VOL.39 Nº3 2022
https://doi.org/10.2271\IMECTA.10770.en

Use of blockchain in the control and traceability of access to data stored in the cloud

Souza Jonatan[a], Pinto Raquel[b], Bruno Schulze[c]
[a]Instituto Militar de Engenharia (IME), jonatangd.souza@gmail.com
[b]Instituto Militar de Engenharia (IME), raquel@ime.eb.br
[c]Laboratório Nacional de Computação Científica (LNCC), schulze@lncc.br

RESUMO: A computação em nuvem oferece diversos serviços, como armazenamento de dados e máquinas virtuais (VMs). Esses recursos são disponibilizados pela Internet e seu pagamento é dado pelo uso. Embora os serviços de nuvem sejam eficientes, há uma preocupação crescente na segurança e privacidade desses serviços prestados por nuvens computacionais. Entre essas preocupações, pode-se destacar o compartilhamento de dados entre usuários. Estes serviços não possuem um mecanismo de permissão que seja auditável pelo proprietário dos dados. Neste contexto, o blockchain tem se destacado principalmente por sua arquitetura de ledgers distribuídos que permite uma trilha imutável e auditável. Além disso, a arquitetura descentralizada do blockchain elimina a necessidade de confiança em terceiros. Portanto, este artigo apresenta uma arquitetura baseada no uso da tecnologia blockchain como repositório seguro e auditável de registro dos acessos e permissões concedidas aos usuários. Como resultado, este artigo apresenta um estudo de caso validando a arquitetura proposta.

PALAVRAS-CHAVE: Computação em nuvem. Blockchain. Compartilhamento de dados. Segurança. Privacidade.

ABSTRACT: Cloud computing offers a variety of services such as data storage and virtual machines (VMs) . These features are made available over the Internet and your payment is per use. While cloud services are efficient, there is growing concern for the security and privacy of these services provided by cloud computing. Among these concerns, the sharing of data between users can be highlighted. These services do not have a permit system that is auditable. In this context, the blockchain has stood out mainly for its distributed ledger architecture that allows an immutable and auditable trail. In addition, the blockchain's decentralized architecture eliminates the need to rely on third parties. Therefore, this article presents an architecture based on the use of blockchain technology as a secure and auditable repository for recording access and permissions granted to users. As a result, this article presents a case study validating the proposed architecture.

KEYWORDS: Cloud computing. Blockchain. Data sharing. Security. Privacy. Data storage.

1. Introduction

Data sharing between users and services has become increasingly common, especially with the growing adoption of the Internet of Things (IoT - Internet of Things). In IoT systems, several smart devices interact with each other generating data from different contexts. Considering the large amount of data and the reduced storage capacity of personal devices, the use of data storage services has been increasingly common both by ordinary users and by large companies.

This storage migration takes place as the limitations of hardware and infrastructure for communication are mitigated. In this way, cloud services are often used. Among the main advantages, we can highlight: its simplicity, low financial cost and high availability of resources.

Cloud computing can be defined as a type of distributed system, consisting of a set of interconnected and virtualized computers. These resources are dynamically available as unified computing resources, whose services are based on service level agreements. [1] The cloud environment is shared among several users where the demand for hardware and software can be hired or sold at any time. Therefore, the cloud must be able to grow elastically, as the demand for these resources. [2]

Considering the aforementioned advantages, cloud services are widely adopted in different contexts, but it is necessary to consider that it is a third-party service for data storage. Therefore, there is a need for the owner to trust this service to allow or revoke access to third-party users, and not just with regard to storage, since access control and permissions are on the service side.

Therefore, there is a growing concern about the security of these services provided by computing clouds: I) the warranty that data is shared only with users authorized by the owner; II) services, in general, do not have a permission mechanism that is auditable by the data owner.

In this context, blockchain has gained attention mainly for its distributed ledger architecture. The blockchain consists of a chain of blocks linked together using one or more hashes of the previous block, providing an immutable and auditable trail.

This work presents a blockchain-based architecture that provides a secure and auditable environment for recording access and granted permissions, thus transferring the need for trust in sharing data with different users, to an environment where this data is auditable and immutable. Therefore, the owner of the data will be able to audit the permissions given and revoked to their data, in addition to verifying when they were accessed.

This article is organized as follows. Section II introduces the current blockchain landscape and key concepts. Section III addresses some of the major related work. Section IV presents the proposed architecture in detail. Section V addresses a case study of the architecture in an IoT context. Finally, Section VI concludes the article with some discussions and future directions.

2. Blockchain Fundamentals

This section presents a brief description of important concepts and technologies adopted in this research study, related to blockchain.

