UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA

FILIPE LAUTERT

DISTRIBUTED DATA PROVENANCE: FOG COMPUTING AND BLOCKCHAINS IMPROVING PRIVACY CONTROL, TRUST AND RELIABILITY

DISSERTAÇÃO DE MESTRADO

CURITIBA

2020

FILIPE LAUTERT

DISTRIBUTED DATA PROVENANCE: FOG COMPUTING AND BLOCKCHAINS IMPROVING PRIVACY CONTROL, TRUST AND RELIABILITY

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação Aplicada da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de "Mestre em Ciências" – Área de Concentração: Computação Aplicada.

Orientador: Luiz Gomes-Jr

Co-orientador: Daniel F. Pigatto

CURITIBA

2020

Dados Internacionais de Catalogação na Publicação

Lautert, Filipe

Distributed data provenance [recurso eletrônico]: fog computing and blockchains improving privacy control, trust and reliability / Filipe Lautert. -- 2020.

1 arquivo eletrônico (75 f.): PDF; 2,33 MB.

Modo de acesso: World Wide Web.

Texto em inglês com resumo em português.

Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Computação Aplicada. Área de Concentração: Engenharia de Sistemas Computacionais. Linha de Pesquisa: Sistemas de Informação, Curitiba, 2020.

Bibliografia: f. 54-56.

1. Computação - Dissertações. 2. Blockchains (Banco de dados). 3. Computação em nuvem. 4. Processamento eletrônico de dados - Processamento distribuído. 5. Arquitetura de software. 6. Interface de programas aplicativos (Software). 7. Sistemas de recuperação da informação. 8. Proteção de dados. 9. Confiabilidade. 10. Aplicações Web. 11. Simulação (Computadores). I. Gomes Junior, Luiz Celso, orient. II. Pigatto, Daniel Fernando, coorient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Computação Aplicada. IV. Título.

CDD: Ed. 23 -- 621.39

Biblioteca Central do Câmpus Curitiba - UTFPR
Bibliotecária: Luiza Aquemi Matsumoto CRB-9/794

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Pesquisa e Pós-Graduação

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CURITIBA

TERMO DE APROVAÇÃO DE DISSERTAÇÃO

A Dissertação de Mestrado intitulada **"Distributed data provenance: fog computing and blockchains improving privacy control, trust and reliability"**, defendida em sessão pública pelo candidato(a) **Filipe Lautert**, no dia 02 de outubro de 2020, foi julgada para a obtenção do Título de Mestre em Computação Aplicada, Área de Concentração Engenharia de Sistemas Computacionais, Linha de Pesquisa Sistemas de Informação, e aprovada em sua forma final, pelo Programa de Pós-Graduação em Computação Aplicada.

BANCA EXAMINADORA:
Prof. Dr. Luiz Celso Gomes Junior – Presidente - UTFPR
Profa. Dra. Rita Cristina Galarraga Berardi - UTFPR
Prof. Dr. Carlos Alberto Maziero - UFPR

A via original deste documento encontra-se arquivada na Secretaria do Programa, contendo a assinatura da Coordenação após a entrega da versão corrigida do trabalho.

Curitiba, 02 de outubro de 2020.

_____

Carimbo e Assinatura do(a) Coordenador(a) do Programa

I dedicate my dissertation work and own a big thanks to my wife Tatiane and daughters Melissa and Sophie, for all the understanding and support throughout the entire program. A special feeling of gratitude to my parents, Nelson and Alice whose always pushed me to study and keep progressing. My sister Raquel and brother Alexandre have always been there for me and are very special.

AGRADECIMENTOS

I would like to thank the following people who have helped me undertake this research: my supervisors Luiz Celso Gomes Jr. and Daniel F. Pigatto for their support, patience and all the time invested on me; the members of the defense board, Rita Berardi and Carlos Maziero for precious guidance and feedback with their reviews; and to my wife Tatiane and daughters Melissa and Sophie for understanding the effort dedicated to this program and supporting me.

