A Blockchain-Based Approach for Secure Data Migration From the Cloud to the Decentralized Storage Systems

Hooria Khan, National University of Computer and Emerging Sciences, Pakistan
Ehtesham Zahoor, National University of Computer and Emerging Sciences, Pakistan*
Sabina Akhtar, Bahria University, Pakistan
Olivier Perrin, Universite de Lorraine, France

ABSTRACT

The use of cloud computing has been on the rise. However, there are many challenges associated with the cloud, such as high bandwidth requirements, data security, vendor lock-in, and others. The recent rise of decentralized file systems (DFSs) can help mitigate some of these challenges. However, they have some limitations of their own, and the current solutions do not provide any mechanism for implementing access control policies. This becomes a hurdle for migrating sensitive data from the cloud as the associated authorization policies cannot be migrated to the DFSs. In this paper, the authors address the problem of migrating data and associated authorization policies from the cloud to the DFS. They have applied the approach on the content and policies from an actual cloud provider, and it migrates data from AWS S3 to the IPFS, and the resource-based authorization policies specified at AWS are added to a custom blockchain solution. The authors have provided implementation details to justify the practicality of the approach.

KEYWORDS

ABAC, Access Control, Authorization Policies, AWS, Blockchain, Cloud Computing, Decentralized, IPFS, RBAC

INTRODUCTION

Cloud computing has gained significant popularity during the evolution of communication and information technology. Major cloud service providers are Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP). Cloud computing offers elastic and on-demand network access to a shared pool of computing resources. The resources include storage, networking, computation, security, and others. A number of organizations use Cloud computing to store a large amount of data remotely instead of keeping it on local devices. The services provided by the Cloud require wide bandwidth with high-speed internet that limits its adoption by many end users. Similarly, vendor lock-in is another challenge in the Cloud computing environment and it is difficult to migrate data from one Cloud provider to another. Recently, decentralized storage system has been introduced for storing the data securely without the third-party assistance. Distributed file system (DFS) is one of its application in which data chunks are stored on different peers over the network. Different implementations include Interplanetary file system (IPFS), {SWARM}, and {SIA} and others. The
interplanetary file system is a peer-to-peer network that provides an alternative solution for storing a large amount of data and provides a platform that works independently from central servers. IPFS uses the concept of storing data based on content based-addressing. It works by dividing data into fixed-size chunks, distribute them over the entire network and then constructing a hash table. Distributed file systems can thus facilitate existing Cloud users to store data locally, giving them more control over the data.

Security is a major concern for the sensitive and private data. Authorization or access control policies allow determining who can access what resources based on some attributes or roles. All major cloud providers provide access control or authorization services, for example, Amazon provides Identity and Access management (IAM) service to specify authorization policies. Migrating private or sensitive data from the cloud to the distributed file system (DFS) is not feasible unless one can migrate the authorization policies, associated with the data on the Cloud, to the DFS. The current implementations of DFS, such as IPFS and Sia, do not provide any authorization policies specification mechanism and thus it is difficult for data to be migrated from the Cloud. The blockchain is another decentralized storage system that works without any central authority and stores data in the form of a list of blocks that are linked using cryptographic hash of previous block. Data in blockchain is stored in the form of blocks linked together to form an immutable chain. Every time a new transaction occurs in the blockchain, it is added to the ledger and sent to all network peers. As the blockchain is a decentralized solution similar to the design of DFS, it is natural to extend the concept of DFS and use a Blockchain for the specification of authorization policies.

This paper addresses the problem of migrating data, and associated authorization policies, from the Cloud to the DFS. The authorization process has remained a highly active research area since the last few decades and several approaches have been proposed to address different aspects related to authorization. In this context, some recent approaches attempt to solve different aspects related to authorization services provided by the Cloud, such as based on IAM service by AWS (Zahoor, Bibi, & Perrin, 2019; Zahoor, Ikram, Akhtar, & Perrin, 2018). However, there exists no approach which handles the case of data and authorization policies migration from the Cloud to the DFS, the problem being addressed in this work. The proposed approach has been mapped to migrate data from AWS S3 to the IPFS and the resource-based authorization policies specified at AWS are added to a custom blockchain solution. Specifically, the contributions of the proposed approach include:

- **Content migration**: In this approach, data items are migrated from Amazon S3 to the interplanetary file system. Different data types are stored in the cloud and transferred to IPFS and distributed in the entire IPFS network.

- **Custom blockchain**: A purpose-built custom blockchain has been provided for the storage of authorization policies associated with the data being migrated.

- **Authorization policies on the chain**: The proposed approach transfers the access control policies, for instance attached with S3 buckets, to the custom blockchain solution. The authorization policies are stored within the blocks which are connected by cryptographic hash functions to make an immutable ledger.

- **Policies-aware data access**: A custom DFS client has been designed and implemented which presents the files to the user. It uses the policies from the blockchain, and data stored at the DFS to list and control access.

- **Implementation**: Detailed performance evaluation results and implementation details have been provided to justify the practicality of the approach.

**BACKGROUND AND RELATED WORK**

This work addresses the problem of migrating data, and associated authorization policies, from the Cloud to the Distributed file systems (DFS). There exists no approach in the literature to handle the
case of data and authorization policies migration from the Cloud to the DFS. This section provides a brief overview about the problem domain and will discuss the state-of-the-art approaches for handling different related aspects.

**Cloud Computing**

The use of the Cloud computing has been constantly on the rise. On-demand dynamic and elastic resource provisioning make the Cloud computing appealing to consumers. Using the internet, Cloud users can access services from anywhere and on any device. The resources provided by the Cloud include storage, networking, computation, security and others. The rise of the Cloud also stems from the advantage of reducing hardware and software costs. It also reduces maintenance costs as in the organization do not need to maintain software and hardware. All applications are running in cloud servers and must be maintained by the providers. Cloud computing service models include Infrastructure-as-a-service (IaaS), Platform-as-a-services (PaaS) and the Software-as-a-platform (SaaS). Major Cloud providers include Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP).