Blockchain is a decentralized data structure that is replicated and shared among members of a network called peers. Each block contains an ordered set of transactions and a hash that are stamped with a timestamp. Each block also includes the timestamp of the previous block in its hash, forming a linked list of blocks, with each additional timestamp reinforcing the previous ones. [14]

The block data structure is composed of a header and a list of transactions. The header contains metadata about a block and the hash of the previous block. For each block N, the hash of block N-1 is considered. The configuration block that initializes components and serves as the first block in a chain is called the genesis block. Blocks are created by the Ordering Service and validated by other network elements.

Decentralization means that none of the peers alone has the ability to control the processing of all transactions on the network.

Blockchains are also architecturally decentralized in that there is no central point of failure, but everyone must agree on a single state through consensus.

The ledger or ledger contains the current state of a business and works like a transaction journal. The blockchain ledger basically has two attributions: I) present the current value of a set of states; and II) maintain the history of the transactions that determined these states.

One of the main features of blockchain is the immutability of stored data. This is due to the recurrence of hashes, where previous blocks cannot have their content violated and remain valid. Therefore, transaction records are permanent, considering that if changed, the entire subsequent blockchain will be invalid.

Considering the decentralized architecture, achieving consensus in this scenario is a challenge for a blockchain network. Achieving consensus ensures that all network nodes agree on a consistent global state of the blockchain. This is important in blockchain as it ensures that stored data cannot be tampered with unless the attacker gains control of (50% + 1) network nodes, since this is the amount that is needed for validation in most implementations. Also, blocks before the last one when modified its content, make all the blocks in front of it invalid. This feature is given by hash recurrence.

Since the blockchain has responsibility for the distributed ledger and the entire immutable data structure, the smart contract extends this function by including a language for terms of agreement and measurements ensuring that certain conditions are met.

In a minimalist way, smart contracts are scripts that act on the blockchain and have a triggering transaction that is responsible for executing actions. [15] As they reside on the blockchain chain, they have a unique address. When executing a transaction that has a smart contract addressed, it executes independently and automatically in a prescribed way in all the nodes of the network, according to the data that were included in the transaction triggering the smart contract. [16] [17]

There are several blockchain platforms available for implementing solutions. These platforms are basically divided into two large groups: public blockchains and private or permissioned blockchains. [18]

Public blockchains, being open, have more robust and computationally more expensive consensus algorithms. Normally, it is necessary to insert the figure of the miner. Private blockchains are the opposite as they are made up of known nodes. In this way, they have simpler consensus algorithms, making transactions faster and allowing architecture changes in a shorter time.

3. Related Works

Cloud services, in general, offer practicality, scalability of service use, high availability and resource management. Although all these offered benefits are relevant for the decision to migrate from a local solution to a cloud service, some issues related to security and integrity require permanent monitoring of this information. In [3] the authors propose a software architecture that allows the storage of files in cloud services, with guaranteed information privacy, in addition to permanent monitoring of the integrity of the files, based on technologies such as blockchain. In this work, the authors highlight two main obstacles to the adoption of blockchain platforms, which are: high energy consumption and low transaction processing speed, in view of the excessive use of cryptography and consensus algorithms among peers, thus justifying the use of HyperLedger Fabric as a more efficient solution for the proposed scenario.

In [4] an architecture based on blockchain, smart contracts and computational trust technologies that is capable of periodically monitoring the integrity of files stored in the cloud is presented. Among the possible applications for the proposed architecture, the authors highlight the storage of database backups of electronic document management systems. These are generally large files and, due to legal issues, need to be stored for long periods of time.

For the implementation of the blockchain, the authors opted for the Ethereum platform, in view of the availability of documentation, the ease of creating a network to perform tests in a local environment, and also the number of tools available to support the use and execution of the test. The work also presents an analysis of the security of the architecture.

The sharing of information stored in cloud services has also raised concerns. Generally, cloud services do not offer an auditable solution, thus requiring the contractor's trust in the third-party service. In [5], the authors point out that dynamic group data sharing, where users anonymously share their data with other group members using the cloud service, can compromise security. Therefore, they highlight the need to design an efficient and secure system for sharing data in dynamic groups. This paper presents a review of the challenges encountered in efficiently designing dynamic group data sharing. Among the challenges encountered, include user authentication, privacy and security, data confidentiality, integrity, and query cost. They also mention service provider-based issues that include user identity and its traceability, and user revocation.

