AutAvailChain: Automatic and Secure Data Availability through Blockchain

Gustavo F. Camilo, Gabriel Antonio F. Rebello,
Lucas Airam C. de Souza, Otto Carlos M. B. Duarte

Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ

***Abstract*—The trust centralization in current data sharing systems restricts the owner's control over their data. Furthermore, the owner's intervention to authorize his/hers data access for each request makes frequent access to popular data tiresome. In this paper, we propose AutAvailChain, an architecture based on software defined networking (SDN) and blockchain to provide secure, automatic, and distributed sharing of IoT data. We develop a prototype using the Hyperledger Fabric platform to implement the blockchain and a smart contract. The results show a quick, secure, and excellent performance of dozens of transactions per second.**

I. Introduction

Classic data storage and availability solutions delegate to centralized authorities, such as governments and companies, the tasks of sharing, controlling, and ensuring proprietary data privacy. These authorities charge high fees and impose terms that compromise data privacy to provide storage and sharing services. The personal data sharing with centralized authorities implies a single point of failure, the loss of control, and the owner's data traceability. Blockchain technology provides the required properties to ensure data sharing security in a distributed and auditable way to maintain owner's data control.

An approach to access authorization to private or sensitive data is to distribute certificates for cryptographic access authentication. Thus, the data owner is responsible for issuing a certificate authorizing the access of another entity that wants or needs to use data to his private data. Nevertheless, in a scenario of tens of thousands of data users requiring access, it is costly to issue certificates manually [1]. Therefore, there is a need for automating data sharing and updating access permissions to stored data. In this scenario, without mutual trust, smart contracts provide the automation needed to receive a data access request, verify that the data consumer meets the conditions imposed by a data owner user, and authorize data access permissions. Software defined networking (SDN) offers the required flexibility due to its network programming capacity for access control enforcement to stored data [2].

This paper proposes, develops, and evaluates an architecture for secure and automatic sharing of IoT data using blockchain and software defined networking in a multi-domain scenario. The proposal guarantees the transparency of data access rights, providing the user with traceability over the data itself. Therefore, we leverage the property of integrity and auditability of the blockchain, which stores the access permissions of each user to IoT data. To prevent personal data leak to the server and third parties, we assume encrypted data storage. The device owner can commercialize its data with data scientists interested in training and creating models based on a real device. The paper presents a blockchain marketplace proposing three types of transactions that allow users to advertise and acquire data by issuing transactions. The proposed transactions guarantee the effective and automatic marketing of IoT data. Thus, the architecture does not require the user acknowledgment of each request. The implementation scenario considers large data centers interconnected through SDN technology. An SDN controller implements data access control. We implement a prototype of the proposed architecture using the open-source platform Hyperledger Fabric [3] for the blockchain, the Ryu SDN controller, and the Mininet network emulator. A smart contract implements the transaction logic proposed in the paper. Results show transaction throughput performance comparable to nationwide e-commerce by controlling access to IoT data securely, automatically, and quickly.

We organize the rest of this paper as follows. Section II discusses related work. Section III presents the attacker model considered in this paper for each entity of the proposed architecture. Section IV details the proposed architecture and presents the proposed transaction types for automatic and effective data sharing. Section V presents the development of the prototype and evaluates the performance of the proposal. The results of transaction throughput and access time are measured and analyzed to evaluate the proposal. Finally, Section VI concludes the paper, presenting the developed work and the directions for future work.

II. Related Work

Several works propose to use blockchain technology to secure data updates, virtual-network slicing [4], virtual-network orchestration [5], [6], and resource access.