Amazon Web Services (AWS) is one of the most popular Cloud providers providing services from datacenters that are spread across the world, called availability zones (AZs). The services provided by AWS being used in this work include Amazon Simple Storage Service (S3) and Amazon Identity and Access Management (IAM) service. Amazon S3 provides a scalable storage area for data backup, collection, and analytics. The data is organized in units called buckets and access policies can be attached to the buckets. AWS IAM handles the authorization and authentication aspects for the services provided by the AWS. Using IAM, administrators can create users and groups and allow (or deny) those users or groups to access resources. The IAM workflow includes the following:

- A principal which is used as a role or an application that performs actions on AWS resources.
- Authentication: It includes confirming the principal’s identity, which is trying to access an AWS product.
- Request: A principal sends a request to AWS resources to get access.
- Authorization: IAM allows access only if part of the request and policy match.
- Actions: It is used to view, delete, or edit a resource.
- Resources: A set of activities that are performed on resource related to AWS.

IAM consists of different components like user, groups, policies, and others. A user is the subject who wants to get access to some resources of AWS. Each IAM user is associated with one account only. Similarly, the collection of users forms the IAM group. Managing groups is relatively easy as the owner can set the permissions for the group, and this permission is automatically applied to all the users of the group. IAM policies set permission and control access. It is stored in the form of JSON documents. For example, a policy could allow a specific user to access the bucket of Amazon S3. The policies contain the following information:

- Who can access resources?
- What action needs to be taken.
- Which AWS resources a user can access.

In IAM, roles are also important as it is a set of permissions that defines what activities are allowed and denied. It is similar to a user who can be accessed by any entity like an individual or AWS service.

The cost benefits are the key driver for the organization to migrate to the cloud services, but security risk is a major issue. According to Pearson, (Pearson, 2013), cloud computing security is the biggest issue for any organization. The main security risk is leakage of data or risk of unwanted
access by some unauthorized parties, resulting from inadequate data handling by cloud providers or inadequate access controls. Similarly, the lack of interoperability between different cloud servers following different storing data standards is also a significant issue. As a result, it is difficult for customers to migrate from one cloud provider to another.

**Blockchain**

The concept of blockchain gained maturity from a cryptocurrency, Bitcoin, introduced in 2008 by Satoshi Nakamoto. The blockchain is a data-structure for storing information that is difficult to alter. In a blockchain network, all peers maintain the same copy of a database called the ledger distributed across the entire network. There is no central authority to manage all, and no single node gives privileges. Blocks are added in a network by a consensus among nodes and construct a chain. Every time a new transaction occurs on the blockchain, it is added to the peer’s ledger. The distributed database maintained by nodes is known as Distributed Ledger Technology (DLT).

In a public blockchain, all data stored in a system are visible to all the nodes. A chain of blocks starts from the genesis block, and all blocks are connected through a cryptographic hash function. Each block contains a header and a series of transactions, and each header contains link pointers of the previous block. So, if anyone wants to tamper with any block, he has to change all the headers that point to the previous node, so this feature makes blockchain immutable and tamper resistant.

Blockchain uses different consensus algorithms for adding a block in a chain. The main idea behind consensus is to make all peers agree on one dataset. Proof of work (PoW), proof of stake (PoS) (King & Nadal, 2012), Byzantine Fault Tolerance algorithm (Sousa, Bessani, & Vukolic, 2018), and the Ripple algorithm (Schwartz, Youngs, & Britto, 2014) are some famous consensus algorithms used in different application depending upon the requirements. Bitcoin uses PoW consensus algorithm to add blocks in a chain. In this algorithm, each node in the network calculates a hash value. The requirement is to calculate a value that must be equal to or less than a specific given value. When one node achieves a target value, it would be broadcasted to the whole network to check its hash value. If other nodes validate the block, it is appended in the blockchain. The nodes that calculate the hash are called miners, and this process is called mining. In the PoS consensus algorithm, the miners have to prove ownership of a certain amount of currency. This method would lead to many problems as people with more currency would attack a network, so many solutions are proposed to combine proof of stake consensus like DPOS (Delegated proof of stake). Depending on the need of the application, blockchain solution can be either a permissioned or a permissionless blockchain.

The permissionless blockchain is accessible to all participants. There is no restriction, and no one has complete control of the network. This feature helps to make data more secure and immutable. The blockchain’s authority is equally divided among nodes in the network, due to which it is fully distributed. It is mainly used by cryptocurrencies like Bitcoin, Litecoin, and Ethereum. The most widely used permissionless blockchain solution is Ethereum initially built by Vitalik Buterin. It is a public open source-based distributed computing platform for building DApps.

The difference between bitcoin and Ethereum is based on capability as Ethereum focuses on running the programming code called a smart contract. In Ethereum, instead of mining a bitcoin, miners work to earn Ether, a cryptocurrency of the network. Similarly, gas is required to be paid for every operation performed in a network. The gas limit refers to the maximum amount of gas that the user is willing to spend on a particular transaction.

One example of a permissioned blockchain solution is Hyperledger Fabric. It is a platform that provides a solution to distributed ledger underpinned by a modular architecture delivering a high degree of flexibility, scalability, and confidentiality. The Linux Foundation founded the Hyperledger project in 2015 to use blockchain technology across the industry. Like other blockchains, it has its ledger, smart contract, and a system in which participants interact. It offers the ability to create channels, allowing a group of participants to create a ledger for a transaction. This is an essential option as in a network; some competitors do not want every transaction they make. Hyperledger
Fabric smart contract is written in chain code. It supports several programming languages like Go, Node, etc. It is made for enabling consortium blockchain with different levels of permissions. The fabric uses resources more efficiently. It has fewer nodes than a public chain and computes data in a parallel fashion. In its architecture, there are two virtual nodes called “Endorsers” and “Consensus Nodes”. The Endorser has the state and can validate and propagate transactions and chain code. The work of the consensus node is to order an already validated transaction. So, it implements a better division of labor in a network as compared to bitcoin.

**Distributed File Systems (DFS)**

A distributed file system is a file system spanning multiple nodes and possibly multiple locations. It allows users to share data and storage resources by using the common file system. Accessing the file relies on client/server architecture. A key feature of DFS is the high availability of information because it continues its functionality after failure as data is replicated on different nodes. DFS also refers to transparency by hiding unnecessary details from a user like replication transparency as a user do not know the existence of replication of data.