Security, privacy and data integrity in cloud services has motivated several researchers. In [6], It is highlighted that public verification techniques may allow a user to employ a third-party auditor to verify data integrity on their behalf. However, existing public verification schemes are vulnerable as they allow auditors to fail to carry out checks in time. In this context, the authors propose a mechanism for public verification of the integrity of cloud storage of files resistant to auditors' procrastination, without the use of certificates, bearing in mind that this type of mechanism uses, for the most part, public key infrastructure (PKI) and therefore suffer from certificate management issues. The mechanism called Certificateless Public Verification scheme against Procrastinating Auditors (CPVPA), uses blockchain technology, and aims to require auditors to record each verification performed in a transaction on the blockchain. Since transactions on the blockchain are time sensitive, strategically, the verification can be time stamped after the transaction is recorded on the blockchain, which allows users to verify that auditors have performed their verifications in the prescribed time.

Due to the limited processing capacity of the devices that normally make up an IoT network, devices often use externally controlled third-party services to perform additional required processing, such as a computational cloud for example. IOT SMART CONTRACT [7], is a proposed solution for the decentralized management of data access using blockchain and the data privacy protection offered by the Intel SGX. This solution aims to establish trust between IoT service providers and the users of these services. Through smart contracts, the proposed platform provides data access management where users have privileges to control how their data is shared or used. Furthermore, it is possible to assign data access rules that are applied autonomously by untrusted third-party services on the blockchain network.

The security and privacy of the Internet of Things (IoT) is an imminent challenge due to the massive scale and distributed nature of IoT networks. Blockchain-based approaches offer decentralized security and privacy, but involve excessive power consumption and increased latency, which can be an issue for most resource-constrained IoT devices.

In [8], The security and privacy of the Internet of Things (IoT) is an imminent challenge due to the massive scale and distributed nature of IoT networks. Blockchain-based approaches offer decentralized security and privacy, but involve excessive power consumption and increased latency, which can be an issue for most resource-constrained IoT devices [9], The use of blockchain for security and privacy in a smart cities scenario is introduced.

IoT devices can suffer different types of attacks, mainly public access ecosystems, such as smart meters. Smart meters help energy utilities optimize their profitability by reducing expenses associated with energy theft and technical losses. Consumers, on the other hand, now have access to real-time energy consumption data, which they can use to increase their energy efficiency, reduce their monthly bills and help the utility stabilize the grid during peak periods. Preventing security threats such as data falsification is a challenge. In [13] a solution using the blockchain is presented to avoid security threats in these ecosystems. In this work, the Zero-Knowledge proof approach is still used, a blockchain anonymity improvement technology that mitigates security threats, such as breach of personal information. The work also proposes the use of smart contracts to avoid data tampering and increase the reliability of meters.

Observing the works listed in this section, it is concluded that although all works use blockchain, most use public blockchain. In this article we propose the use of a private blockchain implemented through the HyperLedger Fabric. This solution adds flexibility and simplicity, as public blockchain solutions demand a robust infrastructure with the addition of miners.

4. Architecture For Data Access Control And Traceability

In view of the limitations of cloud services to provide an auditable trail of data access, as well as transparency in permissions and revocation of data access permissions, the architecture illustrated by **figure 1** presents a solution that meets these requirements.

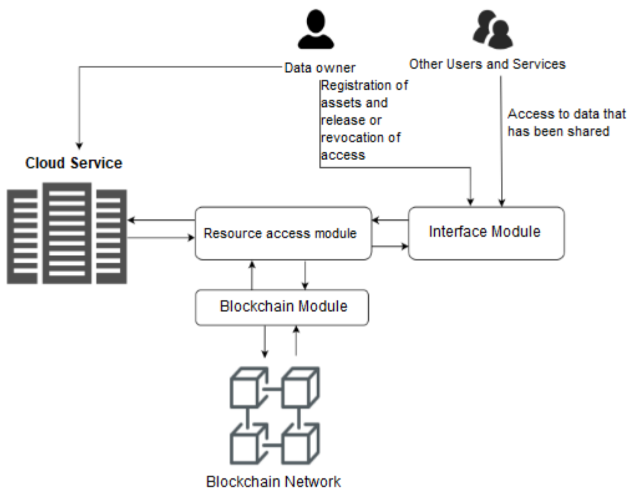

Fig. 1 – Proposed architecture.

In this architecture, owners create accounts, register their assets stored in the cloud and register sharing with third parties through the Interface Module. Users use this module to register and access shared assets.

The Resource Access Module offers APIs used by the Interface Module to access assets in the cloud and register and validate operations performed on the blockchain.

The Blockchain Module, in turn, has a blockchain-specific API and all the smart contracts needed to validate and create transactions.