Ouaddah *et al.* propose the use of smart contracts for access-policies implementation in code format and implementation of authorization tokens, which are digital signatures of the owner, allowing access to a resource [7]. In their proposal, each user receives a single-use token. The drawback of tokens is a long time to access due to the need to consult the owner with each new access. Pinno *et al.* propose ControlChain, an architecture based on four blockchains for authorizing IoT device access [8]. This architecture requires a large blockchain-storage availability and can generate high costs on platforms that charge for running smart contracts, such as Ethereum. Hu *et al.* propose a data-sharing tool focused on anonymity [9], that uses the blockchain as a checkpoint. Their system robustly guarantees privacy, but requires exaggerated latency to share data. Shafagh *et al.* propose a data-sharing system that separates the storage system into a control plane and a data plane [10]. The control plane contains the blockchain, responsible for storing access permissions, and manages and distributes cryptographic keys. The data plane consists of a distributed storage system, such as the distributed hash tables (DHT). The work, however, is not implemented and does not automate access to users.

Pailisse *et al.* propose an architecture for access control between domains using blockchain [1]. In the proposed architecture, the authors use the blockchain to store the permissions that users have on a resource and use the locator/ID separation protocol (LISP) to enforce access control to the requested resource. The access policies are updated in the blockchain and stored in the LISP control plane, and then, routers can verify access permissions. However, the proposal does not include a reward for the owners for making the data available. The control and commercialization of data by the owners are desirable properties for the Internet of things. Besides, owners are responsible for updating access policies, which is not a scalable solution for IoT systems.

Truong *et al.* propose Sash, a framework for sharing IoT data using blockchain, in which data owners can sell their data [11]. The properties of immutability and transparency of the blockchain provide the auditability of access policies. In Sash, smart contracts evaluate access control requests to off-chain encrypted data. The authors do not define how to distribute cryptographic keys to decrypt the acquired data, which may be a service established by the owner or delegated to a key authority. The use of a cryptographic key distributor authority presents problems, such as centralizing the network and relying on authority, weakening the decentralization property, which is one of the main properties of the blockchain. The purpose of this paper, in turn, is a distributed system between domains in which SDN controllers enforce access control automatically.

Unlike the papers above, this paper proposes a simple and efficient system that is secure, distributed, and automatic for the secure commercialization of IoT data between domains using software defined networks. The system simplicity and efficiency is due to the a single blockchain implementation and the fast data access time and throughput that the system presents, reaching 65 transactions/s.

III. Attacker Model

This work uses the "honest, but curious" model to specify attacks on the cloud that stores the owner's data, which is indicated in the literature to model cloud servers [12], because it captures the most common data-leakage attacks on servers. In this model, the entity compromised by an attacker continues to follow the protocol of the system honestly. In this scenario, the attacker's interest is to obtain or leak sensitive information from the organization, instead of compromising the data integrity or entity behavior. We consider intrusions or internal attacks by employees whose main objective is to remain undiscovered. Therefore, the attackers rarely alter the behavior of the entity. We assume that the stored data is encrypted using a symmetric data key. Thus, the data content is inaccessible to the server, and possible leaks cause less significant damage.

For the organization behavior in the blockchain and for the data owner, we consider the Dolev-Yao attacker model, which predicts the most powerful attackers in an insecure scenario [13]. The attacker can read, send, and discard a transaction addressed to the blockchain or any packet on the network. The attacker can passively connect to the network and capture message exchanges or actively inject, reproduce, filter, and exchange information. Blockchain attacks aim to prevent a legitimate transaction or block from being added to the blockchain. The fault-tolerant consensus protocol mitigates this type of attack, requiring the collusion of many participants to control the network to affect the consensus protocol. Otherwise, the issuer of the transaction/block can check its presence in the blockchain. Attacks that require transaction corruption or tampering are impossible when all transactions include their corresponding signed hash.

Attacks on sellers or buyers consist of trying to get private and sensible data information or impersonating the target. Impersonation attacks are not possible because the issuers sign all transactions sent to the blockchain. The encryption of confidential information mitigates attacks that seek to obtain sensible data, in which the attacker must obtain the private key of the victim. Moreover, the proposed architecture allows the audit of all past transactions. Therefore, if an attacker attempts to modify the blockchain using stolen key pairs, the attempt is logged. Upon discovery of an incident, the attacker can easily replace stolen key pairs, restoring security and preventing further damage.

