EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

A Framework for Data Provenance Assurance in Cloud Environment using Ethereum Blockchain

Narayan D G [1], Rashmi B.[2], Pavitra H [3],Yashawardan D [4]

[1, 2, 3, 4] School of Computer Science and Engineering, KLE Technological University, Hubballi, Karnataka, India

Abstract

Cloud data provenance refers to the systematic recording and tracking of the entire lifecycle of digital data within a cloud. Ensuring secure data provenance is crucial for maintaining accountability and confidentiality in cloud environments. However, establishing trust between cloud customers and service providers remains a challenge, highlighting the need for assured data provenance models. In recent times, blockchain technology has emerged as promising solution for designing data provenance assurance models. It provides a decentralized and distributed ledger to record the provenance of digital assets. In this work, we design a blockchain-based framework for ensuring data provenance in cloud storage. Initially, we develop a cloud storage application using OpenStack swift storage which helps in the generation of provenance data. Subsequently, we design a data provenance assurance framework for confidential files of users using the Ethereum blockchain. To evaluate the scalability and performance of the proposed framework, we analyze various performance parameters such as transaction throughput, latency and gas limit using various scenarios. The performance of the system is compared under two consensus algorithms namely proof of work and proof of authority. This analysis helps to assess the effectiveness and efficiency of the blockchain-based solution in ensuring data provenance in cloud storage environments.

Keywords: Data Provenance, Blockchain, Ethereum, POW, POA, OpenStack, Swift API

Received on 06 July 2023, accepted on 30 September 2023, published on 09 October 2023

doi: 10.4108/eetsis.3536

1. Introduction

Cloud computing operates as a model that offers convenient and on-demand access to various resources, including networks, computing power, storage, databases, and services. These resources can be provisioned through cloud service providers, allowing users to leverage features such as on-demand services, availability, and pay-as-you-go pricing. Consequently, cloud computing has attracted millions of users. Given the reliance on cloud services for storing and managing data, it becomes crucial for users to have confidence in the safety and security of their data [1] [2]. Thus, cloud service providers must prioritize robust security measures and implement mechanisms to maintain the confidentiality, i ntegrity, a nd a vailability o f user data. By establishing secure infrastructure, adopting encryption techniques, implementing access controls, and regularly monitoring and auditing systems, cloud service providers can enhance the security and instill trust among their users. One of the foremost concerns in cloud security research revolves around the creation and administration of data provenance within cloud storage.

Data provenance in cloud storage environments has become an important research issue due to the increasing dependence on cloud infrastructure for data storage, sharing, and processing [3] [4]. Data provenance provides a comprehensive record of a data asset's journey, including its creation, modifications, a nd access, thus enabling transparency and accountability. This information is crucial for verifying data authenticity, detecting unauthorized alterations, adhering to regulatory mandates, and conducting efficient audits. Furthermore, data provenance supports error tracing and debugging, promotes collaborative data sharing, enhances resource allocation, and mitigates the risks associated with data breaches and unauthorized access, all of which are essential in today's cloud-centric data ecosystems.

Recently, blockchain has been used to address the data security issues in healthcare and cloud [5] [6] [7]. Furthermore, blockchain is emerging as a promising technology for design of robust data

*Corresponding Author. Email: narayan_dg@kletech.ac.in

provenance mechanism in cloud storage [8] [9]. As cloud storage becomes the backbone of modern data management, ensuring data integrity, security, and transparency is paramount. Traditional centralized systems often fall short in meeting these requirements, leaving data vulnerable to breaches and unauthorized access. Blockchain's decentralized and immutable ledger technology offers a compelling solution by providing an unforgeable, transparent, and auditable record of data transactions and access. This not only enhances data security but also facilitates regulatory compliance, error detection, and efficient auditing. By integrating blockchain into cloud storage data provenance, organizations can establish a trustworthy and verifiable system for tracking data history, instilling confidence in their cloud-based data management processes and addressing the evolving challenges of data security and accountability.

In this study, we introduce a data provenance architecture leveraging blockchain technology to safeguard data activities within a cloud storage application. Our framework records the history of operations as provenance data using OpenStack-based Swift storage, with the corresponding data hashes being stored as blockchain transactions. These blockchain transactions are then grouped into blocks, which must undergo validation by a cluster of nodes before being added to the blockchain. Modifying or tampering with a data log from the provenance would require locating the specific transaction and block. However, it's essential to note that the cryptographic principles underpinning blockchain technology permit alterations to a block record only if an attacker can present a longer chain of blocks compared to the majority of miners' blockchain. Achieving this task is exceedingly challenging, thereby offering a robust level of security against tampering attempts. Through the application of blockchain technology in the real-time cloud storage, our proposed architecture enhances the data provenance and contributes to preserving the integrity and security of recorded data activities.

The contributions of this work are:

- Developed Blockchain as a Service (BaaS) for provisioning of Ethereum blockchain platform on an OpenStack private cloud.
- Developed a drop-box like storage application using OpenStack Swift for logging of users metadata.
- Designed and implemented a data provenance framework for cloud data storage using Ethereum smart contracts.
- Evaluated the performance of blockchain platform on provenance framework considering the impact of network size, load, consensus mechanism, difficulty level, and hardware configurations.

The paper's structure is outlined as follows: In Section 2, we provide an overview of the background work, encompassing related research, OpenStack architecture, Ethereum blockchain, consensus algorithms, and Blockchain as a Service (BaaS). Section 3 delves into the proposed work, encompassing the design of storage application, BaaS, and the proposed data provenance assurance framework using blockchain along with the algorithms. Section 4 examines the impact of the blockchain platform on the data provenance mechanism as part of our result analysis. Lastly, in Section 5, we present the conclusion and outline avenues for future research.

2. Background Study

In this section, we discuss the related work and brief overview of OpenStack, Ethereum and consensus algorithms which are part of the data provenance framework.

2.1. Related work

Within this section, our focus is on review of existing research and literature related to data security. Particularly, we review the research work on the application of blockchain technology for data provenance assurance within cloud storage systems. In the literature, data security is addressed through diverse methodologies, including the application of machine learning [10] and [11]. Additionally, access control to data is discussed in [12], [13], [14], [15] and [16].

Data accountability and privacy are fundamental attributes of cloud computing. However, delivering these features has become challenging for cloud stakeholders due to the underlying infrastructure. To address this challenge, blockchain technology has been used to tackle the issue of cloud data provenance in [17], and [18]. The authors contend that the decentralized and distributed characteristics of blockchain protocols not only meet security prerequisites but also stimulate the advancement of improved security measures for cloud storage. In [19], the authors proposed a data provenance framework for the cloud environment using blockchain technology. They implemented their framework on AWS S3 and demonstrated the use of blockchain using dropbox-like application. The authors collected metadata from user activities occurring in the cloud environment and stored this metadata using IPFS (InterPlanetary File System), which is a decentralized protocol for storing and accessing data. IPFS utilizes a decentralized peer-to-peer approach for data retrieval. In the proposed framework, the metadata was stored

in IPFS, and its cryptographic hash was recorded in the blockchain. Traditional HTML requests for data are typically based on file locations, but in IPFS, data can be requested using the cryptographic hash. The blockchain assigned an ID to each block in the ledger, and the authors used this Block ID to verify the state of the data stored in IPFS.

