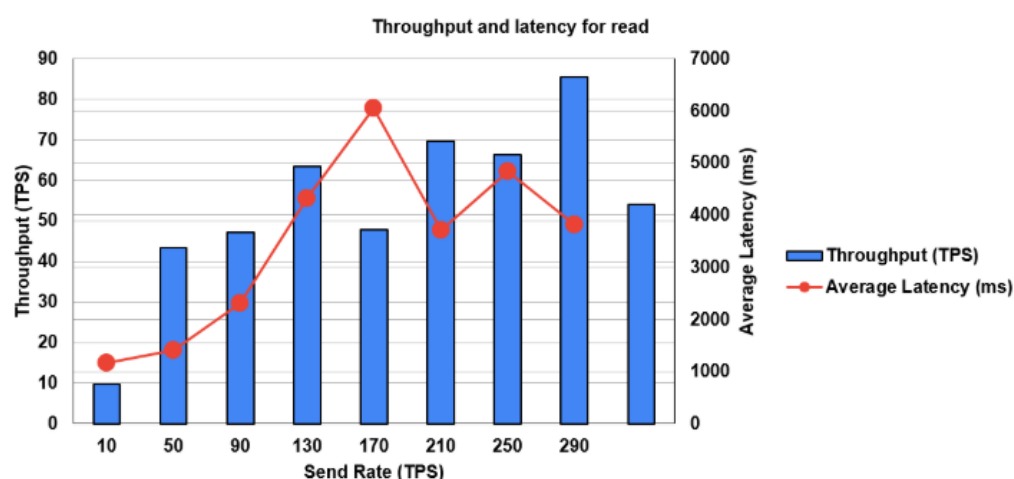


Figure 9. Throughput and latency for the read operation.

Figure 9 indicates that the transaction throughput rose exponentially as the send rate increased until it reached about 130 tps. When the send rate was increased from 130 to 170 tps, the transaction throughput dropped. Then, the transaction throughput began with a series of oscillations up and down when the send rate increased from 170 to 290 tps. The average read operations throughput was 54.1 tps. The transaction latency increased rapidly until the send rate reached around 170 tps. Then, the transaction latency began to exhibit a series of oscillations up and down when the send rate increased from 170 to

290 tsp. The average read operations latency was 3.45 s. The results show that the optimal send rate for write operations is 210 tps where the system latency is 2.27 s with a system throughput of 103.3 tps. On the other hand, the optimal send rate for read operations is 210 tps where the latency is 3.71 s and the throughput is 69.7 tps.

4.4. Security Analysis of the Proposed Framework

Security is always a great concern for Big Data storage and sharing systems where data must be protected from threats to guarantee data privacy and network security. We provide evidence of the security of our proposed framework through the following theorems.

Theorem 1. *Assume that a malicious node tries to join the system network to steal the data; such a node may find it impossible to join the network.*

Proof of Theorem 1. In our proposed framework, before any node or new user joins the system, it is tested and must meet a set of security requirements. This prevents untrusted devices from communicating with the Big Data system. Additionally, information from new nodes is recorded in the policies file that is published securely on the Blockchain, and any malicious node is prevented from joining the system because it is not registered in the policies file. □

Theorem 2. *Assume that a malicious node can access the data storage system; such a node may find it very difficult to retrieve data from our system.*

Proof of Theorem 2. In our proposed system, sensitive data uploaded by users are encrypted using symmetric key encryption. Data are stored in distributed storage nodes. The system returns the meta-data of these data and is stored in the Blockchain. Therefore, it is very difficult to get data that are stored in multiple storage nodes because their metadata are securely stored in the Blockchain and are inaccessible. □

Theorem 3. *Assume that a malicious node tries to access and tamper with policies; such a node may find it very difficult to access policies and tamper them due to our system.*

Proof of Theorem 3. In our proposed system, all policies that are imposed on the data by the owner of the data and the system, for example, the ability to use these data and transfer them from one node to another after storing them, are securely stored in the Blockchain, which prevents tampering. □