In the next subsections, these modules are described in more detail.

4.1 Interface Module

Through the Interface Module, the owner creates an account and registers the necessary credentials, so that later access to the assets is possible in the place where they are stored. With the account created, and with the email and credentials verified, the owner of the data can log in, and thus register the assets that he wants to share with third parties. When registering an asset, it must be validated in the Blockchain Module, in order to verify if the asset already exists. In addition, it must be validated by the Resource Access Module, to verify the existence of data associated with the asset in the cloud.

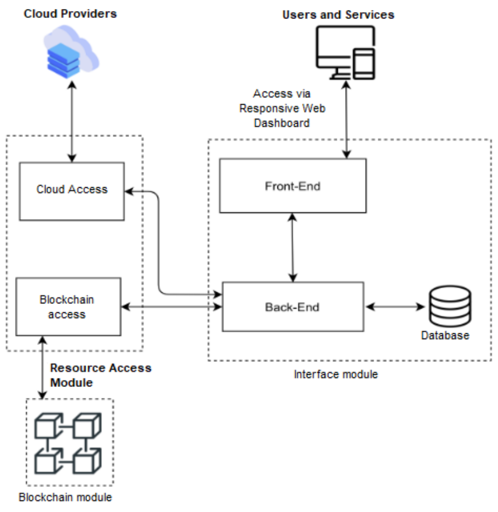

Fig. 2 – Interface Module and Resource Access Module

All registered data needs to be persisted on the blockchain, but there is data that does not require an auditable trail. For this reason, the interface module has a database, which is used to register metadata, additional information, among others.

After registering the assets, the owner can view the list of registered assets, where it is possible to view the history of operations on the asset, as well as grant or revoke access to third parties. Third party users receive an invitation to access the assets. After logging in, a list of shared assets is presented. Thus, the user can request access to assets shared with him. All operations on assets are validated by the blockchain module, which guarantees the recording of transactions, generating an auditable access trail.

In this way, this module has the responsibility of being the gateway for users. **Figure 2** illustrates its main components.

The Front-End is a web portal responsive to different screen sizes, with an intuitive interface, where users perform the operations proposed by the architecture.

The Front-End is a SPA (Single Page Application). This approach allows the development of a more robust and decoupled application from the server.

The Back-End is a REST API able to meet the demands of the Front-End and, in turn, have access to the resource access module and connection to a database. This database, ideally, is unstructured, document-oriented. The choice of this approach is due to the flexibility of the structure that, in addition to making scalability simpler, facilitates insertion and access to data.

Access to the Back-End is only allowed to authorized users via login, where they can only view data or actions performed by them. As an example, a user authorized through login would not be authorized to access details of an asset that does not belong to him, even knowing the identifier of this asset. The only time this type of access is allowed is when the user is on the authorized list. In this context, the information provided about this asset is limited, preserving sensitive information, such as the bucket or cloud service used, for example.

The Back-End must be able to store information about the user, as well as name, email, asset metadata, password hash and information about the cloud provider. In addition, he must be able to access the Resource Access Module via a secret key. Therefore, only the Back-End of the interface module is able to access the resource access module.

By providing the registered credentials, it is possible to map the resources and make them available as assets. When the assets are registered, the Back-End sends a transaction to the blockchain informing the new resource that must be persisted. Once registered, this asset is available to be shared with other users. All requests regarding access to assets are also persisted in the blockchain, through a Back-End request.

4.2 Resource access module

The Resource Access Module has two main objectives: I) Accessing external resources, as the module must be able to access resources in the cloud for availability; and II) Submitting transactions to the blockchain module. **Figure 2** illustrates the details of the resource access module.

For security reasons, the resource access module must only be accessed through the Back-End of the interface module, being in a private network.

Cloud Access is responsible for integrating cloud providers and abstracting the differences into a common interface. The module enables access to data in a bucket, as well as making it available for access by authorized third parties.

The cloud resources are strategically decoupled from the rest of the solution, bearing in mind that a future change of cloud provider or even a local storage solution would have a small and one-off impact on adapting the solution.

Access to the blockchain is responsible for dispatching all requests that must be validated by the blockchain, and works as a second authenticator, where resources are made available only when Access to the blockchain receives positive feedback from the transaction. Otherwise, the initial request will receive the status of unauthorized.

Therefore, Access to the blockchain performs an encapsulation where the HTTP request is prepared in the format expected by the Blockchain Module, being also capable of interpreting the return of the Blockchain Module, and reporting the responses of the peers to the requesting user.