In [20], authors have applied the characteristics of data provenance to the domain of e-science and have created a taxonomy. It categorized their provenance systems based on why they collect provenance, what they describe, how they represent and store provenance. Their main intention was to follow the scientific approach. In [21], the authors address a critical cloud computing problem that is becoming a key concern for cloud stakeholders. Cloud service providers have found that data in cloud computing environments cannot be tracked effectively due to inadequate monitoring software and also because of the logging mechanisms that were used. The logging mechanisms were system-centric which means when there was a shift from owning computing services to buying computing services, there were many concerns regarding the ownership of the data, transparency of the data, and security of course. Hence the authors tried to make it data-centric instead and introduced S2Logger, which collects, visualized, and analyses data operations in the cloud. Thus, the resulting events of data give rise to the provenance data records throughout the data lifecycle. In [22], the authors present a novel virtual machine (VM) authorization mechanism for cloud data centers. They address the challenges associated with the default VM authorization scenario, where users are provided with SSH keys and IP addresses for accessing their VMs. This approach becomes problematic when users request multiple cloud resources, as it results in the storage of numerous keys and IP addresses. To overcome these issues and enhance VM authorization security, the authors propose a method that combines a claims-based authorization system with decentralized storage based on blockchain technology. By leveraging the benefits of blockchain, the proposed system offers a more secure and reliable approach to VM authorization.

In [23], the authors present a Provenance Logger, a kernel-space logger, and call it as Progger. With this Progger, cloud stakeholders can easily trace the data stored in the cloud. The logs from the kernel-space provide every small action taken, thus giving security analysts more detailed data. This provenance data gives way to the creation of many higher-level tools for successful end-to - end monitoring of data provenance. Progger has thus provided log tamper-evidence, eliminating removed false or manual records, providing exact and granular synchronization of timestamps across multiple devices, and effective monitoring of system core usage. This offers strong data protection reliability, and audits of user operation. In [24], authors introduce BSTProv system for sharing secure and reliable data provenance using blockchain. Authors achieve the provenance by dividing the local provenance graph into multiple sub graphs and storing these as transactions in distributed ledger. The proposed system also incorporates the authorization and cross-domain provenance in smart contracts. The implementation is done using consortium blockchain.

In [25], the authors propose a decentralized system that allows various parties to jointly store, operate and execute data and preserve the data privacy. Here, the authors have built a model Enigma, which is highly upgraded model of stable multi - agent computing, assured by a genuine network of secret sharing. A modified distributed hash table is used for storing secret-shared data using an external blockchain. It provides authorisation and serves as an event log which is tamper-proof. Like Bitcoin, Enigma removes confidence in a third party and ensures autonomous control of personal data. With this, users can share their data for the first time with cryptographic protections for their privacy. In [26], the authors focus on utilizing blockchain techniques to enable data tracing throughout the entire data life cycle, with a particular emphasis on tracing data transfers within cloud data centers. In [27], the authors propose a data provenance mechanism that employs a lightweight mining algorithm. This mechanism establishes a legal and ethical framework for important categories of provenance. However, it does not specifically address data stored in cloud storage.

In [28], the authors present a hybrid algorithm based on blockchain to enhance the privacy and efficiency of existing techniques. They evaluate their approach using a virtual cloud environment. In [29], the authors introduce a system called DistProv, which combines IPFS and blockchain technology. The system secures digital documents using permission-based access control through Ethereum smart contracts, and zero-knowledge proofs. Authors in [30] design a data provenance scheme for multi-cloud storage. The provenance data is stored and validated using using smart contracts. Authors evaluate the security and efficiency through simulation experiments. Authors in [31] propose architecture for ensuring secure storage in a cloud environment using smart contract. Authors use blockchain and the IPFS for to store and validate the storage data.

2.2. OpenStack Architecture

OpenStack [32] is a widely used open source cloud operating system designed for the deployment of public and private clouds. OpenStack offers a set of APIs that enable users to access and manage the

various components of a cloud infrastructure. Figure 1 illustrates the architecture of OpenStack with different layers and components involved in its operation.

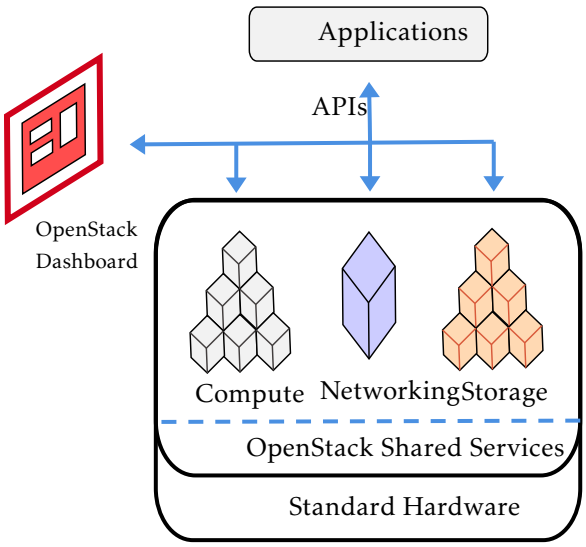

Figure 1. OpenStack Architecture

OpenStack consists of three services namely compute, network, and Storage service. It has additional services such as Heat, Glance, Keystone, Horizon, etc. Three important functions are described below.

Compute Service:

The OpenStack Compute service is a critical component of IaaS. It helps in hosting and managing virtual instances for users. NOVA handles the life cycle management of virtual machines (VMs) and supports various hypervisors to create and run these VMs.

Networking Service:

OpenStack networking service called Neutron, plays a crucial role in managing networks within OpenStack. It enables the creation of network topologies that are independent of the underlying hardware from different vendors. Neutron takes care of various network-related tasks such as handling IP addresses, DNS, load balancing, security groups, DHCP (Dynamic Host Configuration Protocol), and firewall policies. Users can create and manage network configurations for guest virtual machines (VMs) in their cloud projects. It provides the necessary tools and APIs to define and control network resources, allowing users to customize their network setups based on their specific requirements within the OpenStack environment.

Storage Service:

Storage as a Service (STaaS) refers to the management of storage resources through a set of remote APIs, allowing users to access and utilize storage capabilities without needing to deal with the underlying implementation details. This approach abstracts the complexities of storage infrastructure and provides a standardized interface for storing, backing up, and sharing data. OpenStack supports three categories of storage's namely object, block and shared file.

2.3. Ethereum

Ethereum Blockchain [33] provides secure peer-to-peer applications. Ethereum is a decentralized, open-source framework to create private and public blockchain network. It is used to create crypto currency as well as the distributed applications. Ethereum platform provides incentives in terms of Ethers to the miners. Ethereum uses smart contracts for creating transactions. Ethereum architecture can be viewed as a layered architecture for developing distributed applications as shown in Figure 2.

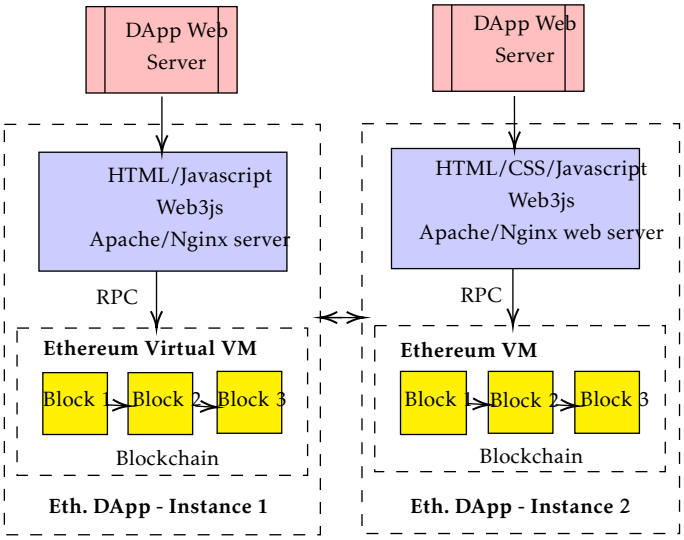

Figure 2. Ethereum Framework Architecture

Ethereum Dapp:

In Ethereum framework architecture, each web browser or client interacts with its instance of the application. Hence, we call this as the decentralized application as there is no central server to monitor and control. Ethereum consists of two main components. First is the database, where every transaction in the network is stored in the blockchain. Anyone can verify this transaction as there is complete transparency. Second is the code which contains the main logic of the application. The web server interacts with the EVM using RPC as shown in Figure 2.