Theorem 4. *There is no third party in our system, which ensures data privacy.*

Proof of Theorem 4. In our proposed system, external parties, such as cloud storage platforms, are eliminated. The data are stored in an internal local system, which preserves the data from being used by external parties and thus maintains the privacy of the data. □

4.5. Comparison of the Proposed Framework with Related Work

We discuss comprehensive security and privacy requirements in the related works section. In this section, we compare our framework with related work in this domain based on these requirements. In contrast to existing works, our proposed framework offers the ability to verify new nodes to ensure that there are no malicious nodes in which attackers can access data through them.

Our proposed framework enforces access control policies to restrict the access of data to specific users to ensure the system's security. Access policies are enforced through several phases. The first is to ensure that new devices meet minimum security requirements, and this is followed by the authentication phase and the authorization phase. After verifying the trustability of the new users, they join the Blockchain system as new nodes, and information about new nodes is recorded in the policies file that is published securely

on the Blockchain. Malicious nodes cannot join the system because they are not registered in the policies file.

In our proposed system, sensitive data uploaded by users are encrypted using symmetric key encryption. Data are stored in distributed storage nodes. The system returns meta-data of these data, and the meta-data are stored in the Blockchain. The Blockchain ensures data integrity, because it prevents data tampering.

Our proposed system offers policy protection. All policies imposed on the data by the owner of the data and the system, such as the ability to use the data and transfer them from one node to another after storage, are securely stored in the Blockchain, preventing their modification and manipulation.

In our proposed system, external parties, such as cloud storage platforms, are eliminated. The data are stored in an internal local system, which preserves the data from being used by external parties and thus maintains their privacy.

We evaluated the security level reported by six existing works on data storage management systems for Big Data (Table 2) based on five requirements (as shown in Figure 10): the trustability of new nodes, the enforcement of access control policies, the security and integrity of data, policy protection and data privacy [23].

Table 2. Comparison of related works and the proposed framework.

Paper citation	Trustability of New Nodes Connected to the Big Data Environment	Access Control Enforcement	Security and Integrity of Data During Storage and Transportation	Policy Protection	Data Privacy
Shubhi [25]	N	Y	Y	N	Y
Yiu [26]	N	Y	Y	N	N
Zeyad [27]	N	Y	Y	N	N
Yinghui [28]	N	Y	Y	N	Y
Kai [29]	N	Y	Y	N	Y
Aujla [15]	N	Y	Y	Y	Y
Proposed framework	Y	Y	Y	Y	Y

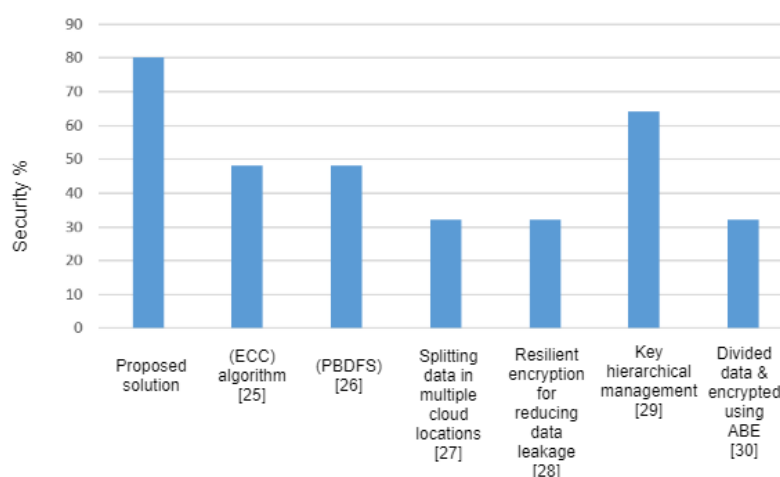


Figure 10. The security level of Big Data storage systems.