4.3 Blockchain Module

In the proposed architecture, a private blockchain network is used to store all transactions carried out with data stored in the cloud. The choice of using blockchain instead of a distributed database has two main motivations: I) Given its construction nature in subsequent blocks, where its cryptographic content has a hash that points to the next block, the data becomes immutable , invalidating the string if a previous record is changed; and II) Blockchains are decentralized, this characteristic added to the consensus mechanisms, guarantee a high level of security for the solution, since the attacker would need to have in his control the majority (50% + 1) of the network nodes in his control.

The decision to use a private blockchain network is based on the nature of the solution. Both private and public blockchain networks are decentralized solutions, have a consensus mechanism and provide high security for information and transactions. [18] However, in the public blockchain, any entity can participate in the network, while in the private blockchain, there are mechanisms that control the entry of new nodes in the network. In addition, in the public blockchain network, there is an incentive for users who contribute to validations, called cryptocurrency. These users are called miners, as the process of working to encrypt data in exchange for small fractions of cryptocurrencies is called the mining process. In private blockchains there is no such incentive, since the network is usually created for a specific purpose. Finally, as the number of nodes on the private blockchain is smaller, transactions take less time to register as the consensus algorithm converges faster. Unlike public chains, with thousands of participants.

Considering that a private blockchain network does not allow direct access to its peers, it is necessary to create a module that encapsulates the blockchain functionalities and makes them available for applications to use. The Blockchain Module is responsible for making these functionalities available in a secure way, so that other services can access the resources provided by the Blockchain, figure 3 illustrates its operation.

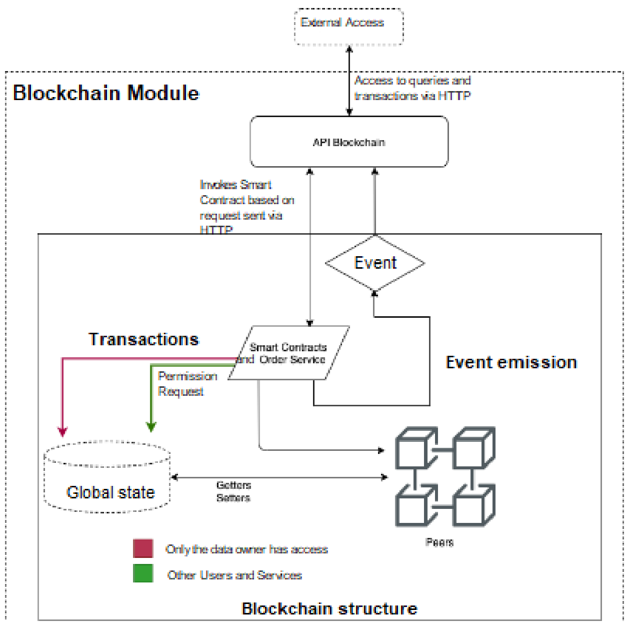

Fig. 3 – Blockchain Module

In order to have an auditable trail, every interaction from the outside world with the blockchain is considered a transaction. Therefore, the module receives all transactions via HTTP, validates the integrity of the information and then the transaction proposal is built.

Each peer has a copy of the smart contracts responsible for ensuring the correct validation of each transaction. In the context of the proposed architecture, the following smart contracts must be implemented:

- **Query asset:** this transaction performs the reading of an asset based on its identifier;
- **Register asset:** this transaction creates an asset in the ledger, with the necessary data for access permission and revocation;
- **Grant access to asset:** transaction in which the owner grants access to a third party;
- **Revoke access to asset:** transaction in which the owner revokes access to a third party;
- **Query by owner:** this transaction returns all assets from the requesting owner;
- **Request access permission to asset:** this transaction is responsible for granting or revoking access to an asset, having a third party as the requester; and
- **Query asset history:** this transaction returns the auditable trail of all operations performed on an asset;

All mentioned transactions carry out the following validations: I) if the identifier of the requested asset exists in the ledger; and II) if the requester can perform the requested operation. For the permission request, a new update is generated in the asset, recording the requested access details.

When registering the user in the system, the Back-End Module activates the Blockchain Module, which, in turn, is responsible for creating the user and generating all the cryptographic material necessary for this user to have access permission to the data. . At this time, all users are created with the same profile, being able to submit new transactions to the blockchain. However, the user can only submit transactions referring to assets that they own. Therefore, any user is allowed to register new assets, as well as submit transactions to change them. Through the validation of smart contracts, a user when trying to submit a transaction related to a third-party asset, this transaction will have the status of failure, and a record of this attempt will be available in the auditable trail provided by the blockchain.