Geth:

Ethereum Foundation has created Geth client application which is written in the Go programming

language. A copy of the blockchain is downloaded here. It continuously interacts with other nodes to keep its copy of the blockchain up to date. It can also mine blocks and add transactions to the ledger, verify the block transactions and also carry out the transactions.It also functions as a server via the provision of APIs with which you can communicate through RPC. This also comes with a javascript application (geth console), which can be used for connecting to the blockchain.

RPC:

Remote Procedure Calls (RPC) for Ethereum allow developers to interact with nodes from remote applications. They enable the submission of requests to node and return relevant information on the Ethereum blockchain.

Web3 Js:

Web3.js helps us to design clients who connect with the Ethereum Blockchain. It's a library allows operations such as transferring ethers and writing smart contracts.

2.4. Consensus Algorithms

We use two consensus algorithms to evaluate the scalablity of proposed data provenance framework. The distributed network such as blockchain, unlike conventional architecture, has nodes that act as both a server and a host, and it needs to share information with other nodes to achieve consensus. Often some nodes are down or offline, and some malicious nodes also occur, which can seriously affect or break the consensus process. Hence, a secure and efficient consensus protocol can accommodate the presence of these factors and reduce the damage.

We use the following consensus algorithm as part of our evaluation.

Proof-of-Work:

PoW consensus is most widely used algorithm in blockchain platforms. It is used in both Bitcoin and Ethereum, the two best known cryptocurrencies. Blockchain transactions need verification, and 'miners' do that. The transaction verifier needs to know the cryptographic hash value of the last documented block and that's secret from everybody. PoW 's advantage is that finding the hash requires immense computational power, and the mathematical puzzle is overhauled every 14 days making it complex. PoW is very secure as performing a DDoS attack requires 51% percent of the total computing power. But this occurs at a very high cost, electrical energy and continuous upgrade of hardware.

Proof-of-Authority:

Proof of Authority (PoA) algorithm is based on reputation which provides effective solution for blockchain networks particularly in private networks. The advantages are it allows having valid and trustworthy identities. It has a standard for approving the validator which makes it difficult to become a validator. However, the validators are visible to everyone and the third-party can cause manipulation using the identities of the validators they know. It sacrifices decentralization to achieve high throughput and scalability.

2.5. Blockchain as a Service in Cloud Environment

Developers often encounter difficulties in finding a convenient and effective approach to implement, manage, and supervise their applications on cloud platforms, which hampers the reliability and security of their applications. The complexity of the infrastructure itself is a key factor contributing to these challenges. During the development of business code, developers are often unaware of the potential consequences associated with the intricate nature of the operating platform, making it challenging for them to proactively address future inaccuracies. Towards this, Blockchain-as-a-Service (BaaS) has emerged as important service. Companies such as IBM and Microsoft have introduced their own BaaS platforms, yielding positive outcomes by mitigating deployment and development challenges.

With integration of blockchain platform leveraging the cloud services, the developers can benefit from streamlined deployment and maintenance. This enables the delivery of user-friendly and high-performance blockchain ecosystems and related services to developers while mitigating concerns associated with the underlying architecture. In our work, we offer a BaaS platform on an OpenStack-based cloud, providing developers with a comprehensive ecosystem.

3. Proposed Work

Within this section, we initially discuss the design of BaaS platform and storage application in OpenStack cloud. Later, we discuss the design of data provenance assurance mechanism using Etherum blockchain.

3.1. Blockchain as a Service on OpenStack Private Cloud

Blockchain is a platform that allows users to host and use their apps, functions, and smart contracts on a blockchain. It behaves like a full-fledged platform that eases the development process. A blockchain network is built by adding nodes to it. These nodes are called peers in the network. Before adding the peer to the network, the initial step is to install all the services and have the genesis file initialized. This is done by a simple user interface using the Django framework. Then the enode of the geth client needs to be shared

across all the peers in the network. All the peers verify the consensus rules and then add it to the network. Now, this peer gets the updated copy of the blockchain. Any peer can carry out transactions with any other peer in the network, for which they get rewarded with ethers. All the transactions are carried out securely with complete transparency and data integrity.

Algorithm 1: Blockchain as a Service

```
Input: Blockchain Network name C, number of
peers N, configuration of peers, S
Output: File F containing IP address and
keypair K
Begin
    for N nodes do
    Create a VM using Openstack SDK
    Create a keypair K
    end for
    for N nodes do
    Create paramilko session
    Change the hostname in host file
    Install services T:ethereum,geth,solidity
    if blockchain services T == True then
        Initialize the genesis file
        Open the geth console
        Open new session of VM and
        attach to geth console
        Get enode-id of geth instance
        and write to F
        Replace loopback address with private IP
        of the VM in enodeid in file F
    end if
    end for
    Mail file F with IP address and keypair K
    for N nodes do
        Add the enode of all (N-1) peers
        to the Nth peer
    end for
End
```

Algorithm 1 describes how Blockchain-as-a-Service is implemented using OpenStack VMs. The required number of VMs requested by a user are created using OpenStack SDK and keypair is generated. Then, to create private blockchain network, paramiko sessions are created in all the VMs. Required libraries and packages are installed. Later, from the geth console, unique enode-id of respectives nodes are fetched in a file and this file is exchanged among other nodes to form peers in the private blockchain network.

3.2. Storage Application using OpenStack SWIFT

In this subsection, we discuss the design of storage of application using OpenStack SWIFT storage. This application is designed for the private cloud deployed in the campus. Later, we discuss the use this application to create provenance data for the user's files.

There are many Internet-based cloud services that are offered by industry giants like Google and Amazon, which are the most efficient and secure in their field. But, they lack certain factors. Public cloud services require high bandwidth to upload or download files from their servers. No matter where we store our data on the internet, there is always a risk of losing its confidentiality. Since uploading and downloading of files would happen within the college campus network in the private cloud, no internet access would be required for the purpose, hence making the procedure faster and more flexible. Thus we designed a drop-box like application for private cloud with features like upload, download, share, and rename. We also implemented additional functionalities like synchronization of files.

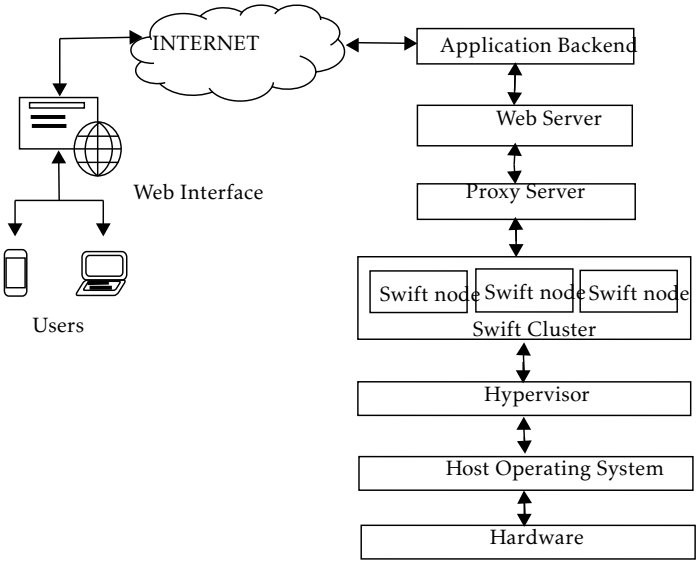

Figure 3. Storage Application using OpenStack SWIFT

Figure 3 illustrates the storage application built using OpenStack Swift at the backend which is similar to Dropbox. The OpenStack cloud architecture uses four primary nodes namely Compute, Neutron, Controller, and Storage for computing, networking, managing networking resources, and storage respectively. These nodes are deployed on top of a hypervisor. Users are connected through a web interface to the OpenStack application backend. The application has four main modules as represented in the figure 3. The user files module has the user container where all the objects of the user are stored, which can be accessed and various operations like download, upload, share, etc can be performed over it only by the owner. End users can share objects with other users via email or directly to their account through the sharing module. One can

make a folder public, which makes the objects in it accessible to all other users that can search for their profile in the showcase module. The desktop client module syncs a selected folder between the cloud web client and the local machine desktop client. All the modules are accessible only after authentication through a service called the keystone.