5. Conclusions

This study presented a solution for enhancing Big Data security and privacy. We identified critical challenges for current Big Data storage systems and proposed an efficient solution to address these challenges through the implementation of a real prototype. Our focus was to design a trustworthy and comprehensive framework based on a Blockchain that can be used in Big Data environments to ensure efficient and secure data storage and sharing. To evaluate the performance of the proposed solution, we deployed an Ethereum

Blockchain on the Google cloud. Additionally, to achieve decentralized data storage, we integrated the Blockchain with the distributed IPFS storage system. We used two private Blockchain layers to store metadata after splitting data into several chunks, and we used data access policies to ensure the security of metadata and policies, which greatly improved Big Data security and privacy and prevented the manipulation of policies. We also used a mobile agent to check the security requirements of the new user device to achieve benefits such as reducing traffic overheads and typifying the high level of intelligence and mobility of our solution. We carried out a performance evaluation of our proposed framework in order to verify its efficacy and determine the need for improvement in future research. We also discussed how our idea differed from existing solutions. Based on the merits of our model, we think that the Blockchain solution is a step toward increasing data security and privacy and has the potential to be used in many Big Data applications.

The framework implemented is only a prototype that exploits the use of Blockchain and IPFS to proof of concept. In future research, we plan to implement a real-world system in a real-world setting and will focus on improving system performance by experimenting with solutions related to the system architecture, Blockchain architecture, and Blockchain type. Additionally, we aim to use high-performance computing (HPC) in sensitive data encryption and extend in using mobile agents in order to increase system efficiency.

Author Contributions: Conceptualization, K.A.A. (Khalil Ahmad Alsulbi), M.A.K., and F.E.E.; methodology, K.A.A. (Khalil Ahmad Alsulbi) and K.A.A. (Khalid Ali Almarhabi); software, K.A.A. (Khalil Ahmad Alsulbi); validation, M.A.K., A.A.B., F.E.E., K.M.J., and K.A.A. (Khalid Ali Almarhabi); writing—original draft preparation, K.A.A. (Khalil Ahmad Alsulbi) and K.A.A. (Khalid Ali Almarhabi); writing—review and editing, K.A.A. (Khalil Ahmad Alsulbi) and M.A.K.; supervision, M.A.K., F.E.E., and K.M.J.; project administration, M.A.K., F.E.E., and K.M.J.; funding acquisition, M.A.K., A.A.B., F.E.E., and K.M.J. All authors have read and agreed to the published version of the manuscript.

Funding: This project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under grant no. (RG-9-611-38). The authors, therefore, acknowledge with thanks DSR for technical and financial support.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to thank the editor and the anonymous reviewers, whose insightful comments and constructive suggestions helped us to significantly improve the quality of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Baig, M.I.; Shuib, L.; Yadegaridehkordi, E. Big data adoption: State of the art and research challenges. *Inf. Process. Manag.* **2019**, *56*, 102095. [\[CrossRef\]](#)
2. Samsudeen, S.N.; Haleem, A. Impacts and challenges of big data: A review. *Int. J. Psychosoc. Rehabil.* **2020**, *7*, 479–487.
3. Bao, R.; Chen, Z.; Obaidat, M.S. Challenges and techniques in Big data security and privacy: A review. *Secur. Priv.* **2018**, *1*, e13. [\[CrossRef\]](#)
4. Yang, P.; Xiong, N.; Ren, J. Data security and privacy protection for cloud storage: A survey. *IEEE Access* **2020**, *8*, 131723–131740. [\[CrossRef\]](#)
5. Alex, W. Global DataSphere to Hit 175 Zettabytes by 2025, IDC Says. *Datanami* **2018**, *17*, 13237–13244.
6. Subbalakshmi, S.; Madhavi, K. Security challenges of Big Data storage in Cloud environment: A Survey. *Int. J. Appl. Eng. Res.* **2018**, *13*, 13237–13244.
7. Venkatraman, S.; Venkatraman, R. Big data security challenges and strategies. *AIMS Math.* **2019**, *4*, 860–879. [\[CrossRef\]](#)
8. Prajapati, P.; Shah, P. A review on secure data deduplication: Cloud storage security issue. *J. King Saud-Univ.-Comput. Inf. Sci.* **2020**, in press. [\[CrossRef\]](#)
9. Sun, P.J. Privacy protection and data security in cloud computing: A survey, challenges, and solutions. *IEEE Access* **2019**, *7*, 147420–147452. [\[CrossRef\]](#)