Network Attacks represent the attempt to isolate a single target, thus preventing sellers and buyers from issuing transactions or from the blockchain controller reading content. The mitigation of this attack category, which includes classic network attacks, establishes redundant paths between the blockchain and organizations, customers, and owners. This work assumes a redundant public network, such as the Internet, which interconnects all participants.

IV. The Proposed System

We propose a simple, secure, efficient, and distributed system for the automated commercialization of IoT-personal data. The system is robust against denial of service attacks because even if an attacker compromises one node in the network, other nodes continue to provide the same service. We also achieve low latency to access the data in the storage server by automating access control and thus removing the need for the seller's intervention on each access request.

The objective of the proposed system is to register data access requests and authorize access automatically if the user

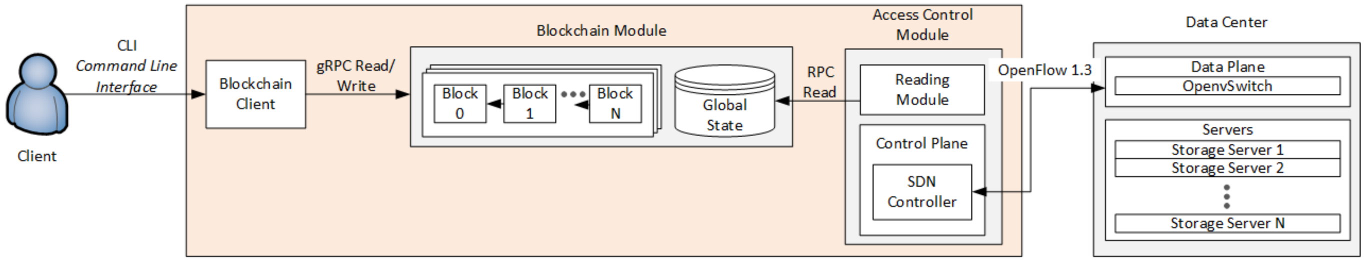


Fig. 1: The proposed system architecture. The buyer (seller) announces its data (acquires data) by communicating with the blockchain. The Access Control Module reads the data purchase transactions and changes the buyer's access permissions to the resource.

meets the established requirements. We consider a multi-domain scenario composed of different storage companies, henceforth called organizations, which interconnect through the Internet and use the software defined networking (SDN) technology. Each organization controls a large data center through an SDN controller. The system uses the blockchain auditability property to transparently register data access permissions in the form of transactions. Smart contracts automate data commercialization by associating each purchase transaction with an advertisement transaction without the need for seller verification. Then, because the transactions are associated, the SDN controller automatically reads its users' advertisement transactions on the blockchain and authorizes access whenever a buyer completes a purchase transaction. We assume the seller encrypts the data using a symmetric key before uploading it to a storage server.

The system architecture is divided into two modules, as shown in Figure 1: i) the blockchain module, which stores a record of access permissions of a user to a resource, and ii) the access control module, which controls access to data by reading transactions from the blockchain.

The Blockchain Module records data advertisements and purchases in the system. It contains the blockchain, which records an immutable history of all transactions issued in the system, and the global state. The global state is a mutable database that indexes transactions in a hash table for fast searches. Data sellers or buyers use the blockchain client command line interface (CLI) to send or obtain information about the blockchain. The general-purpose remote procedure calls (gRPC) establishes the communication between the blockchain client and the blockchain module.

The Access Control Module manages access permissions to a resource and contains the control plane and the reading module. The reading module reads transactions in the blockchain through remote procedure calls (RPC) and provides the acquired information to the control plane. The control plane updates access permissions and allows users to connect to the purchased data in a storage server. The controller stores the users' access permissions for a specific storage server based on the source and destination IP addresses registered in a transaction. If an unauthorized user attempts to connect to the server, the controller verifies the blockchain through the reading module. If the access permissions are out of date, the controller updates the local version of the access permissions by processing new transactions. If the version is up to date or if the user does not have permissions after the update, the controller discards new packages received from the user.