Algorithm 2 shows how metadata is generated from OpenStack Dropbox like storage application. All the authenticated cloud users can make file operations on those files for which they are authorized to make. Upon any action by any user, the provenance data is written to a file consisting of Record_ID, Date_Time, User_Name, File_Name, Action performed.

Algorithm 2: Generating provenance data from OpenStack Storage application

```
Input: File operations done by users on
OpenStack Swift application, file F
Output: Provenance data
Begin
    FILE *fp
    for each user do
        fp ← fopen(provenance_data,"a")
        if fp == NULL then
            print "ERROR"
        end if
        for each file operations of user do
            Record_ID ← record_id
            Date_Time ← timestamp
            User_Name ← username
            File_Name ← filename
            Action ← action_performed
        write data into the file F
        end for
        fclose(fp)
    end for
End
```

3.3. Cloud Storage Provenance Assurance using Blockchain

We use the Swift storage application that is designed as shown in Figure 3. Figure 4, describes the working of the proposed system. User can request the Cloud Service Provider to access the data that he has stored in Swift storage application (1). Data changes caused by these operations are monitored and validated by using the proposed framework. These operations are recorded as provenance data. The Cloud user who stores his data in the Swift storage, can perform various file operations on those files for which he is authorized (2). The Cloud Service Provider who provides a cloud-based platform, application, or storage services stores this provenance data in blockchain (3b) as well as in a local database (3a). Blockchain is a distributed network that stores the provenance data records in the form of blocks. The database records all the provenance data having attributes like date, time, username, filename, affected user, an action performed by the cloud user, and the corresponding transaction hash and block hash. Through the blockchain-based provenance system, cloud providers can detect the data breach and take necessary action. To check the integrity of the data stored in the cloud, the user requests the auditor to validate the data (4). In turn, the auditor asks the blockchain network to receive a blockchain receipt for a given data (5). The blockchain returns that transaction's blockchain receipt (6). It provides parameters like blockHash, blockNumber, from-address, gas consumed, input hash, transaction hash, to-address. During validation, the auditor matches the input hash data record in the provenance data file with the input hash present in the blockchain receipt, by fetching the receipt. The auditor then updates the local database with the validation status (7).

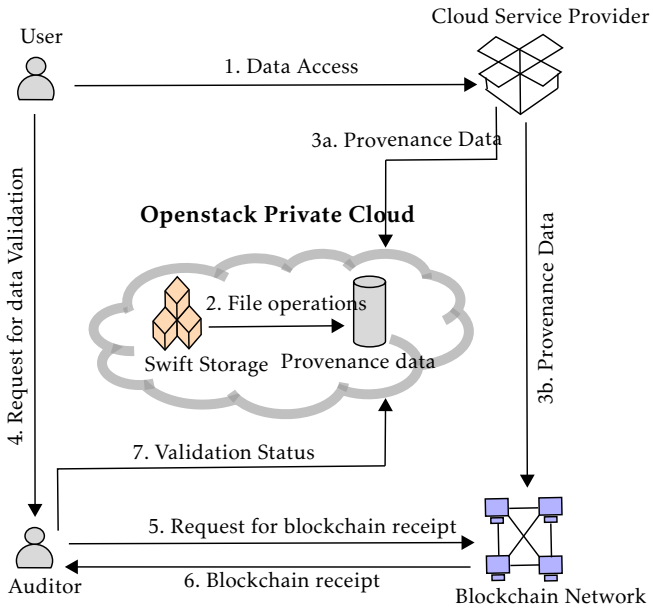

Figure 4. A Framework for Data Provenance Assurance

The proposed framework is built on top of a storage application that is designed using OpenStack SWIFT, which provides user control of personal data and also file sharing. Users can perform various operations like download, upload, share, etc. These operations are monitored and recorded by the service provider and stored as provenance data in a local database including various attributes like Date, time, username, filename, affected user who is affected by the particular operation.

Table 1, shows sample provenance data. Note that tx-hash, block hash, and validation columns are added when the auditor verifies the data record.

The data provenance architecture is built on a blockchain. Blockchain provides the ability to perform operations for cloud storage. Figure 5 describes the higher-level architectural design consisting of three different layers functioning at different levels. The initial step involved here is the collection of provenance data from the private cloud. The second step involves making transactions, where the data collected is encrypted and stored in the blockchain as a hash in a block. The third step is updating the database with file-related information like the file name, time, hash, the operation performed, and location of files.

3.4. Algorithms

We have designed and implemented two algorithms as part of the proposed work. Algorithm to store the provenance data on the blockchain is illustrated in Algorithm 3. Once the provenance data is collected, the data is published to the blockchain network through the smart contract, which has the logic to store the data in the blockchain. Every transaction that is made returns the transaction hash which uniquely identifies input hash which is the hash of the data sent along with the transaction.

Algorithm 3: Store Provenance Data

```
Input: Provenance data file
Output: Transaction hash and Input hash
Begin
while NOT EOF do
    Read record from provenance file
    Start geth console in the node
    if Eth_Account is locked then
        Unlock Eth_Account
    end if
    Create a Smart contract for the transaction
    Start Ethereum Mining and Consensus
    if Mining_and_Consensus == True then
        Write the Transaction as smart contract
        Wait for block creation and validation
        Record Tx and input hash from Ethereum
        Append TX and input hash to Metafile
    end if
    end while
End
```

The algorithm 4 illustrates the process of validating data records. An auditor requests the user for blockchain receipt for validation of the data records stored in the blockchain. The receipt for blockchain includes data record, transaction hash and input hash. The auditor compares the Blockchain hash with provenance file input hash to verify the data record. If true is returned, then the transaction is certified as authentic. If it returns false, it means the block has been tampered with. By adding the validation status the auditor updates the data record in the database.

Algorithm 4: Validation of Data record by the Provenance auditor

```
Input: Transaction Tx, Provenance hash i
Output: True / False
Begin
    Start Ethereum geth client console
    Start Ethereum Mining and Consensus
    if Mining_and_Consensus == True then
        Ethereum_Hash ← Transaction hash, Tx
    end if
    ProvenannceDB_Hash ←
        ProvenannceDB[hashi]
    if Ethereum_Hash == ProvenannceDB_Hash
    then
        return True
    else
        return False
    end if
End
```

4. Results and Discussion

In this section, we discuss the experimental setup and performance analysis of blockchain networks for storing the provenance data. We also carryout the t-test analysis of consensus algorithms to assess the suitability of consensus algorithm for proposed cloud data provenance assurance framework.