Endorsing peers receive transaction proposal inputs as arguments to invoke the smart contract. Each peer appends a block to the channel chain, and for each valid transaction, write sets are committed to the current state database. An event is emitted by each peer to notify the client application that the transaction has been immutably attached to the chain, as well as a notification whether the transaction has committed or invalidated.

Having the appropriate responses, the Blockchain Module returns the response to the requesting application. This flow allows all accesses to be transformed into transactions, forming the immutable trail for future audits.

4. Study Case

PTo apply the solution, a case study was developed in an IoT context, where the owner of the data generated by the devices is interested in sharing them with other users and services. However, access to this data must be controlled and the system in question must provide an auditable trail of users who have accessed the data.

For implementation, a responsive web application was created, providing an intuitive solution for users. This web application accesses an API called Gateway Service API, responsible for accessing the blockchain infrastructure and cloud infrastructure. An environment configured with Hyper Ledger Fabric was also available, where smart contracts were implemented. To make smart contract functionalities available outside the HyperLedger context, the smart contract service API was created. This API is accessed only by the Gateway Service API, **figure 4** illustrates how the implementation is organized.

The Gateway Service API was integrated with the cloud provider Google Cloud, through the provision of credentials. The directories with the data you want to share are registered as assets in the blockchain API. Once this registration procedure has been carried out, the owner can share his assets with other users.

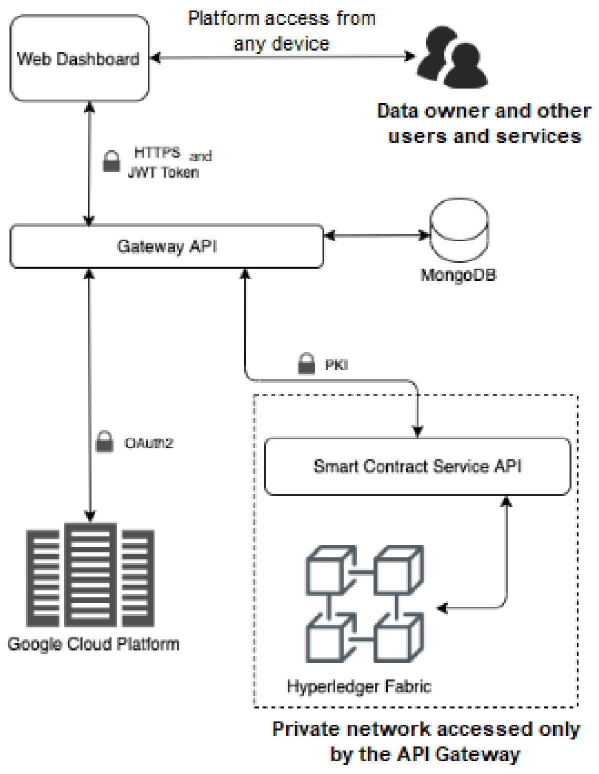
Web Dashboard
Platform access from any device
Data owner and other users and services
HTTPS and JWT Token
Gateway API
MongoDB
PKI
OAuth2
Smart Contract Service API
Google Cloud Platform
Hyperledger Fabric
Private network accessed only by the API Gateway

Fig. 4 - Implementation of case study

The user who will receive authorization to access the shared data receives an invitation via email to register on the platform. Once registered through the web dashboard, the user sees the list of assets that have been shared with him. Still through the web dashboard, the user requests access to the desired asset. At that point, the Gateway Service API submits a transaction to the blockchain API. The transaction is validated by the peers and entered into the ledger. Once the return of the transaction is validated, the Gateway Service API accesses the cloud provider Google Cloud, obtains the data related to the asset and provides a page for download. In the next subsections the implementation components are described in more detail.

5.1 Web Dashboard

The Web Dashboard implements the architecture's Interface Module. It is responsive, allowing its usability on any device that has a browser, regardless of screen size. The dashboard was developed using the Angular Framework, which allows the creation of Single Page Applications (SPA). For styling and usability, the Ionic Framework was used, which is a framework made up of a series of mobile-first components, which allow the creation of pages with an approach similar to that used in native mobile applications.

The Web Dashboard offers functionality to users, and has two profiles: data owners and guest users.

For owners, it is possible to register devices in the form of assets. The assets are available in a list, where you can access the details of the selected asset. On the detail page, the data owner is able to grant or revoke access to a third party, edit information and access the history of performed operations. These operations include: any attempt to gain access by the owner or a third party, edits to information or permissions. This auditable trail is available in the form of a timeline containing the date and time the operation took place.