- **Swarm**: It is a content distribution service based on Ethereum and data accessibility in a network is location independent. With deep integration to Ethereum, it has benefits of both smart contract and Filecoin. Two significant features of the Swarm distinguished from other DFS are the ‘upload and disappear’ (upload content and allowed to go offline) and its incentive system. For the incentive mechanism, the service of delivering chunks is chargeable, and nodes can exchange resources with a resource. To motivate the nodes, Swarm uses SWAP (Swarm Accounting Protocol). In Swarm, nodes store data chunks and profit for selling when some retrieved request is received; otherwise, the request is sent to the nearest neighbor node. The main objective of Swarm is to provide infrastructure for developing dApp.

- **Storj**: Another decentralized platform that stores data without the assistance of third-party services is Storj (Wilkinson, Boshevski, Brandoff, & Buterin, 2018). It is a peer to peer network which uses client-side encryption. Initially, the file is divided into smaller chunks, and the distributed hash table is constructed where all information regarding shards are saved. The file’s uploader is the only person who has access to shards of its original file as the data owner key encrypts the hash table. In the network, farmers are used who lend their free drives to store those shards of file. They receive micro-payment for storing and maintaining a file. Therefore, they are incentivized to remain active on the network.

- **Sia**: It is a blockchain-based cloud platform that provides a storage system without the assistance of a third party. Peers in Sia can rent their hard drives and get rewards. The two main components of Sia’s network are renters and the host. The hosts rent their storage area to renters by promoting their storage resources. Hosts also have the right to refuse renters if data is illegal or too sensitive.

- **IPFS (Interplanetary File system)**: It is a peer-to-peer version-controlled file system (Benet, 2014) in which files are stored in the form of content-based addressing. It is one of the most popular implementations and it will be explained in detail in the next section.

**Interplanetary File System (IPFS)**

It is a peer to peer version-controlled file system (Benet, 2014). In which files are stored in the form of content-based addressing. It uses the concept of Distributed Hash Table, Bitswap (inspired by BitTorrent), and Merkle DAG (inspired by Git Protocol). The main aim of IPFS is to replace HTTP because large files cannot be transferred through HTTP. Similarly, client-server is used, which leads to low latency; real-time streaming is also difficult in HTTP. All these failures are handled in IPFS. Unlike HTTP, which stored data based on IP, IPFS stores data based on content addressed. When any data is uploaded on the IPFS network, IPFS returns its hash, and the data is requested using
this hash. Anyone providing storage to the IPFS network, gets incentives with crypto tokens. Data is replicated and distributed throughout the network, which increases data availability and provides backup. When data is requested from the user, it is searching for the nearest copy of that data, which leads to high availability and reduces bottleneck points. Following are important features of IPFS:

- **Distributed Hash Table:** It is used to store data across all nodes in a network. It is similar to a hash table. Using DHT, any node on the network can request an object (file, video, etc.) corresponding to a hash key.

- **Block Exchange:** It features technology used in BitTorrent (knows as BitSwap) to exchange data in a network. It is a peer to the peer file system which exchanges data between untrusted swarms. Bitswap protocol is one example that makes IPFS unique from other DFS.

- **Merkle DAG:** Merkle DAG is the best part of IPFS due to content-based addressing storage and tampers resistance features. IPFS is a data structure where hashes are used to reference data blocks and objects in a DAG. It used a Merkle Tree or Merkle DAG tree similar to the one used in the Git Version Control system. If the file exceeds the block size, it divides a file into chunks, stores it on different peers in the form of hash, and constructs the Merkle DAG tree. Moreover, it tracks all the versions of the file on the network in a distributed way.

- **IPFS Nodes:** Every node in a network has a unique Node ID, a hash of the public key as shown in Figure 1. They store files in their local storage and gets incentives. Each node maintains a DHT, used to save information about nodes and what data they stored. IPFS has many advantages, like users in a local network can communicate with each other, even if the internet is down. Similarly, it is fully distributed as no server is required, which reduces the network’s cost.

**Figure 1. Unique Node ID in form of hash**

![Unique Node ID in form of hash](image)

- **Content Addressing:** The hash refers to IPFS objects. If the user wants some particular file, then the IPFS server will ask for the hash corresponding to the file. It uses content addressing at the HTTP layer. All hashes in IPFS using base 58 and starts with “Qm”.
– **Versioned File Systems**: IPFS represents the data structure by using the technique of Git that controls versions for a file system. It used the property of Commit, which points to the file system like names parent0, parents1, etc.

– **Distributed hash table (DHT)**: IPFS combines the features of DHT (distributed hash table) and an SFS (self-certifying file system). Using DHT is more scalable, fault tolerant, and decentralized data storage compared to the cloud. Moreover, by using a self-certifying file system, no permission is required for data exchange. IPFS works similarly to the web as uploading a file to the IPFS system will get the unique cryptographic hash string through which data is accessed. Moreover, the hash string works just like a URL on the web. Filecoin (Benet & Greco, 2018) is adopted as an incentive mechanism for IPFS.

**Authorization Process**

Authorization or access control process allows for the controlled access of resources. The authorization process follows the authentication process, which is used to verify the identity of someone they claim to be. It includes a login/password. In comparison, authorization process uses policies to specify and enforce who can access which resources, under which conditions, and for what purpose. It involves the Subject, Object, and Action. Different access control policy models are used like UBAC (User-based access control), ABAC (Attribute-based access control), RBAC (Role-based access control) and others.

– **RBAC (Role-based access control)**: The role-based access control model is based upon the roles in the organization. The administrator determines the roles and access privileges that a role should have. This access control facilitates that one user can be assigned to multiple roles. The RBAC model is easier to implement. Many organizations use this because RBAC does not need to be changed every time they leave the organization or change jobs. Similarly, new employees can be granted access relatively quickly. Although RBAC has many benefits, it has some significant disadvantages like scalability, and role explosion.