4.1. Experimental Setup

We provisioned Blockchain as a service as PaaS in the OpenStack private cloud. Using this, we evaluate the performance of Ethereum Blockchain, by varying the transactions. Transactions refers to the metadata or provenance data received from Swift-based storage application. The scalability and efficiency of blockchain platform affects the design of data provenance mechanism. Thus, we evaluate the performance of the proposed system using six different scenarios in terms of transaction and query delay. We use small, medium and large size VM's for creating blockchain networks as part of BaaS. For the fifth scenario i.e. impact of hardware, we have used medium and large VMs of RAM sizes 4 GB and 8GB GB respectively with 2VCPUs and 20GB HDD. We read the provenance data from the file and store it in Ethereum

Table 1. Data Provenance File

Record ID	Date and Time	User Name	File Name	Affected User	Action	Tx. Hash	Input Hash	Validation
1	07-02-2020 10:00:00	A	X	None	Create a file	k-bits	m-bits	TRUE
2	07-02-2020 20:00:00	A	Y	None	Create a file	k-bits	m-bits	TRUE
3	08-02-2020 09:00:00	A	X	None	Read a file	k-bits	m-bits	TRUE
4	08-02-2020 22:00:00	A	X	B	Share file to user B	k-bits	m-bits	TRUE
5	09-02-2020 10:00:00	A	Y	B	Write to file Y	k-bits	m-bits	TRUE
6	09-02-2020 16:00:00	B	Z	A	Copy file from user B	k-bits	m-bits	TRUE
7	11-02-2020 10:00:00	B	Z	None	Read a file	k-bits	m-bits	TRUE
8	12-02-2020 19:00:00	B	Z	A	Write to file Z	k-bits	m-bits	TRUE
9	12-02-2020 08:00:00	A	U	None	Create a file	k-bits	m-bits	TRUE
10	13-02-2020 17:00:00	A	S	None	Create a file	k-bits	m-bits	TRUE

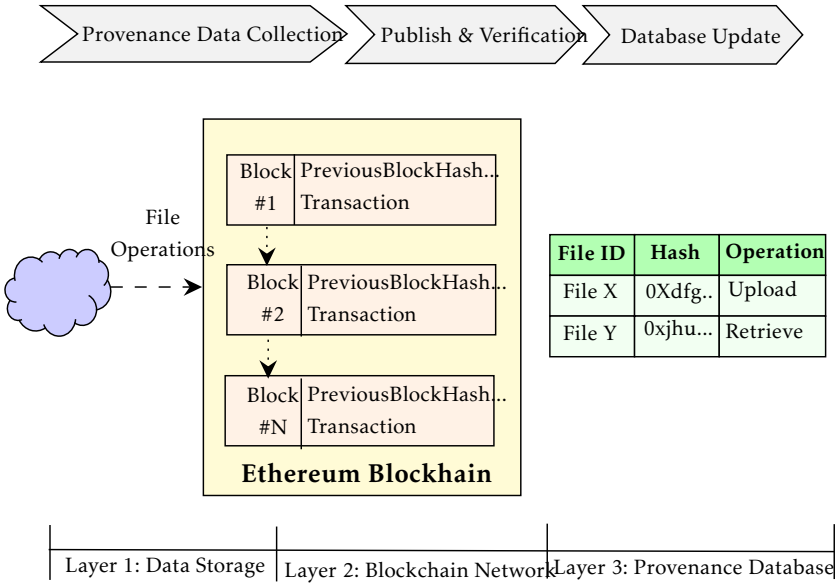

Figure 5. Layered Architecture

blockchain network. We compute transaction and read time for executed records. The system configuration and their versions are presented in Table 2. For all the scenarios, we use multi-node setup created using Blockchain as a Service. BaaS helps in provisioning of different blockchain networks based on the scenarios and the parameters.

Table 2: Software and Versions

software	**Version**
Operating System	Ubuntu v.16.04
Ethereum Client	Geth v.1.7.2
Smart Contract	Solidity v.0.4.17
Application API	Web3.js v.1.0

4.2. Performance Analysis

In this subsection, we discuss the scalability and performance evaluation of blockchain networks for data provenance assurance using different scenarios.

Scenario 1: Impact of Difficulty Level. Mining and consensus in Ethereum is performed using computational puzzle using hashing function. The computational puzzle uses previous hash, nonce and transaction root to compute block hash. The miner who solves the puzzle first create a block and appends it to the ledger. The puzzle has an important component in mining and consensus process which is difficulty level. Difficulty level represents the mining difficulty in terms of trailing zeros in computed block hash. Mining becomes complicated and difficulty by increasing the difficulty level. Thus, performance analysis of proposed provenance mechanism in varying difficulty level becomes an important research issue.

Figure 6 illustrated the latency obtained by varying difficulty levels. In this experiment, 100 transactions are kept constant and difficulty level is varied. We have computed the read and transaction latency for different difficulty levels. We found that, as the difficulty level increased, the both read latency and transaction latency increased, because miners will require more computational power and time to solve the mathematical puzzle and compute the hash. Hence, it creates longer delays in creation of blocks.

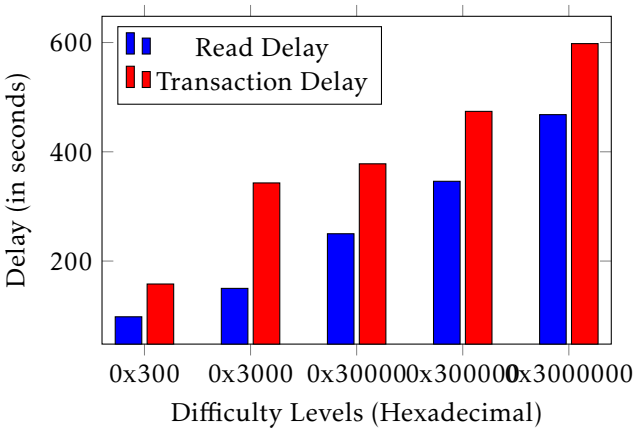

Figure 6. Latency V/s Varying Difficulty Level

Scenario 2: Impact of Load. A Blockchain is predominantly used for making transactions. In our work, the provenance entries are the transactions which need to be written or read from blockchain network. The number of transactions grow over a period of time for each user. Hence, evaluation of transaction write and read latency becomes an important aspect. Furthermore, block difficulty increases continuously in case of POW. Hence, the average computational power consumed when mining a block increases, leading to higher electricity consumption. Gradually, the systems become slower and expensive with the increase in transactions or load.

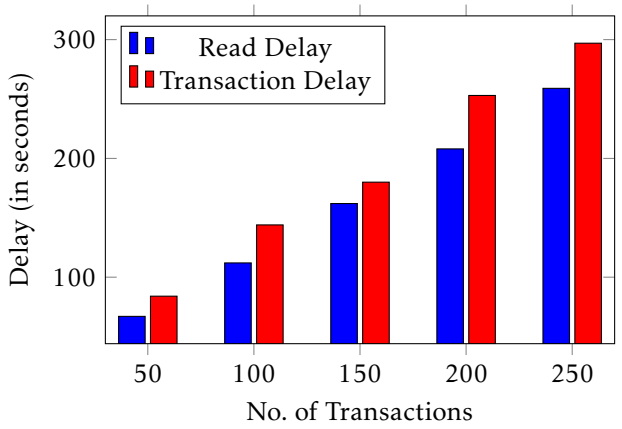

Figure 7. Latency V/s. Varying Load

Figure 7 shows effect of read latency and transaction latency with varying load. We have made a multi-node setup consisting of three peers and used POW algorithm on all nodes with a constant difficulty level of 0x300. We vary the load in terms of 50, 100, 150, 200 and 250 transactions. The results reveal that transaction and read latency increase linearly with varying load. The transaction latency is more compared to read latency. This is because; the write transaction involves mining and consensus mechanism. However, read latency does not involve mining process. The node whichever is nearer to the user, retrieves the required data from blockchain network.