The blockchain in the proposed architecture acts as a data marketplace, in which owners advertise, and buyers query the blockchain looking for IoT data. We define two types of users: sellers and buyers. Sellers are data owners who advertise their data expecting a financial reward. Interested buyers query the blockchain to acquire data. Despite defining two types of users, a participant can assume both roles, either by advertising their data or by purchasing data from other owners. The use of smart contracts allows the implementation of a token-based system that works as an asset exchange. Organizations can acquire data by paying the required number of tokens and agreeing to an off-chain equivalent payment [11].

We propose three types of transactions for securing data commercialization: i) advertisement transactions, ii) purchase transactions, and iii) response transactions. Figure 2 shows the UML sequence that describes the commercialization process.

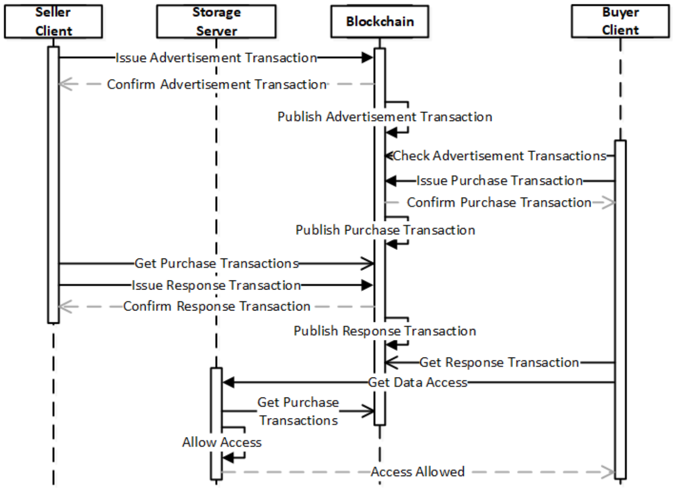

Seller Client
Storage Server
Blockchain
Buyer Client
Issue Advertisement Transaction
Confirm Advertisement Transaction
Publish Advertisement Transaction
Check Advertisement Transactions
Issue Purchase Transaction
Confirm Purchase Transaction
Publish Purchase Transaction
Get Purchase Transactions
Issue Response Transaction
Confirm Response Transaction
Publish Response Transaction
Get Response Transaction
Get Data Access
Get Purchase Transactions
Allow Access
Access Allowed

Fig. 2: A sequence diagram that represents a data sale between two users. By the end of the process, the blockchain contains the data sale transactions and the buyer can access the data.

An owner who is interested in selling their data issues advertisement transactions. The owner, henceforth referred to as the seller, first uploads the data to a storage server capable of storing and processing large amounts of data [11]. Then, the seller issues a signed advertisement transaction using asymmetric encryption to ensure transaction authenticity and integrity. The advertisement transaction must contain the data price and a brief description of the data type. We define an advertisement transaction TX_{ad} as:

$$TX_{ad} = [TX_{ID_{ad}} | Sig_{ow} | IP_{sp} | Pr | O_{ow} | DTD], \quad (1)$$

where $TX_{ID_{ad}}$ is the advertisement transaction identifier, Sig_{ow} is the seller's signature, IP_{sp} is the IP address of the storage server, Pr is the data price, O_{ow} is the organization that stores the data and DTD the data type description.

Users who are interested in acquiring advertised data issue purchase transactions. Such users, henceforth referred to as buyers, look for data in the blockchain by querying advertisement transactions. The system allows specific queries, such as queries by data type and queries for all advertisement transactions. In a purchase transaction, the buyer must reference the identifier of the corresponding advertisement transaction and inform an IP address for the SDN controller to allow access to the data. The purchase transaction must also include the amount to be paid for the data. If the amount offered is less than the data price in the corresponding advertisement transaction or if the buyer does not have enough funds to complete the purchase, the transaction is not valid. We define a purchase transaction TX_{pur} as:

$$TX_{pur} = [TX_{ID_{pur}} | TX_{ID_{ad}} | Sig_{b} | IP_{b} | Pay | O_{src}], \quad (2)$$

where $TX_{ID_{pur}}$ is the purchase transaction identifier, $TX_{ID_{ad}}$ is the transaction identifier for the corresponding advertisement transaction, Sig_b is the buyer's digital signature, IP_b is the buyer's IP address, Pay is the amount to be paid for the data and O_{src} is the buyer's organization.