Users who received the share, only view the assets that were shared, not being able to perform any editing operations. When selecting an asset, the user receives the OTP (One Time Password), and thus is able to download the content. It is important to emphasize that each access is unique, and when performed, it generates a new transaction on the blockchain, thus creating an auditable trail of the accesses granted.

5.2 Gateway Service API

The Resource Access Module was implemented through the Gateway Service API. This API provides access over the internet to the cloud service, where IoT System data is stored, and to the blockchain. The API has three main responsibilities: I) Establishing the connection with the smart contract service API; II) Make available the data stored in the cloud provider; and III) Expose login-protected end-points to accept HTTP requests and perform corresponding operations.

Bearing in mind that blockchain operations are slower, especially when the number of peers increases, it is not good practice to use this structure to store all data related to the operation of applications. For this reason, it is common for solutions to use conventional databases to support the blockchain for less critical operations or operations that do not require the features provided by the blockchain.

Therefore, the API gateway, in addition to the blockchain structure for storage, also relies on MongoDB, which is a non-relational database, in SQL and document-oriented. The decision to use a document-oriented database is due to the versatility that this structure allows, considering that the solution may require the addition of new properties not only in the metadata, but also in the addition of new collections of documents. The purpose of this database is to store data that you are not interested in being auditable or creating an immutable trail. Therefore, the login data of each user is stored in MongoDB, in addition to the metadata of users or assets.

5.3 Smart Contract Service API

Among private blockchain platforms, Hyperledger Fabric [19] features a blockchain architecture that provides flexibility, scalability, and confidentiality. Another feature of Fabric is that its architecture segregates the flow of transactions into three stages, whose executions can be carried out by different entities: execution of the transaction and verification of its correctness, ordering of transactions using a consensus algorithm and validation of transactions from the network consensus. [20] In this way, HyperLedger Fabric was used as a blockchain implementation.

The Smart Contract Service API implements the Blockchain Module. The blockchain infrastructure created in the context of this work is private, and moreover, it is not able to communicate through the HTTP protocol. Therefore, it is necessary to create a REST API responsible for providing the functionalities of smart contracts. The Smart Contract Service API is a REST API responsible for making blockchain transactions available via the HTTP protocol.

Through the HyperLedger Fabric SDK, one of the available APIs is used to generate a transaction proposal. The proposal is a request that will trigger a smart contract with the input parameters, with the intention of reading and/or updating the ledger.

The HyperLedger SDK, in this context, works as a wrapper to package the transaction proposal in the right format and architecture (buffer protocol over gRPC) and also deliver the user's cryptographic credentials to produce a unique signature for this transaction proposal.

Although the API makes blockchain functionalities available, for security reasons it must be in a private network, thus not allowing external access, only allowing the API gateway to be authorized to consume the resources.

The API was developed with NodeJS, using the ExpressJS Routing Framework, and uses HTTP verbs to perform operations on the blockchain. This integration is performed through the SDK provided by the HyperLedger Fabric framework.

5.3 Experiments Performed

After populating the system with users and assets, some experiments were carried out to demonstrate the correct functioning of the implementation.

In the first experiment, a user gave access to his assets to third-party users. These users then verified the addition of these assets to the list of assets shared with them and were able to access the corresponding data on Google Cloud. Later, the owning user revoked access to an asset to a third party user. This third party is no longer able to access the resource, given that none of the peers validates this access request transaction. The API returns the request as 403 Forbidden, even before establishing a connection with the cloud provider. This HTTP 403 Forbidden error status response code indicates that the server understood the request but refuses to authorize it.

The user who owns an asset can view all operations performed on the asset through an auditable trail stored on the blockchain.

The second experiment aimed to demonstrate the resilience of the implementation. The MongoDB bank, responsible for authenticating the application used by users, was corrupted, and no data in the context of the blockchain was changed. Therefore, after restoring the authentication service and MongoDB, each user must register again in the system with the same login and password. Then, the application returns to its previous state, showing the asset data, as well as their respective histories that are restored from the blockchain. This demonstrates that the solution is robust, considering that the data persisted in the blockchain is resilient because it is a distributed and not centralized environment.

6. Conclusion

In this article, an architecture was proposed that controls access to data stored through third-party services, offering traceability of these accesses to data owners. The architecture proposes the use of blockchain to record information about the accesses performed, in view of its immutability characteristic.

The case study allowed putting the proposed architecture into practice in a scenario with real integration with a Cloud provider and a blockchain structure. The implementation allowed demonstrating, in a simplified way, how users are able to register their assets, allow or revoke access. In addition, the user is allowed to access the history of operations performed on the assets he owns, demonstrating the proposed auditability.