10. Big Data Working Group, Expanded Top Ten Big Data Security and Privacy Challenges. Available online: https://downloads.cloudsecurityalliance.org/initiatives/bdwg/Expanded_Top_Ten_Big_Data_Security_and_Privacy_Challenges.pdf (accessed on 27 February 2016).
11. Wright, C.S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: <https://ssrn.com/abstract=3440802> (accessed on 21 June 2021).
12. Conoscenti, M.; Vetro, A.; De Martin, J.C. Blockchain for the Internet of Things: A systematic literature review. In Proceedings of the 2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), Agadir, Morocco, 29 November–2 December 2016; pp. 1–6. [\[CrossRef\]](#)
13. Zahid, J.I.; Ferworn, A.; Hussain, F. *Blockchain: A Technical Overview*; IEEE Internet Policy Newsletter; IEEE: Piscataway, NJ, USA, 2018; pp. 1–3.
14. Dai, H.N.; Zheng, Z.; Zhang, Y. Blockchain for Internet of Things: A survey. *IEEE Internet Things J.* **2019**, *6*, 8076–8094. [\[CrossRef\]](#)
15. Aujla, G.S.; Chaudhary, R.; Kumar, N.; Das, A.K.; Rodrigues, J.J. SecSVA: Secure storage, verification, and auditing of big data in the cloud environment. *IEEE Commun. Mag.* **2018**, *56*, 78–85. [\[CrossRef\]](#)
16. Dinh, T.T.A.; Wang, J.; Chen, G.; Liu, R.; Ooi, B.C.; Tan, K.L. Blockbench: A framework for analyzing private blockchains. In Proceedings of the 2017 ACM International Conference on Management of Data, Chicago, IL, USA, 14–19 May 2017; pp. 1085–1100. [\[CrossRef\]](#)
17. Lei, K.; Zhang, Q.; Xu, L.; Qi, Z. Reputation-based byzantine fault-tolerance for consortium blockchain. In Proceedings of the 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS), Singapore, 11–13 December 2018; pp. 604–611. [\[CrossRef\]](#)
18. Shahnaz, A.; Qamar, U.; Khalid, A. Using blockchain for electronic health records. *IEEE Access* **2019**, *7*, 147782–147795. [\[CrossRef\]](#)
19. Mohanta, B.K.; Panda, S.S.; Jena, D. An overview of smart contract and use cases in blockchain technology. In Proceedings of the 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 10–12 July 2018; pp. 1–4. [\[CrossRef\]](#)
20. Bahga, A.; Madiseti, V.K. Blockchain platform for industrial internet of things. *J. Softw. Eng. Appl.* **2016**, *9*, 533–546. [\[CrossRef\]](#)
21. Krupansky, J. What Is a Software Agent. 2015. Available online: <https://medium.com/@jackkrupansky/what-is-a-software-agent-6089dfe8f99> (accessed on 5 July 2021).
22. Mobile Agent Framework. 2013. Available online: <http://maf.sourceforge.net/> (accessed on 5 July 2021).
23. Alsulbi, K.; Khemakhem, M.; Basuhail, A.; Eassa, F.; Jambi, K.M.; Almarhabi, K. Big Data Security and Privacy: A Taxonomy with Some HPC and Blockchain Perspectives. *Int. J. Comput. Sci. Netw. Secur.* **2021**, *21*, 43–55.
24. Centonze, P. Security and privacy frameworks for access control big data systems. *Comput. Mater. Continua* **2019**, *59*, 361–374. [\[CrossRef\]](#)
25. Gupta, S.; Vashisht, S.; Singh, D. Enhancing Big Data Security Using Elliptic Curve Cryptography. In Proceedings of the 2019 International Conference on Automation, Computational and Technology Management (ICACTM), London, UK, 24–26 May 2019; pp. 348–351. [\[CrossRef\]](#)