Sellers automatically issue response transactions after one of its advertisement transactions receives a purchase transaction. The response transaction contains the symmetric data key that decrypts the data. The data key is encrypted using the buyer's public key to ensure only the buyer has access to to the decrypted data. This prevents the storage organization from accessing, leaking or sharing personal data with third parties. We define the TX_{res} response transaction as:

$$TX_{res} = [TX_{ID_{res}} | TX_{ID_{b}} | OK_{Enc\{PK_b\}}], \quad (3)$$

where $TX_{ID_{res}}$ is the response transaction identifier, TX_{ID_b} is the identifier of the corresponding purchase transaction, and $OK_{Enc\{PK_b\}}$ is the symmetric data key encrypted with the buyer's public key.

Transactions go through the blockchain module for execution before being added to a block. The smart contract execution defines whether the transaction is valid or not by verifying if: i) the amount to be paid Pay is greater than or equal to the price Pr of the data requested by the seller, and ii) the organization has enough funds to complete the purchase. Our proposal validates the transactions that meet the previous requirements and marks the other transactions as invalid. If the transaction is valid, the system subtracts the token amount paid from O_{src} and adds it to O_{ow}.

V. Prototype Development and Results

We develop a prototype[1] using the open-source Hyperledger Fabric 2.0 platform [3] for the blockchain, the Mininet network emulator and the controller Ryu[2]. Hyperledger Fabric is a platform for the implementation of permissioned blockchain for organizations. The Hyperledger platform is available for free, easily programmable, and without any currency or associated cost. The organizational aspect of Hyperledger meets our multi-domain proposal scenario, in which companies commercialize the data. An Intel i7-2600 CPU 3.40 GHz computer with 32 GB RAM and 8 processing cores executes the blockchain network nodes as Docker containers. All experiments are presented with an average value and a confidence interval of 95%. We assume 100 transactions per block, as used in previous work on the performance evaluation of the Hyperledger Fabric [14] platform. The nodes in the Fabric architecture represent the entities that take part in the blockchain. The Hyperledger Fabric architecture features three types of nodes: clients, peers, and orderers. Clients represent users and issue transactions that need to be executed by endorser peers, who are responsible for verifying the transaction validity. If the transaction is valid, the client receives the signed transaction by the endorsing peers and sends it to ordering nodes. The ordering nodes execute a consensus protocol and order in a block the transactions by their timestamp. Hyperledger Fabric uses a special type of node called an anchor peer to advertise ordered blocks. We configure the prototype used in the experiments described below using five ordering nodes and the Raft consensus protocol. Figure 3 shows the architecture of the Hyperledger Fabric with the scenario used.

A smart contract written as a self-executing code in Go runs in all peers, eliminating a centralized trust entity and implementing the transaction logic[3] described in the previous section, in addition to a system of tokens and organization accounts. These tokens act as currencies that organizations can use to sell and buy data. The real value of these tokens can be established off-chain between organizations. We implement a pending transaction queue, which stores strings in JSON format containing the IP address of the buyer IP_b, the IP address where the data is stored IP_{sp}, and the identifier of the purchase transaction $TX_{ID_{pur}}$. For every TX_{pur} received, the contract updates the queue, adding a pending transaction to be processed by the software-defined network controller. The pending transaction queue implementation reduces significantly the required reading time to obtain information about

[1] Available at https://github.com/GTA-UFRJ-team/blockchain-marketplace

[2] Available at https://osrg.github.io/ryu/

[3] The response transaction TX_{res} will be implemented in future works.