Scenario 3: Impact of Network Size. The important feature of Blockchain is distributed and decentralized network, which makes it different from the traditional centralized system and thus increases reliability, durability and fault tolerance. The cryptographic currency used in a blockchain enables distributed internet hubs to merge. In Ethereum Blockchain, Ethers is the currency used. A well-designed network manages the distribution of tokens across all peers in the network to maximize the growth of the network. The peers in the network directly exchange ethers and therefore control funds. The ether increases in value as the success and utility of the project increases. That is why scaling out is becoming an essential factor in blockchain systems.

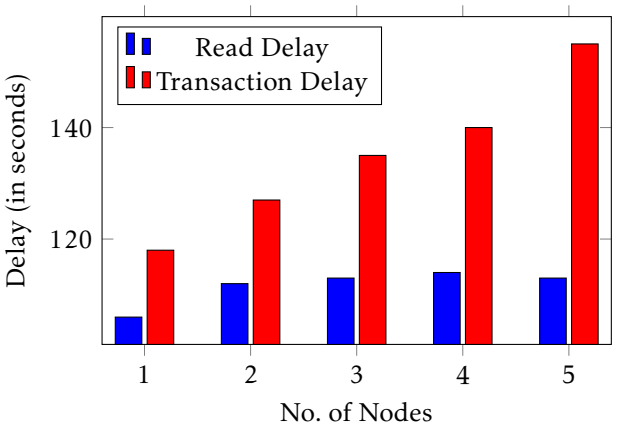

Figure 8. Latency V/s Varying nodes

Figure 8 shows a graph of increase in latency by varying number of nodes in the network. Number of Transactions are 100 with a constant difficulty level and POW consensus algorithm. Network size is increased each time by adding one, two, three, four and five peers respectively. Each node is of 2 GB RAM size. As we scale out our network, transactions and blocks take more time for broadcasting and validating. Thus, write and read latency increases with the increase in number of nodes. Furthermore, transaction latency is higher than read latency as it involves solving computational puzzle as part of mining process.

Scenario 4: Impact of Consensus Algorithm. The consensus algorithm is an important component of any blockchain system. This algorithm affects transaction throughput and latency. Thus, there is a need for choosing a suitable consensus protocol for the proposed provenance framework. Hence, we evaluate the performance of data provenance mechanism using two popular consensus mechanisms supported by Ethereum platform.

Ethereum platform used POW-based consensus algorithm. Though it avoids DDoS attacks and makes miners' work complicated, it only happens at high cost, electrical energy and continuous upgrade of hardware. Thus, latest versions of Ethereum support Proof of Authority (POA) and Proof of Stake (POS) algorithms. Proof of Authority is a reputation-based consensus mechanism which offers high security, privacy, throughput, and performance. This consensus mechanism uses set of authorised validators based on the stake using ether's. Thus, this mechanism helps in designing high throughput systems. In this scenario we have implemented and evaluated POW and the Proof of Authority (PoA) consensus algorithms.

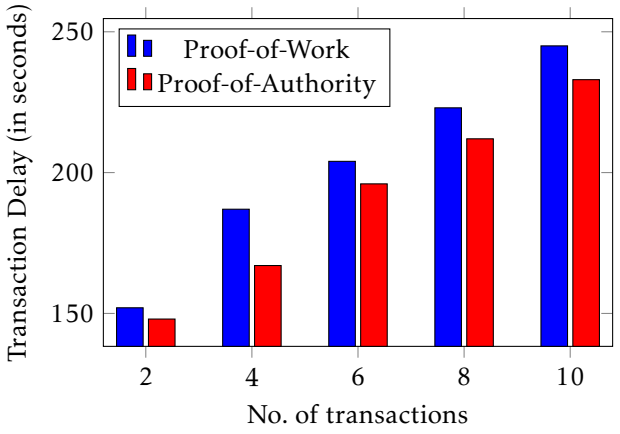

Figure 9. Transaction Delay V/s Varying Load

Figure 9 shows the transaction and ready latency with varying load using two consensus algorithms. The experiment is conducted for 2, 4, 6, 8 and 10 peers in the ethereum private network. A difficulty of 0x300 is kept constant and 200 transactions are carried out. Initially, Proof of Work is implemented and transaction latency is calculated. And the same experiment is conducted by changing only the consensus algorithm to Proof of Authority. We observe that Proof of Authority has a high hash rate, hence requires less computational power when compared to Proof of Work. POA has fixed validators for validating transactions which increases the throughput and security of the network. Whereas POW is resource intensive and difficulty level goes on increasing requiring high power, energy, time, and cost. Therefore we observe lesser transaction latency for POA.

Figure 10 shows a graph of the number of nodes v/s read latency for 2, 4, 6, 8 and 10 nodes with constant difficulty level of 0x300. This experimentation is carried out POW and POA algorithms. We use use 200 transactions to compute the latency. The results reveal that, POW consensus algorithm takes more time to read

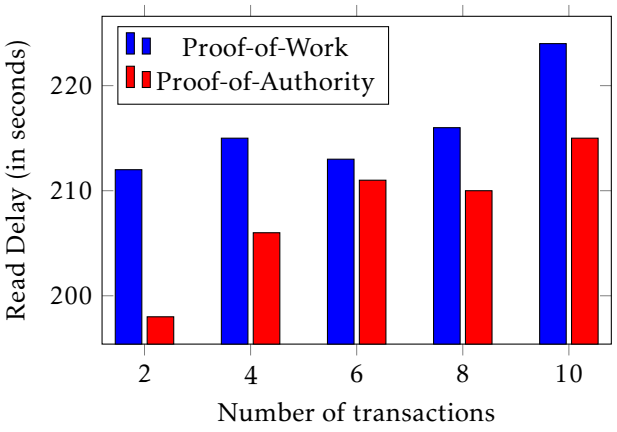

Figure 10. Read Delay V/s Varying Load

from blockchain than the proof of authority. Thus, POA-based consensus algorithms help in designing a scalable and efficient data provenance mechanism using POA.

Scenario 5: Impact of Hardware. In this scenarios, we evaluate the impact of hardware configuration on the transaction and query latency of blockchain networks. In this experiment, we have carried out 100 transactions by keeping difficulty level of 0x300 and proof of work consensus algorithm as constant. We have compared the transaction and query latency considering virtual machines of 4GB and 8GB nodes to check the performance of the blockchain concerning scalability and network infrastructure. As expected, as the RAM size increases, the speed of transactions increases. Also, we observe a linear pattern for transaction and query latency as number of nodes increases which is illustrated in Figure 11 and 12.

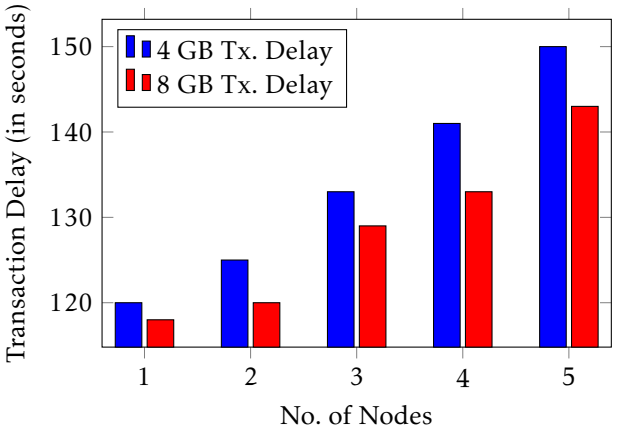

Figure 11. Transaction Delay V/s Varying Nodes

Scenario 6: Impact of Gas limit. Every transaction that is carried out consumes some amount of gas. Gas limit is nothing but the amount of units of gas we are spending on a transaction. And this gas limit will be initialised