Furthermore, the implementation also demonstrated the architecture's fault tolerance and recoverability.

Finally, the literature review revealed that most of the available works use Ethereum as an implementation of public blockchain networks. This choice is due to the amount of material available for development, when compared to HyperLedger Fabric, whose version 2.0 is recent. However, the use of public blockchain involves additional costs, due to the insertion of the figure of the miner, resulting in several challenges and limitations. On the other hand, the HyperLedger Fabric allows the creation of blockchains according to the availability of hardware, since the network administrator controls the peers that make up the blockchain network.

As an additional result, this work demonstrates that the use of Hyper Ledger Fabric is an interesting alternative, as in addition to allowing the use of a fully known network, it also provides data security and network resilience. With this, the use of this type of approach can be encouraged to further assist researchers and developers in the field of distributed and cloud computing, with the aim of mitigating the access of attackers to systems through the network, in view of the difficulty introduced by the blockchain to carry out invasions.

References

[1] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 25(6), 599-616.

[2] Bachiega, N. G. (2014). Algoritmo de escalonamento de instância de máquina virtual na computação em nuvem.

[3] Pinheiro, A., Canedo, E. D., Albuquerque, R. D. O., & de Sousa Júnior, R. T. (2021). Validation of Architecture Effectiveness for the Continuous Monitoring of File Integrity Stored in the Cloud Using Blockchain and Smart Contracts. Sensors, 21(13), 4440.

[4] Pinheiro, A., Canedo, E. D., De Sousa, R. T., & Albuquerque, R. D. O. (2020). Monitoring File Integrity Using Blockchain and Smart Contracts. IEEE Access, 8, 198548-198579.

[5] Kotha, S. K., Rani, M. S., Subedi, B., Chunduru, A., Karrothu, A., Neupane, B., & Sathishkumar, V. E. (2021). A comprehensive review on secure data sharing in cloud environment. Wireless Personal Communications, 1-28.

[6] Zhang, Y., Xu, C., Lin, X., & Shen, X. S. (2019). Blockchain-based public integrity verification for cloud storage against procrastinating auditors. IEEE Transactions on Cloud Computing.

[7] Ayoade, G., Karande, V., Khan, L., & Hamlen, K. (2018, July). Decentralized IoT data management using blockchain and trusted execution environment. In 2018 IEEE International Conference on Information Reuse and Integration (IRI) (pp. 15-22). IEEE.

[8] Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., & Njilla, L. (2017, May). Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (pp. 468-477). IEEE.

[9] Biswas, K., & Muthukkumarasamy, V. (2016, December). Securing smart cities using blockchain technology. In 2016 IEEE 18th international conference on high performance computing and communications; IEEE 14th international conference on smart city; IEEE 2nd international conference on data science and systems (HPCC/SmartCity/DSS) (pp. 1392-1393). IEEE.

[10] Dorri, A., Kanhere, S. S., & Jurdak, R. (2016). Blockchain in internet of things: challenges and solutions. arXiv preprint arXiv:1608.05187.

[11] Dorri, A., Kanhere, S. S., Jurdak, R., & Gauravaram, P. (2017, March). Blockchain for IoT security and privacy: The case study of a smart home. In 2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops) (pp. 618-623). IEEE.

[12] Huh, S., Cho, S., & Kim, S. (2017, February). Managing IoT devices using blockchain platform. In 2017 19th international conference on advanced communication technology (ICACT) (pp. 464-467). IEEE.

VOL.39 Nº3 2022
https://doi.org/10.2271\IMECTA.10770.en

[13] Lee, C. H., & Kim, K. H. (2018, January). Implementation of IoT system using block chain with authentication and data protection. In 2018 International Conference on Information Networking (ICOIN) (pp. 936-940).

[14] IEEE.Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 21260.

[15] Cong, L. W., & He, Z. (2019). Blockchain disruption and smart contracts. The Review of Financial Studies, 32(5), 1754-1797.

[16] Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. Ieee Access, 4, 2292-2303.

[17] Alharby, M., & Van Moorsel, A. (2017). Blockchain-based smart contracts: A systematic mapping study. arXiv preprint arXiv:1710.06372.

[18] Wüst, K., & Gervais, A. (2018, June). Do you need a blockchain?. In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT) (pp. 45-54). IEEE.

[19] Sajana, P., Sindhu, M., & Sethumadhavan, M. (2018). On blockchain applications: HyperLedger fabric and ethereum. International Journal of Pure and Applied Mathematics, 118(18), 2965-2970.

[20] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., ... & Yellick, J. (2018, April). HyperLedger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the thirteenth EuroSys conference (pp. 1-15).