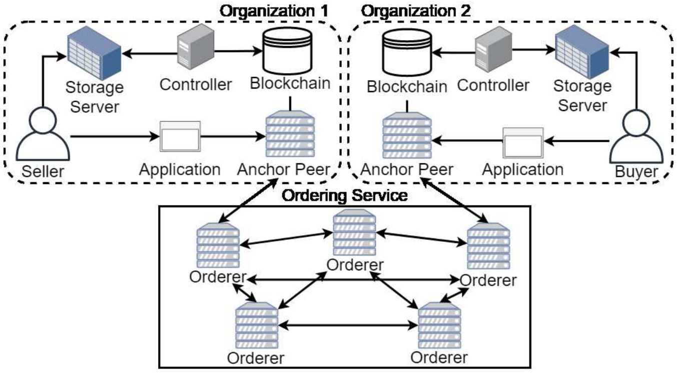


Fig. 3: Permissioned blockchain architecture in the Hyperledger Fabric with the controller and server of the proposed scenario. Users, sellers, and buyers from each organization, use applications to issue transactions that later are ordered in a block by the ordering service using a consensus protocol.

access permissions updates, by avoiding the reading of the full blockchain. The controller, upon receiving a packet from an unknown IP address, calls a function of the reading module to collect the strings stored in the queue and update access permissions. The reading module, written in Python 2.7, reads from the pending transaction queue, gets the buyer's and seller's IP address, and sends the addresses to the controller. The controller stores user's access permissions in a key-value structure. The key is the storage-server IP address and the value is a list, which stores the IP addresses that can access the storage server corresponding to the key. To authorize the access, the controller checks whether the source IP of the packet is in the list corresponding to the destination IP.

The first experiment consists of increasing the number of clients in a single organization and measuring the transaction rate. This experiment simulates a single organization dominating the majority commercialization operations of the data in the blockchain. Two organizations with two peers each compose the blockchain network. We check the transaction throughput while increasing the number of clients who issue 50, 500, and 5000 transactions to the endorsing peers concurrently. The transaction rate corresponds to the ratio between the total number of transactions issued and the required time for all clients to issue all transactions. Figure 4a shows that the transaction rate initially grows until it stabilizes with the increase in the number of clients. This stabilization occurs because a low number of clients issuing transactions restricts the rate in the transaction rate that these clients issue. The transaction rate stabilizes around 65 transactions per second. This result proves that the proposed service fits well for a Brazilian e-commerce national level since MercadoPago has an average throughput of 30 transactions per second[4]. We implement the same environment distributed in three computers to simulate a real scenario. An Intel i7-8700 CPU 3.40 GHz with 32 GB RAM hosts Organization 1. We deploy Organization 2 in an Intel i7-7700 CPU 3.60 GHz with 64 GB RAM. An Intel i7-2600 CPU 3.40 GHz with 64 GB RAM hosts the ordering service. We increase the number of clients issuing 50 and 500 transactions in Organization 1, which is the only organization issuing transactions, and measure the transaction rate. Figure 4b depicts the result with an increasing number of clients running in three computers. The transaction rate peaks around 162 transactions per second, more than twice the value in a single computer. The growth in the node processing power explains the higher transaction rate in the distributed scenario compared to a single computer.