– **ABAC (Attribute-based access control)**: This type of access control depends on the characteristics called “attributes”. The attribute may be user attributes, environmental attributes, and resource attributes. User attributes include user’s name, role, organization name, ID. Environmental attributes can include the time of access, location, and others. Resource attributes include things like resource owner, filename, and others. ABAC can be considered as a more generic model as it subsumes RBAC by considering roles as an attribute.

**RELATED WORK**

The Authorization domain has been highly active and widely studied research direction and a number of approaches have been proposed in the literature to handle different aspects related to the authorization. The focus of this section is to highlight the approaches which either use blockchain for the access-control or use a DFS implementation, such as IPFS.

The framework is proposed in the (Wang, Zhang, & Zhang, 2018) combines the features of IPFS, Ethereum, and attribute-based encryption for achieving fine-grained access control over data. The authors proposed a scheme in which no third-party key generator is required. To achieve fine-grained access control, the data owner can generate secret keys for users and encrypt the data using an access policy. To secure personal data from untrusted entities, they presented (Zichichi, Ferretti, D’Angelo, & Rodríguez-Doncel, 2020) smart contract-based solution. They used the concept of trust from multiple nodes for granting access to the data. The proposed scheme introduced two possible alternative methods: a Secret Sharing scheme and Threshold Proxy Re-Encryption scheme and compared both in terms of execution time. When compared to the proposed approach, their approach is based on attribute-based encryption and not the authorization aspects. The modified version of IPFS
with Ethereum is used in (Steichen, Fiz, Norvill, Shbair, & State, 2018). They have written a smart contract for maintaining an access control list, and IPFS software enforces it (Steichen et al., 2018). When the user uploads a file, it goes to IPFS and is divided into chunks. These chunks have a content identifier (CID) collected and added to the smart contract. The permission storage also verifies the transaction corresponding to the CID and ensures it is not empty or duplicated with the same owner name. After verification, control goes back to IPFS, and the file is stored in it successfully. If someone wants to retrieve a data file, the request is sent to the data owner, and he denies or grants access by using smart contracts. Their proposed approach does not handle the case of data and associated authorization policies migration from the Cloud. The authorization model they have provided is not expressive enough to model Cloud authorization policies such as the ones provided by AWS IAM. In this proposed approach, a custom-built blockchain instead of Ethereum giving more control and eliminating the need for transaction costs.

In (Maesa, Mori, & Ricci, 2019, 2018) authors have proposed to codify the access control policies by using smart contracts on a blockchain. They use a scenario where the resources to be protected are smart contracts. The attribute-based-access control policy (ABAC) is written in XACML using different points like PEP, PAP, and AMs. The authors’ aims are that the resource owner cannot fraudulently deny access without leaving an auditable trace of the behavior. In (Ali, Dolui, & Antonelli, 2017), authors proposed a decentralized access model for IoT using network architecture called modular consortium for IoT and blockchain. The paper’s main contribution is to implement a software stack of smart contracts with IPFS to give authority to IoT users over their data. In (Cruz, Kaji, & Yanai, 2018), authors have used RBAC-SC framework to describe access policies. Their motivation is that the user is only able to access data based on its role. A smart contract is used for the creation of the user-role lists. In (Kiran, Dharanikota, & Basava, 2019), the model is proposed for securing user data access by using the Ethereum framework. They used two types of smart contracts: a policy contract used by the data owner to specify different policies regarding how much data is revealed to the particular requestor. The other is a data access smart contract. The user uploads its details in IPFS and stores the root of the data in the blockchain. The requestor does not deploy any website but interacts with the data owner to access user data. The data owner returns the details to the requestor based on the permission that has been granted to the website by the user.

Data sharing platforms depend upon a trusted third party (TTP) which lacks transparency and security. To overcome this issue, (Naz et al., 2019) have proposed a system based on blockchain, IPFS, and encryption. The proposed scheme achieves security and authenticity of the owner by using smart contacts. Likewise, (Zichichi, Ferretti, & DAngelo, 2020) used the decentralized storage system (DFS), and (Distributed Ledger Technologies) DLTs features for the development of decentralized Personal information management system. For real-world evaluation, they implemented Intelligent Transportation system use-case. In (Nyaletey, Parizi, Zhang, & Choo, 2019), a new approach is proposed based on IPFS and Hyperledger technology, which can provide audit access to file as it tells who has downloaded the files, thus provide evidence for both dispute resolution and forensic. Another scheme is proposed in (Zaghloul, Li, & Ren, 2018) that enables patients to control the sharing of their health record by using an attribute-based encryption scheme in a distributed file system. They show that along with privacy protection, it also provides the facility of data confidentiality. The proposed approach can also be compared with Smart Vault (Smart Vault, 2019). The platform provides a facility for the data owner to share its file with a specific group of people. The access control list is managed by using a smart contract. Their proposed approach does not handle the case of data and associated authorization policies migration from the Cloud. The authorization model they have provided is not expressive enough to model Cloud authorization policies such as the ones provided by AWS IAM. In this proposed approach, a custom built blockchain has been provided instead of Ethereum, giving more control and eliminating the need for transaction costs.

In (Younis, Kifayat, & Merabti, 2014), access control model is proposed to ensure the secure sharing of resources and support different access permission to the same cloud users. The authors
proposed access control as a service (ACaaS) for the public IaaS cloud (Wu, Zhang, Ahn, Sharifi, & Xie, 2013). The architecture configures and manages multiple access control policy models. A novel EHRs sharing framework is used that combines features of blockchain and IPFS on mobile cloud computing is presented in (Nguyen, Pathirana, Ding, & Sen-eviratne, 2019). They designed an access control mechanism to secure EHRs sharing among different entities (patients and medical providers). They used mobile Android application and AWS for testing and considered different security analysis parameters like avoiding a single point of failure, availability, and integrity. Their proposed access control model can identify and prevent unauthorized access to data of e-health. Similarly, (Guo, Shi, Zhao, & Zheng, 2018) handles the issue in e-health with multiple authorization using blockchain technology. In their proposed approach, the patient has rights to disclose their personal data to trusted parties only.