26. Yau, Y.C.; Khethavath, P.; Figueroa, J.A. Secure Pattern-Based Data Sensitivity Framework for Big Data in Healthcare. In Proceedings of the 2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD), Honolulu, HI, USA, 29–31 May 2019; pp. 65–70. [\[CrossRef\]](#)
27. Al-Odat, Z.A.; Al-Qtiemat, E.M.; Khan, S.U. A big data storage scheme based on distributed storage locations and multiple authorizations. In Proceedings of the 2019 IEEE 5th Intl Conference on Big Data Security on Cloud (Big Data Security), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS), Washington, DC, USA, 27–29 May 2019; pp. 13–18. [\[CrossRef\]](#)
28. Zhang, Y.; Yang, M.; Zheng, D.; Lang, P.; Wu, A.; Chen, C. Efficient and secure big data storage system with leakage resilience in cloud computing. *Soft Comput.* **2018**, *23*, 7763–7772. [\[CrossRef\]](#)
29. Fan, K.; Lou, S.; Su, R.; Li, H.; Yang, Y. Secure and private key management scheme in big data networking. *Peer-Peer Netw. Appl.* **2018**, *11*, 992–999. [\[CrossRef\]](#)
30. Naz, M.; Al-zahrani, F.A.; Khalid, R.; Javaid, N.; Qamar, A.M.; Afzal, M.K.; Shafiq, M. A secure data sharing platform using blockchain and interplanetary file system. *Sustainability* **2019**, *11*, 7054. [\[CrossRef\]](#)
31. Zhang, C.; Li, Y.; Sun, W.; Guan, S. Blockchain Based Big Data Security Protection Scheme. In Proceedings of the 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 12–14 June 2020; pp. 574–578. [\[CrossRef\]](#)
32. Desai, S.; Shelke, R.; Deshmukh, O.; Choudhary, H.; Sambare, S.S. Blockchain based secure data storage and access control system using IPFS. *J. Crit. Rev.* **2020**, *7*, 1254–1260. [\[CrossRef\]](#)
33. Chen, J.; Lv, Z.; Song, H. Design of personnel big data management system based on blockchain. *Future Gener. Comput. Syst.* **2019**, *101*, 1122–1129. [\[CrossRef\]](#)
34. Jung, Y.M.; Jang, W.J. Data management and searching system and method to provide increased security for IoT platform. In Proceedings of the 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, 18–20 October 2017; pp. 873–878. [\[CrossRef\]](#)
35. Javed, M.U.; Rehman, M.; Javaid, N.; Aldegheshem, A.; Alrajeh, N.; Tahir, M. Blockchain-based secure data storage for distributed vehicular networks. *Appl. Sci.* **2020**, *10*, 2011. [\[CrossRef\]](#)

-
36. Truffle Suite. Available online: <https://truffleframework.com/docs/ganache/overview> (accessed on 23 May 2021).
 37. Metamask. Available online: <https://metamask.io/> (accessed on 23 May 2021).
 38. Node.js. Available online: <https://nodejs.org/en/> (accessed on 25 May 2021).
 39. Truffle Framework. Available online: <https://www.trufflesuite.com/truffle> (accessed on 25 May 2021).
 40. Hyperledger Besu. Available online: <https://www.hyperledger.org/use/besu> (accessed on 19 July 2021).
 41. Hyperledger Caliper. Available online: <https://hyperledger.github.io/caliper/> (accessed on 19 July 2021).
 42. Performance and Scale Working Group. Hyperledger Blockchain Performance Metrics. Available online: <https://www.hyperledger.org/learn/publications/blockchain-performance-metrics> (accessed on 19 May 2021).