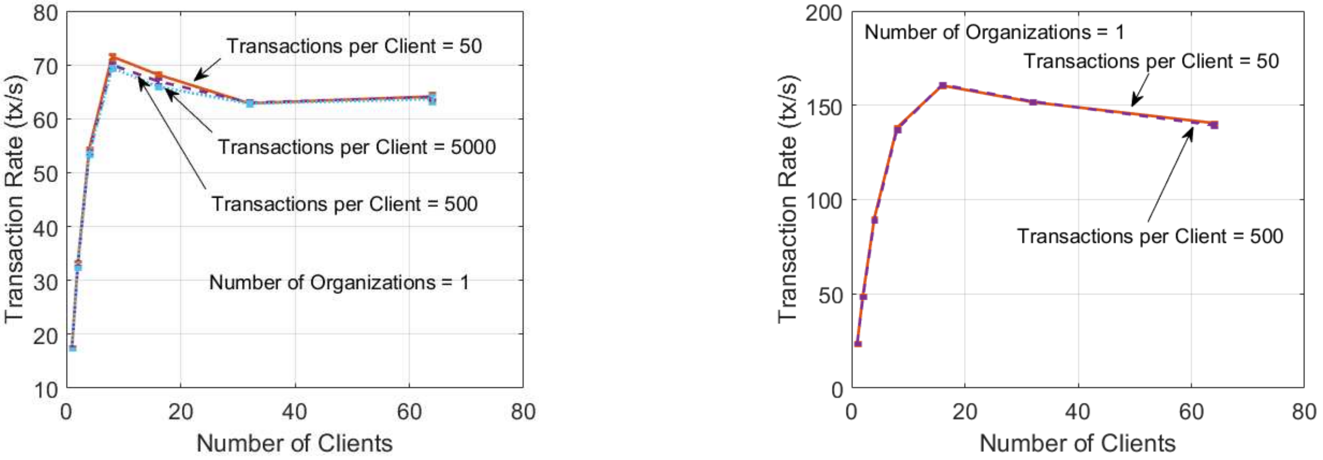
The second experiment implements a distributed environment with 6 computers. We deploy Hyperledger Fabric containers in multiple hosts to simulate a real-world scenario with multiple organizations. An Intel i7-2600 CPU 3.40 GHz with 32 GB RAM hosts the ordering service. We deploy Organization 1 in an Intel i7-8700 CPU 3.20 GHz with 32 GB RAM and 6 processing cores. An Intel i7-7700 CPU 3.60 GHz with 64 GB RAM hosts Organization 2. We add three Intel Xeon E5-2609v4 CPU 1.70 GHz with 8 GB RAM in the previous scenario as Organizations 3, 4 and 5. Figure 5 illustrate the transaction rate for four different number of clients equally distributed in the organizations. The throughput peaks around 243 transactions per second, almost twice the single machine with all clients throughput. The results shows the transaction rate increases for a scenario that the organizations issue transactions in the same proportion, twice the results of a single organization issuing all transactions. The result proves that, even when the number of organizations increases, the transaction rate is adequate for a Brazilian national level.

The third experiment evaluates the data access time after confirmation of the purchase in the blockchain. The experiment involves creating a Mininet network with three hosts, a switch, and an SDN controller. The host h_1 simulates a server that stores the data for sale in the blockchain. A buyer client issues a purchase transaction TX_{pur} by sending the IP address of the host h_2, which makes a request to h_1 and waits for a response. The first request time is longer because of the controller checks and processes the pending transactions in the queue. The average response time obtained is 0.473 seconds. This result proves that the access time to the acquired data is fast and imperceptible, adequately meeting the interaction with the purchasing customer.

VI. Conclusion

This paper presented an architecture based on software defined networking and blockchain that allows the commercialization of data in a secure, automatic, and distributed way between users of the system. We designed and realized a smart contract to securely and automatically control IoT-personal data trading between domains. We developed a prototype running on the Hyperledger Fabric platform, in which a personal data owner can easily announce and commercialize his data in a secure, efficient, and automatic way. The results prove

[4] MercadoPago had 227 million sales in the third quarter of 2019. Available at: https: //ideias.mercadolivre.com.br/sobre-mercado-livre/mercado-livre-cresce-368-em-vendas-e-atinge-us-76-bilhoes-em-volume-de-pagamentos-com-mercado-pago-no-3o-tri/



(a) Transaction rate for an increasing number of clients, assuming a single organization running in a single computer.

(b) Transaction rate for an increasing number of clients, assuming a single organization running in three computers.

Fig. 4: Transaction rate assuming a single organization.