In (Gan, Saini, Zhu, Xiang, & Zhang, 2020) authors have proposed a blockchain-based access control scheme with an incentive mechanism in e-health systems. In their design, a patient has a right to sharing their record with a particular person. Moreover, the incentive mechanism encourages patients to share their medical records actively. Another blockchain-based solution is proposed in (Zyskind, Nathan, & Pentland, 2015), which allows data owners to control their data and get better privacy. Their primary concern is fine-grained access control at the time of sign-up. To solve privacy and security issues in big data, (Es-Samaali, Outchakoucht, & Leroy, 2017) have proposed blockchain-based access control framework to enhance security in the system.

In (Do & Ng, 2017) authors have proposed a decentralized storage system using blockchain and implement privacy keyword search scheme in the proposed system. In (Jiang, Guo, Liang, Lai, & Wen, 2020) authors proposed a blockchain-based solution for privacy keyword search scheme in the decentralized storage system. The proposed method is still in the theoretical stage and the stability and feasibility of the system has not been established. In (Zhang, Kasahara, Shen, Jiang, & Wan, 2018), an IoT based solution is given to implement distributed and trustworthy access control policies. They implemented multiple access contracts (ACCs) in Ethereum blockchain and manage access control between various subjects and object pairs.

In summary, the authorization process plays a critical role in implementing security principles and an erroneous policy can lead to severe consequences. The challenges associated with the authorization process have thus led to a number of approaches handling different related aspects. The authorization models have matured from the earlier models including Discretionary Access Control (DAC) and Mandatory Access Control (MAC) models to current implementations based on Role Based Access Control (RBAC) and Attribute-based access control (ABAC). In this context, some recent approaches attempt to solve different aspects related to authorization services provided by the Cloud, such as based on IAM service by AWS (Akhtar, Zahooor, & Perrin, 2017; Zahooor, Asma, & Perrin, 2017). The recent rise of decentralized file systems allows for data storage in a decentralized manner spanning multiple nodes and possibly multiple locations. Interplanetary File system (IPFS) is amongst the most popular implementations of such file systems. IPFS allows for content addressing and a resource is represented by its hash instead of location-based addressing as used by the traditional Web. Naturally, the authorization process for handling these resources needs to be decentralized as well and the traditional centralized authorization approaches are not suitable. This work addresses the problem of migrating data, and associated authorization policies, from the Cloud to the Distributed file systems (DFS). As the authorization process itself needs to be decentralized, a custom-built blockchain is developed for the storage of authorization policies associated with the data being migrated. The proposed approach thus transfers the access control policies, for instance attached with S3 buckets, to a custom blockchain solution. There exists no approach in the literature to handle the case of data and authorization policies migration from the Cloud to the DFS.
This work addresses the problem of migrating data, and associated authorization policies, from the Cloud to the Distributed file systems (DFS). Different components of the proposed approach are highlighted in Figure 2. The objectives of this work include to transfer the data files from the Cloud to the distributed file system. In DFS, no node is given the privilege and storage of all the data is based on content-addressing. In addition, all the files stored in the DFS should be protected from unauthorized access and only the subject with the right attributes can gain access to any part of the data. This can be achieved by implementing role-based or user-based access control on the data, as provided by major Cloud providers. The proposed approach has thus two main components; data migration to the DFS and the implementation of the access control on the migrated data.

To better discuss the components of the proposed approach, it needs to be mapped to existing DFS and the Cloud provider. For this purpose, Amazon Web Services (AWS) are chosen as the Cloud Provider and Interplanetary file system (IPFS) as the DFS, as shown in the Figure 2. The data files are
thus migrated from the Amazon S3 buckets to the IPFS, and the authorization policies associated with the S3 buckets, the resource based IAM policies, are then used for implementing the access control on the IPFS files. The proposed approach can be applied to other Cloud providers and DFS as well.

Content Migration from the Cloud

The services provided by AWS being used in this work include Amazon Simple Storage Service (S3) and Amazon Identity and Access Management (IAM) service. AWS IAM handles the authorization and authentication aspects for the services provided by the AWS. Using IAM, administrators can create users and groups and allow (or deny) those users or groups to access resources.

Amazon S3 provides a scalable storage area for data backup, collection, and analytics. The data is organized in units called buckets and access policies can be attached to the buckets. An object consists of any file and metadata that describes the file. For storing an object in Amazon S3, the user uploads the files in buckets, as shown in the Figure 3.

Figure 3. AWS S3 buckets

| Name         | AWS Region             | Access                           |
|--------------|------------------------|----------------------------------|
| adminonly    | US West (Oregon) us-west-2 | Bucket and objects not public   |
| managementdocs | US West (Oregon) us-west-2 | Bucket and objects not public   |
| patientsdata | US West (Oregon) us-west-2 | Bucket and objects not public   |

Buckets are just like folders where users can upload different types of files. The user must mention the bucket name where data files are stored. It cannot be changed once the buckets name is set. AWS also allows the user to select a region of the bucket. The bucket’s region is just a geographic location where AWS provides physically separated servers to increase throughput for accessing and retrieving data.

For the proposed approach, the first step is to transfer these object files (the actual content) from the S3 buckets and distribute these to IPFS (interplanetary file system). IPFS is a peer to peer versioned controlled system. It stores data based on content-addressing. When some files are transferred to IPFS from Amazon S3, it is divided into chunks of 256Kbs (the default size of the chunk in IPFS) and distributed in the entire network. Each chunk is identified by its unique chunk ID and stored on different peers. The distributed hash table (DHT) maintains the chunk IDs and corresponding peer IDs that store the chunks. When the user requests the file, it traverses to nodes where the hash exists using the DHT. When the user gets a root hash in the form of $Qm$, they can retrieve a specific file. All files are stored with a unique hash, so data duplication is not possible. Moreover, users can pin essential files to their node.
Go routines are used to migrate all buckets from Amazon S3 to IPFS. For example, there is an item stored in the Amazon bucket named: admin.txt. When admin.txt file is uploading to IPFS using the API, it is divided into chunks of 256kb. These chunks are in the form of a hash that represents the content in it. A CID is created for each chunk and these chunks combined to form a hierarchical data structure called Merkle DAG. IPFS network also maintains distributed hash table against the information of peers ID that stored the chunks and chunks of the file as shown in the Figure 4.