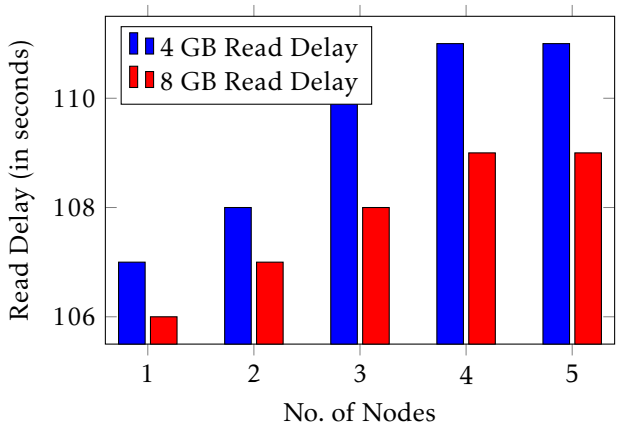

Figure 12. Read Delay V/s Varying Nodes

in the genesis file. Lowering the gas limit wont help us, instead our transaction will get failed. Hence its advisable to add enough gas limit. The unused gas to will be refunded to the account after the transaction becomes successful. We have experimented with Gas Limit (GL) values 0x33000000 (GL1), 0x330000000 (GL2), 0x3300000000 (GL3), 0x33000000000 (GL4) and 0x330000000000 (GL5). From Figure 13, we observe that as we increase the gas limit, we can allow more number of transaction to execute within Ethereum blockchain. Hence, latency decreases.

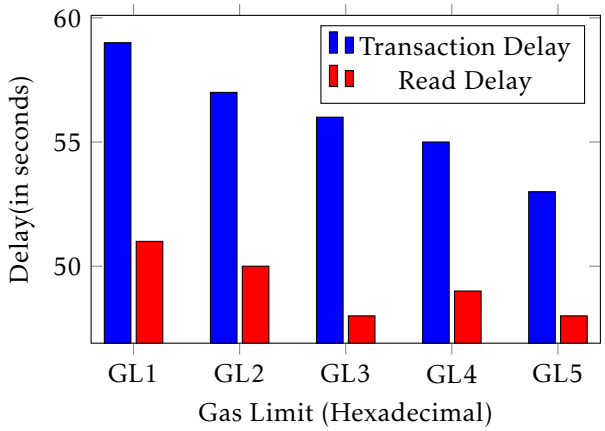

Figure 13. Delay V/s Gas Limit

4.3. t-test Analysis

In the following section, we have used the statistical inference technique t-test to prove that the Proof-of-Authority outperforms Proof-of-Work consensus Algorithm. The confidence level of 0.05 is used. It introduces a null hypothesis (referred to as H0) and an alternative hypothesis (referred to as H1). A test is made with an assumption that the null hypothesis is true. Based on the result we can say if we need to accept the null hypothesis or reject it. Table 3 displays the transaction latency of both the consensus algorithms where difficulty level is kept constant and number of transactions are varied from 25, 50, 75, 100 uptil 250. Sample consists of 10 experimental results. Table 4 describes the statistical test summary. The t-Test Paired Two Sample for Means tool performs a paired two-sample t-Test to check the null hypothesis. The result of this tool is a calculated t-value. P(T <=t) two tail is the probability that a value of the t-Statistic would be observed that is larger in absolute value than t.

Table 3: Experimental Results

PoW(seconds)	PoA(seconds)
80	37
92	67
120	73
140	92
162	108
177	133
199	167
222	189
249	201
301	259

Table 4: t-test Summary

Statistical Analysis Terms	Results
$H_0(NullHypothesis)$	$\mu_d = 0$
$H_1(AlternateHypothesis)$	$\mu_d > 0$
Observations(N)	10
Mean	POW=174.2, POA=132.6
Variance	POW=4940.8 POA=4918.7
df	9
Alpha α	0.05
t Stat	14.76535396
P(T < = t) two-tail	1.29356E-07
t Critical two-tail, μ_L	2.262157163

Based on the given information, the hypothesis testing conducted suggests the following conclusions: The null hypothesis (H0) assumes that the true mean difference (d) is equal to zero. The alternative hypothesis (H1) assumes that d is greater than zero. The mean of PoW is 174.2, and the mean of PoA is 132.6. The degrees of freedom (df) is 9. The t-statistic obtained from the actual t-test is 14.76535396. To determine the statistical significance, we compare the t-statistic to the critical t-value. In this case, the critical t-value is 2.262157163. The probability (p-value) that the absolute value of the t-statistic would be observed to be larger than the critical t-value is calculated as P(T <= t) two-tail (1.29356E-07). Since the p-value is less than the significance level (alpha) of 0.05, we reject the

null hypothesis. This means that there is a significant difference in the means of the two samples. Therefore, we can conclude that there is strong evidence to support the alternative hypothesis (H1) that there is a major difference in the means of the two samples.

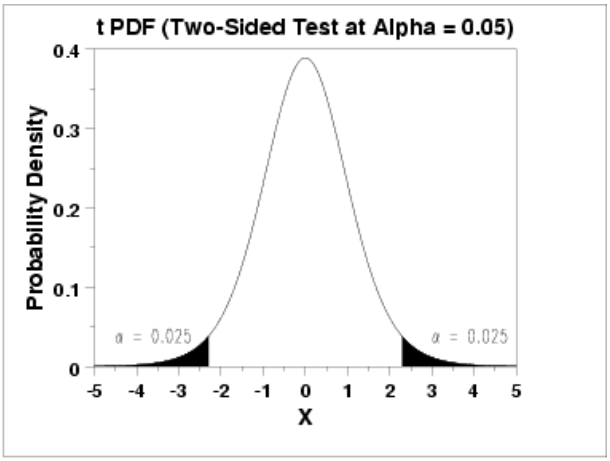

Figure 14. Two-tailed t test curve

From Figure 14, we can see that t Stat (t) value equal to 14.765 lies in rejection region. We can conclude that using PoW, the latency will be more when compared to PoA. PoA outperforms PoW because PoW involves all the miners in the mining process also requires high computational power and energy, thus making it complicated. Whereas PoA on the other-hand has limited validators for sealing the block, reducing the time and providing high throughput. Thus, we conclude that POA is suitable for providing assurance of cloud data provenance.

5. Conclusion

To ensure the integrity and trustworthiness of data stored in the cloud, it becomes important to trace the origin and history of each data object. To tackle this challenge, we introduce a blockchain-based cloud data provenance system designed to provide assurance for data operations within a cloud storage application by recording every action taken on data objects. Our approach involves the development of a cloud storage application utilizing OpenStack's Swift storage and generating comprehensive provenance data. This system acts as a robust defense against malicious activities in the cloud environment improving the reliability of data operations. Using open-source Ethereum blockchain, we have constructed a framework for data provenance, wherein all sensitive files find secure storage. We conducted a performance analysis focusing on scalability, revealing that Proof of Authority (POA) performs better than Proof of Work (POW) in terms of efficiency. It is evident that blockchain performance is influenced by both network size and hardware specifications.

Consensus algorithms play a significant role in scalability and performance of blockchain networks and distributed applications. As future work, we plan to design consensus algorithms suitable for cloud storage provenance assurance using game theory.

References

[1] Butt, Umer Ahmed, Rashid Amin, Muhammad Mehmood, Hamza Aldabbas, Mafawez T. Alharbi, and Nasser Albaqami. "Cloud security threats and solutions: A survey." Wireless Personal Communications 128, no. 1 (2023): 387-413.

[2] Jangjou, Mehrdad, and Mohammad Karim Sohrabi. "A comprehensive survey on security challenges in different network layers in cloud computing." Archives of Computational Methods in Engineering 29, no. 6 (2022): 3587-3608.