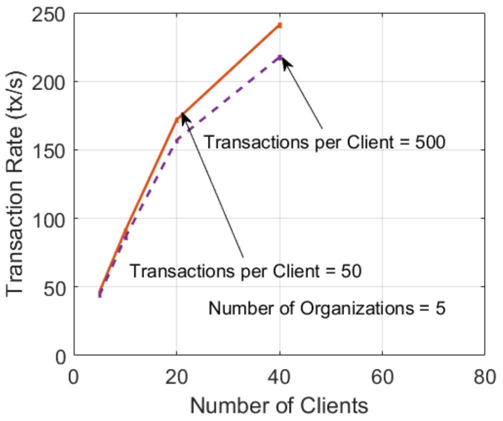


Fig. 5: Transaction rate for an increasing number of clients, assuming five organizations running in five different servers.

that the proposed system implementation meets the Brazilian e-commerce demand, reaching over 50 transactions per second even when a large number of users issue simultaneous transactions with a physical test server. Extending the test scenario to multiple physical servers with higher processing power can significantly increase the transaction rate.

As future work, we plan to implement a reputation system to mitigate malicious behavior such as the sale of corrupted data or deletion of data from the server after purchase and implementation in a cluster of servers.

VII. Acknowledgment

This work was financed by CNPq, CAPES, FAPERJ and FAPESP (18/23292-0, 2015/24514-9, and 2014/50937-1).

References

[1] J. Paillisse, J. Subira, A. Lopez, A. Rodriguez-Natal, V. Ermagan, F. Maino, and A. Cabellos, "Distributed access control with blockchain," in *IEEE - ICC 2019*, May 2019, pp. 1–6.

[2] M. A. Lopez, D. M. F. Mattos, and O. C. M. B. Duarte, "An elastic intrusion detection system for software networks," *Annals of Telecommunications*, vol. 71, no. 11, pp. 595–605, 2016.

[3] E. Androulaki *et al.*, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Thirteenth EuroSys Conference*. ACM, 2018, p. 30.

[4] G. A. F. Rebello, G. F. Camilo *et al.*, "Providing a sliced, secure, and isolated software infrastructure of virtual functions through blockchain technology," in *IEEE HPSR*, 2019, pp. 1–6.

[5] G. A. F. Rebello, I. D. Alvarenga, I. J. Sanz, and O. C. M. B. Duarte, "BSec-NFVO: A blockchain-based security for network function virtualization orchestration," in *ICC 2019*, 2019, pp. 1–6.

[6] I. D. Alvarenga, G. A. F. Rebello, and O. C. M. B. Duarte, "Securing configuration management and migration of virtual network functions using blockchain," in *IEEE/IFIP Network Operations and Management Symposium NOMS 2018*, Apr. 2018, pp. 1–9.

[7] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, "FairAccess: A new blockchain-based access control framework for the internet of things," *Security and Commun. Networks*, vol. 9, pp. 5943–5964, 2016.

[8] O. J. A. Pinno, A. R. A. Grégio, and L. C. De Bona, "ControlChain: A new stage on the IoT access control authorization," *Concurrency and Computation: Practice and Experience*, 2019, e5238 cpe.5238.

[9] Y. Hu, S. Kumar, and R. A. Popa, "Ghostor: Toward a secure data-sharing system from decentralized trust," in *17th USENIX NSDI 20*. Santa Clara, CA: USENIX, Feb. 2020, pp. 851–877.

[10] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of IoT data," in *Cloud Computing Security Workshop*, ser. ACM CCSW '17, 2017, p. 45–50.

[11] H. T. T. Truong, M. Almeida, G. Karame, and C. Soriente, "Towards secure and decentralized sharing of IoT data," in *2019 IEEE Blockchain*, 2019, pp. 176–183.

[12] M. Krämer, S. Frese, and A. Kuijper, "Implementing secure applications in smart city clouds using microservices," *Future Generation Computer Systems*, vol. 99, pp. 308–320, 2019.

[13] I. Cervesato, "The Dolev-Yao intruder is the most powerful attacker," in *16th Logic in Computer Science—LICS*, vol. 1, 2001.

[14] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling hyperledger fabric to 20,000 transactions per second," in *2019 IEEE ICBC*, 2019, pp. 455–463.