Now if any user wants that admin.txt he must mention the root hash of it. IPFS network checks Merkle DAG tree for the existence of the file and uses its DHT for the discovery of peers that stored that chunks and content of the file admin.txt.

**Policies Specification at the Cloud**

AWS IAM handles the authorization and authentication aspects for the services provided by the AWS. For the authorization aspects using IAM, administrators can create users and groups and allow (or deny) those users or groups to access resources. The IAM workflow includes the following:

- A principal which is used as a role or an application that performs actions on AWS resources.
- Authentication: It includes confirming the principal’s identity, which is trying to access an AWS product.
- Request: A principal sends a request to AWS resources to get access.
- Authorization: IAM allows access only if part of the request and policy match.
- Actions: It is used to view, delete, or edit a resource.
- Resources: A set of activities that are performed on resource related to AWS.

Amazon manages authorization by creating policies and attaches them to IAM identities or resources. IAM identities include roles, users, or groups. In AWS, the policy is evaluated when an
IAM principal makes a request. Amazon offers multiple types of policies which are used by user or organization to restrict access to data. Some of the policies are as follows:

- **Identity-based policies**: They are the inline policies that allow access based on identities like username, group, or role.
- **Resource-based policies**: They are attached directly to the resources. For example, users can attach resource-based policy directly to S3 buckets. It grants permissions based on the Principal.
- **Access control lists (ACLs)**: They are similar to resource-based policies, but it is the only policy type that does not support JSON document structure. It is cross-account permission that grants access based on the IAM principal.

Amazon access polices mostly support JSON document structure as shown in Figure 5. Access policy contains different components such as:

- **Statement**: It is the main policy container that contains different elements. Users can add more than one statement to a policy.
- **Effect**: It indicates whether access policy is denied or allowed.
- **Resource**: It specifies the list of resources to which actions are applied.
- **Action**: It specifies the list of actions the principal is allowed or denied.