[3] Bofeng Pan, Natalia Stakhanova, and Suprio Ray. 2023. "Data Provenance in Security and Privacy.", ACM Comput. Surv. 55, 14s, Article 323 (December 2023), 35 pages. https://doi.org/10.1145/3593294

[4] Oludare Isaac Abiodun, Moatsum Alawida, Abiodun Esther Omolara, Abdulatif Alabdulatif, "Data provenance for cloud forensic investigations, security, challenges, solutions and future perspectives: A survey", Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 10, Part B, 2022, pp. 10217-10245, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2022.10.018.

[5] Gupta, Manish, and Rajendra Kumar Dwivedi. "Blockchain-Based Secure and Efficient Scheme for Medical Data." EAI Endorsed Transactions on Scalable Information Systems 10, no. 5 (2023).

[6] Tripathi, Abhinandan, and Jay Prakash. "Blockchain Enabled Interpolation Based Reversible Data Hiding Mechanism for Protecting Records." EAI Endorsed Transactions on Scalable Information Systems 10, no. 5 (2023).

[7] Gong, Jianhu, and Nima Jafari Navimipour. "An in-depth and systematic literature review on the blockchain-based approaches for cloud computing." Cluster Computing 25, no. 1 (2022): 383-400.

[8] Tosh, Deepak, Sachin Shetty, Xueping Liang, Charles Kamhoua, and Laurent L. Njilla. "Data provenance in the cloud: A blockchain-based approach." IEEE consumer electronics magazine 8, no. 4 (2019): 38-44.

[9] Abhishek, P., Y. Akash, and D. G. Narayan. "A Scalable Data Provenance Mechanism for Cloud Environment using Ethereum Blockchain." In 2021 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), pp. 1-6. IEEE, 2021.

[10] Uppalapati, Padma Jyothi, Bhogesh Karthik Gontla, Priyanka Gundu, S. Mahaboob Hussain, and Kandula Narasimharo. "A Machine Learning Approach to Identifying Phishing Websites: A Comparative Study of Classification Models and Ensemble Learning Techniques." EAI Endorsed Transactions on Scalable Information Systems 10, no. 5 (2023).

[11] Yin, Jiao, MingJian Tang, Jinli Cao, Mingshan You, Hua Wang, and Mamoun Alazab. "Knowledge-Driven Cybersecurity Intelligence: Software Vulnerability Coexploitation Behavior Discovery." IEEE transactions on industrial informatics 19, no. 4 (2022): 5593-5601.

[12] Ezhil Arasi, V., K. Indra Gandhi, and K. Kulothungan. "Auditable attribute-based data access control using blockchain in cloud storage." The Journal of Super computing 78, no. 8 (2022): 10772-10798.

[13] You, Mingshan, Jiao Yin, Hua Wang, Jinli Cao, Kate Wang, Yuan Miao, and Elisa Bertino. "A knowledge graph empowered online learning framework for access control decision-making." World Wide Web 26, no. 2 (2023): 827-848.

[14] Ge, Yong-Feng, Maria Orlowska, Jinli Cao, Hua Wang, and Yanchun Zhang. "MDDE: multitasking distributed differential evolution for privacy-preserving database fragmentation." The VLDB Journal 31, no. 5 (2022): 957-975.

[15] Wang, Hua, and Lili Sun. "Trust-involved access control in collaborative open social networks." In 2010 fourth international conference on network and system security, pp. 239-246. IEEE, 2010.

[16] Kudtharkar, Akshatha N., Neha S. Bidarkundi, P. Geethika, Laxmi Kamoji, D. G. Narayan, and Pooja Shettar. "Attribute Based Access Control for Cloud Resources using Smart Contracts." In 2023 International Conference on Networking and Communications (ICNWC), pp. 1-5. IEEE, 2023.

[17] Mughal, Aamir, and Alex Joseph. "Blockchain for cloud storage security: a review." In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 1163-1169. IEEE, 2020.

[18] Hasan, Syed Saud, Nazatul Haque Sultan, and Ferdous Ahmed Barbhuiya. "Cloud data provenance using IPFS and blockchain technology." In Proceedings of the Seventh International Workshop on Security in Cloud Computing, pp. 5-12. 2019.

[19] Patil, Abhishekh, Amit Jha, Mohammed Moin Mulla, D. G. Narayan, and Shivaraj Kengond. "Data provenance assurance for cloud storage using blockchain." In 2020 International Conference on Advances in Computing, Communication and Materials (ICACCM), pp. 443-448. IEEE, 2020.

[20] Simmhan, Yogesh L., Beth Plale, and Dennis Gannon. "A survey of data provenance in e-science." ACM Sigmod Record 34, no. 3 (2005): 31-36.

[21] Suen, Chun Hui, Ryan KL Ko, Yu Shyang Tan, Peter Jagadpramana, and Bu Sung Lee. "S2logger: End-to-end data tracking mechanism for cloud data provenance." In 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, pp. 594-602. IEEE, 2013.

[22] Aditya, C., M. Akash, P. Akash, M. Amitkumar, K. Nagarathna, D. Suraj, D. G. Narayan, and S. M. Meena. "Claims-Based VM Authorization on OpenStack Private Cloud using Blockchain." Procedia Computer Science 171 (2020): 2205-2214.

[23] Ko, Ryan KL, and Mark A. Will. "Progger: An efficient, tamper-evident kernel-space logger for cloud data provenance tracking." In 2014 IEEE 7th International Conference on Cloud Computing, pp. 881-889. IEEE, 2014.

[24] Sun, Lian-Shan, Xue Bai, Chao Zhang, Yang Li, Yong-Bin Zhang, and Wen-Qiang Guo. "BSTProv: Blockchain-Based Secure and Trustworthy Data Provenance Sharing." Electronics 11, no. 9 (2022): 1489.

[25] Zyskind, Guy, Oz Nathan, and Alex Pentland. "Enigma: Decentralized computation platform with guaranteed privacy." arXiv preprint arXiv:1506.03471 (2015).

[26] Li, Haochen, Keke Gai, Zhengkang Fang, Liehuang Zhu, Lei Xu, and Peng Jiang. "Blockchain-enabled data provenance in cloud datacenter reengineering." In Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure, pp. 47-55. 2019.

[27] Worley, Carl, Lu Yu, Richard Brooks, Jon Oakley, Anthony Skjellum, Amani Altarawneh, Sai Medury, and Ujan Mukhopadhyay. "Scrybe: A second-generation blockchain technology with lightweight mining for secure provenance and related applications." Blockchain Cybersecurity, Trust and Privacy (2020): 51-67.

[28] Darwish, Marwan Adnan, Eiad Yafi, Mohammed A. Al Ghamdi, and Abdullah Almasri. "Decentralizing privacy implementation at cloud storage using blockchain-based hybrid algorithm." Arabian Journal for Science and Engineering 45 (2020): 3369-3378.

[29] Gouru, Navya, and NagaLakshmi Vadlamani. "DistProv-data provenance in distributed cloud for secure transfer of digital assets with Ethereum Blockchain using ZKP." In Cyber Warfare and Terrorism: Concepts, Methodologies, Tools, and Applications, pp. 866-890. IGI Global, 2020.

[30] Wang, Feiyu, Jian-Tao Zhou, and Xu Guo. "BMDP: Blockchain-Based Multi-Cloud Storage Data Provenance." In 2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 703-708. IEEE, 2023.

[31] Jyoti, Amrita, and R. K. Chauhan. "A blockchain and smart contract-based data provenance collection and storing in cloud environment." Wireless Networks 28, no. 4 (2022): 1541-1562.

[32] Rosado, Tiago, and Jorge Bernardino. "An overview of openstack architecture." In Proceedings of the 18th International Database Engineering Applications Symposium, pp. 366-367. 2014.

[33] Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum project yellow paper 151, no. 2014 (2014): 1-32.