---

Figure 5. An example JSON based AWS authorization policy

```json
{
    "Statement": [
        {
            "Sid": "Stmt01",
            "Effect": "Allow",
            "Action": [
                "aws-portal:ViewBilling"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```
AUTHORIZATION POLICIES MIGRATION FROM THE CLOUD

The previous sub-sections discuss how the content can be migrated from the Cloud and how the policies are specified at the Cloud. The key question however remains that how these policies can be stored and used in a decentralized setting. As the content storage is decentralized (based on DFS) it is important to use a decentralized approach for the policies specification as well. This can be achieved using a blockchain based approach. In a blockchain network, all peers maintain the same copy of a database called the ledger which is distributed across the entire network. Blocks are added in a network by a consensus among nodes and construct a chain. For the decentralized storage of the authorization policies, a custom-built blockchain is used specifically optimized for the proposed approach.

In proposed custom-built blockchain, central node set policies or rules for joining other nodes or peers. The central node is named Satoshi that starts a network using AWS login and migrates Amazon access policies to the proposed custom-built blockchain. When policies are migrated, they are added to blocks in the private blockchain and formed an immutable chain. Each block stored the hash of the previous block along with an access policy that prevents data tampering, as shown in Figure 6 and Figure 7. The complete JSON document representing the resource-based policy from the Cloud is stored to the blockchain. It contains information about subject, object and the actions and the JSON document is later parsed from the blockchain when a request is made. When other users want to join the network, they used their Amazon credentials and act as nodes in the blockchain. Access policies are checked when any user wants to join the network, and blockchain is replicated on each node. The user can see all the data files to which he has granted permission. That’s how by using features of IPFS and blockchain, data is migrated from the Cloud and authorization policies are applied at the decentralized network. In proposed custom-built blockchain username is a key feature stored under the principal component in AWS policies. All-access are granted based on it. The proposed custom-built blockchain is replicated to every joining node and it traverses whole blocks for granting access to files based on AWS username.

Figure 6. Blocks (for S3 buckets) of the custom blockchain
WORKFLOW OF CLIENT REQUESTS

Any user who wants to join the network for accessing data files has to mention his AWS credentials and his AWS username. After the authentication process succeeds, the user logs into the system and the proposed custom-built blockchain replicates on their system. After this authorization procedure starts, all blocks in blockchain stored access policy and all are linked. When a user wants to access a particular folder, all blocks in blockchain traversed and check folder name against AWS username if folder access policy has its AWS user name access is granted and root hash of all files stored in the folder are displayed. These all hashes are in the form of ‘Qm’. The user used these root hash on the IPFS network and the IPFS network collects all chunks from peers using a distributed hash table and rearranged these chunks and provide the file the user is demanding.

IMPLEMENTATION DETAILS

To justify the practicality of the proposed approach, a Web application has been developed using Golang to integrate various components. Let us consider the case of S3 buckets as shown in Figure 3. The process starts when the Satoshi node has joined the network using its AWS access ID, access Key, and buckets region, as shown in the Figure 8-a. Satoshi node acts as a main node for the network. It is responsible for the data migration from Amazon S3 buckets and for authorization policies migration from the Cloud to proposed custom-built blockchain.
Proposed application uses Golang “text/template” package to generate HTML output against the input code. To use AWS SDK in Golang, a session has to be created containing configuration information for the service client used to interact with AWS services. The session can include information about the region, credentials, and some additional request handlers. To create a session in Golang application, some base packages need to be imported which help create sessions for bucket region and credentials used for interacting with AWS services. After this, a new instance of the S3 client is created to help the proposed application interact with the Amazon Web server (AWS). On the successful login, the user can see all the buckets (ListBuckets API function) and their items (ListObjects API function) as shown in Figure 8-b.

The user can then start the migration process from Amazon S3 buckets to the distributed file system (IPFS). IPFS is written in Golang, so the interaction between IPFS and web application becomes easier using go-ipfs-api. Before the migration process can begin, it is required to initialize your node in IPFS network, IPFS daemon command is used to display some information such as API server listening port, Gateway server listening port and others. When the node is ready, the Add file function is called to add data items in the IPFS network. This function uploads the file to IPFS and returns hash in the form of ‘Qm’ as shown in the Figure 8-c.

After data files are added to the IPFS network, there is a need to transfer the access control policies attached to buckets migrated from the Cloud to the custom private blockchain. Let us consider the case of a user named Alice having access only the adminonly S3 bucket, as implemented by the authorization policy on the Cloud as shown in Figure 9-a.

![Figure 8. Content migration by the Satoshi node](image1)

![Figure 9. Authorization policy for S3 buckets named adminonly and managementdocs](image2)
Let us consider another user named Bob, who does have access to only the `managementdocs` bucket as shown in Figure 9-b. The policies migration process is initiated from the Satoshi node by clicking on the policy button, and this will add policies to proposed custom-built blockchain. In the blockchain, all blocks contain access policies and linked together that avoids data tempering as shown in the Figure 6 and Figure 7.

At this stage both the content and authorization policies have been migrated from the Cloud to the IPFS and the blockchain. When Alice logs in with her credentials, she has access to `adminonly` bucket, but she is unable to access the management bucket. Similarly, Bob can only access the `managementdocs` bucket.

**PERFORMANCE EVALUATION**

For the performance evaluation, multiple test cases have been considered. As the proposed approach migrates data from the Cloud to IPFS, there are many costs (in terms of time) involved in the process. The first being the cost to fetch the resources from the Cloud. Then there is some cost to add the policies associated with the resources to the proposed custom-built Blockchain. Further, there is some cost involved to transfer the files to the IPFS. Finally, whenever a user accesses a resource, it would take time to traverse the Blockchain and find the matching policy for authorization decisions.

Amongst the four different times mentioned above, the time to fetch resources (and associated policies) from the Cloud and also the time for adding these files to the IPFS strongly depends on network connectivity and conditions. From the performance evaluation perspective, it is more interesting to consider the time it takes to add resource information (including authorization policies) to the Blockchain and also the time it takes to make access decisions. For each case, it further considers the case where a block contains a single transaction or multiple (10) transactions. The source code for proposed implementation, highlighting the aspects related to the performance evaluation results, can be downloaded from the link below⁴.

---

**Figure 10. Performance evaluation results**

![Performance Evaluation Results](image-url)
The performance evaluation results are shown in Figure-10 with the X-axis showing the number of blocks being added and the Y-axis showing the time taken (average of multiple runs). The performance evaluation tests were conducted on a personal machine having Intel Core i5 CPU@2.6GHz with 8GB RAM and running MacOS Mojave 10.14.6. There are four test-cases in total with two each for measuring block insertion and authorization decision cases. The results are very encouraging and even on a personal machine, it takes few milliseconds to insert blocks to the chain. For the decision test-cases, the principal has been intentionally set such that it does not match any policies and thus the complete chain needs to be traversed for measuring the worst-case performance. The results are very encouraging for this test-case as well and the time it takes to reach an authorization decision, that is to traverse the chain for a matching policy, only takes few microseconds.

CONCLUSION

The use of the Cloud computing has been constantly on the rise. On-demand dynamic and elastic resource provisioning make the Cloud computing appealing to consumers. However, there are still many challenges associated with it, such as high bandwidth requirements, data security, vendor lock-in and others. This work addresses the problem of migrating data, and associated authorization policies, from the Cloud to the Distributed file systems (DFS). There exists no approach in the literature to handle the case of data and authorization policies migration from the Cloud to the DFS.

The proposed approach has been applied on the content and policies from an actual Cloud provider, the Amazon Web Services (AWS). In the proposed approach, data items are migrated from Amazon S3 to the interplanetary file system. Different data types are stored in the cloud and transferred to IPFS and distributed in the entire IPFS network. The proposed approach transfers the access control policies, for instance attached with S3 buckets, to proposed custom-built blockchain solution. The access policies are stored in the form of blocks that are connected to make an immutable ledger. This work proposed and implemented a custom DFS client which presents the files to the user. It uses the policies from the blockchain, and data stored at the DFS to list and control access.

One limitation of the proposed approach is that the provided implementation at this stage is just a proof-of-concept, and it can be further extended to include resources and authorization policies from multiple Cloud providers in a multi-cloud solution. This can be an interesting future research direction given that authorization policies provided by different Cloud providers are both syntactically and semantically different. Further, there is a difference in the authorization models being supported by different Cloud providers. The proposed approach can also be further extended to include other DFS solutions in addition to IPFS.
REFERENCES

Akhtar, S., Zahoor, E., & Perrin, O. (2017). Formal verification of authorization policies for enterprise social networks using pluscal-2. In I. Romdhani, L. Shu, T. Har, Z. Zhou, T. J. Gordon, & D. Zeng (Eds.), Collaborative computing: Networking, applications and worksharing - 13th international conference, collaboratecom 2017, edinburgh, uk, december 11-13, 2017, proceedings (Vol. 252, pp. 530–540). Springer.

Benet, J., & Greco, N. (2018). Filecoin: A decentralized storage network. Protoc. Labs.

Cruz, J. P., Kaji, Y., & Yanai, N. (2018). Rbac-sc: Role-based access control using smart contract. IEEE Access: Practical Innovations, Open Solutions, 6, 12240–12251. doi:10.1109/ACCESS.2018.2812844

Do, H. G., & Ng, W. K. (2017). Blockchain-based system for secure data storage with private keyword search. In 2017 IEEE world congress on services (services) (pp. 90–93). IEEE.

Es-Samaali, H., Outchakoucht, A., & Leroy, J. P. (2017). A blockchain-based access control for big data. International Journal of Computer Networks and Communications Security, 5(7), 137.

Gan, C., Saini, A., Zhu, Q., Xiang, Y., & Zhang, Z. (2020). Blockchain-based access control scheme with incentive mechanism for ehealth systems: Patient as supervisor. Multimedia Tools and Applications, 1–17.

Guo, R., Shi, H., Zhao, Q., & Zheng, D. (2018). Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems. IEEE Access: Practical Innovations, Open Solutions, 6, 11676–11686. doi:10.1109/ACCESS.2018.2801266

Jiang, P., Guo, F., Liang, K., Lai, J., & Wen, Q. (2020). Searchain: Blockchain-based private keyword search in decentralized storage. Future Generation Computer Systems, 107, 781–792. doi:10.1016/j.future.2017.08.036

King, S., & Nadal, S. (2012). Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. self-published paper.

Kiran, A., Dharanikota, S., & Basava, A. (2019). Blockchain based data access control using smart contracts. In Tencon 2019-2019 IEEE region 10 conference (tencon) (pp. 2335–2339). IEEE.

Maesa, D. D. F., Mori, P., & Ricci, L. (2018). Blockchain based access control services. In 2018 IEEE international conference on internet of things (itthings) and IEEE green computing and communications (greencom) and IEEE cyber, physical and social computing (cpscom) and IEEE smart data (smartdata) (pp. 1379–1386). IEEE.

Maesa, D. D. F., Mori, P., & Ricci, L. (2019). A blockchain based approach for the definition of auditable access control systems. Computers & Security, 84, 93–119. doi:10.1016/j.cose.2019.03.016

Naz, M., Al-zahrani, F. A., Khalid, R., Javaid, N., Qamar, A. M., Afzal, M. K., & Shafiq, M. (2019). A secure data sharing platform using blockchain and interplanetary file system. Sustainability, 11(24), 7054. doi:10.3390/su11247054

Nguyen, D. C., Pathirana, P. N., Ding, M., & Sen-eviratne, A. (2019). Blockchain for secure ehr sharing of mobile cloud based e-health systems. IEEE Access: Practical Innovations, Open Solutions, 7, 66792–66806. doi:10.1109/ACCESS.2019.2917555

Nyaletey, E., Parizi, R. M., Zhang, Q., & Choo, K.-K. R. (2019). Blockipfs- blockchain-enabled interplanetary file system for forensic and trusted data traceability. In 2019 IEEE international conference on blockchain (blockchain) (pp. 18–25). IEEE.

Pearson, S. (2013). Privacy, security and trust in cloud computing. In Privacy and security for cloud computing (pp. 3–42). Springer. doi:10.1007/978-1-4471-4189-1_1

Schwartz, D., Youngs, N., & Britto, A. (2014). The ripple protocol consensus algorithm. Academic Press.

Smart Vault. (2019). https://docs.lightstreams.network/products/smart-vault

Sousa, J., Bessani, A., & Vukolic, M. (2018). A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In 2018 48th annual IEEE/IFIP international conference on dependable systems and networks (dsn) (pp. 51–58). IEEE.
Steichen, M., Fiz, B., Norvill, R., Shbair, W., & State, R. (2018). Blockchain-based, decentralized access control for IPFS. In 2018 IEEE international conference on internet of things (ithings) and IEEE green computing and communications (greencom) and IEEE cyber, physical and social computing (cpscom) and IEEE smart data (smartdata) (pp. 1499–1506). IEEE.

Wang, S., Zhang, Y., & Zhang, Y. (2018). A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access: Practical Innovations, Open Solutions, 6*, 38437–38450. doi:10.1109/ACCESS.2018.2851611

Wilkinson, S., Boshevski, T., Brandoff, J., & Buterin, V. (2018, October). Storj: A decentralized cloud storage network framework (Whitepaper). Retrieved from https://storj.io/storjv3.pdf

Wu, R., Zhang, X., Ahn, G.-J., Sharifi, H., & Xie, H. (2013). Acaas: Access control as a service for iaas cloud. In 2013 international conference on social computing (pp. 423–428). Academic Press.

Younis, Y. A., Kifayat, K., & Merabti, M. (2014). An access control model for cloud computing. *Journal of Information Security and Applications, 19*(1), 45–60. doi:10.1016/j.jisa.2014.04.003

Zaghloul, E., Li, T., & Ren, J. (2018). An attribute-based distributed data sharing scheme. In 2018 IEEE global communications conference (globecom) (pp. 1–6). doi:10.1109/GLOCOM.2018.8647167

Zahoor, E., Asma, Z., & Perrin, O. (2017). A formal approach for the verification of AWS IAM access control policies. In F. D. Paoli, S. Schulte, & E. B. Johnsen (Eds.), *Service-oriented and cloud computing - 6th IFIP WG 2.14 european conference, ESOC 2017, oslo, norway, september 27-29, 2017, proceedings* (Vol. 10465, pp. 59–74). Springer.

Zahoor, E., Bibi, U., & Perrin, O. (2019). Shadowed authorization policies - A disaster waiting to happen? In R. Cheng, N. Mamoulis, Y. Sun, & X. Huang (Eds.), *Web information systems engineering - WISE 2019 - 20th international conference, hong kong, china, november 26-30, 2019, proceedings* (Vol. 11881, pp. 341–355). Springer.

Zahoor, E., Ikram, A., Akhtar, S., & Perrin, O. (2018). Authorization policies specification and consistency management within multi-cloud environments. In N. Gruschka (Ed.), *Secure IT systems - 23rd nordic conference, nordsec 2018, oslo, norway, november 28-30, 2018, proceedings* (Vol. 11252, pp. 272–288). Springer.

Zhang, Y., Kasahara, S., Shen, Y., Jiang, X., & Wan, J. (2018). Smart contract- based access control for the internet of things. *IEEE Internet of Things Journal, 6*(2), 1594–1605. doi:10.1109/JIOT.2018.2847705

Zichichi, M., Ferretti, S., & D’Angelo, G. (2020). On the efficiency of decentralized file storage for personal information management systems. In *2020 IEEE symposium on computers and communications (iscc)* (pp. 1–6). IEEE.

Zichichi, M., Ferretti, S., D’Angelo, G., & Rodriguez-Doncel, V. (2020). Personal data access control through distributed authorization. In *2020 IEEE 19th international symposium on network computing and applications (NCA)* (pp. 1–4). IEEE.

Zyskind, G., Nathan, O., & Pentland, A. S. (2015). Decentralizing privacy: Using blockchain to protect personal data. In *2015 IEEE security and privacy workshops* (pp. 180–184). IEEE.

ENDNOTE

1 https://www.icloud.com/iclouddrive/0CSWHu_j8Ug4ZBJ8RNSfMBliA#JWSR%5FIPFS%5FS3%5FSources