RESUMO

LAUTERT, Filipe. DISTRIBUTED DATA PROVENANCE: FOG COMPUTING AND BLOCKCHAINS IMPROVING PRIVACY CONTROL, TRUST AND RELIABILITY. 75 f. Dissertação de Mestrado – Programa de Pós-Graduação em Computação Aplicada, Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

Sistemas para Proveniência de Dados permitem o rastreamento da origem e evolução da informação, melhorando a confiança entre as partes. Este é um requisito importante para uma ampla gama de aplicações, como vigilância sanitária de alimentos, cadeias de abastecimento e monitoramento de surtos epidêmicos, entre outros. Muitas dessas aplicações são inerentemente distribuídas e exigem altos níveis de privacidade e confiança.

Computação em névoa e Blockchains são soluções tecnológicas recentes que nasceram de avanços da computação em nuvem e distribuída. A computação em névoa concentra-se em trazer a nuvem para mais perto do usuário final, enquanto Blockchain fornece transparência sem uma entidade centralizada. Ambos podem ser complementares, com a computação em névoa distribuindo dados e o armazenamento, enquanto o Blockchain mantém a consistência e confiabilidade.

Este trabalho descreve uma arquitetura que permite o rastreamento da proveniência dos dados em uma camada de névoa distribuída. Blockchains são empregados para fornecer transparência, e cada nó da névoa tem controle sobre o que é tornado público na nuvem. Por meio de serviços que mantêm as informações consistentes entre todas as partes interessadas, a arquitetura proposta permite proveniência de dados rápida e confiável para clientes executando na névoa. As informações do sistema estão associadas a uma prova de autenticidade, e os autores têm controle sobre a publicação dessas informações.

A arquitetura foi desenvolvida com base no modelo de proveniência Prov definido pelo W3C, simplificando a adoção da solução. Foi desenvolvida uma aplicação composta por um cliente e uma API restful que é capaz de armazenar e compartilhar informações de proveniência em um Blockchain usando padrões abertos. São relatados resultados de testes extensivos demonstrando que a arquitetura proposta tem um desempenho adequado em vários cenários de recursos e diferentes níveis de confiabilidade de rede.

Palavras-chave: Proveniência de dados, Blockchain, computação em névoa

ABSTRACT

LAUTERT, Filipe. DISTRIBUTED DATA PROVENANCE: FOG COMPUTING AND BLOCKCHAINS IMPROVING PRIVACY CONTROL, TRUST AND RELIABILITY. 75 f. Dissertação de Mestrado – Programa de Pós-Graduação em Computação Aplicada, Universidade Tecnológica Federal do Paraná. Curitiba, 2020.

Data Provenance systems enable tracking of the origin and evolution of information, improving trust among parties. This is an important requirement for a wide range of applications such as food safety, supply chains, and monitoring of epidemic outbreaks. Many of these applications are inherently distributed and require high levels of privacy and trust.

Fog computing and Blockchains are recent technological solutions that were born from advancements in Cloud and distributed computing. Fog computing focuses on bringing the Cloud closer to the edge user while Blockchain provides transparency without a trusted centralized entity. Both can be complimentary as Fog computing spreads the data and computer storage while Blockchain can keep it consistent and trustworthy.

This dissertation describes an architecture that allows the tracking of data provenance in a wide-area distributed Fog layer. While Blockchains are employed to provide transparency, each Fog node has control over what is made public on the Cloud. The architecture proposed in this paper enables fast and reliable data provenance for clients executing in the Fog node using software services that keep the information consistent across all interested parties in the Cloud. Information in the system is associated with a proof of authenticity, but authors have control over the eventual publication of such information.

The architecture was built upon the well established provenance model W3C Prov, which simplifies adoption of the framework. An application was developed consisting of a client and a restful API that is able to store and share provenance information in a Blockchain using open standards. Results from extensive tests are reported showing that the proposed architecture performs adequately in several scenarios of varying resources and levels of network reliability.

Keywords: Data Provenance, Blockchain, Fog Computing

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

W3C	World Wide Web Consortium
API	Application Programming Interface
DBMS	database management systems
POW	Proof of Work
POS	Proof of stake
HDFS	Hadoop Distributed File System
IoT	Internet of Things
BFT	byzantine fault tolerance

SUMÁRIO

1 INTRODUCTION

Provenance is the process or technique used to track the origin, authorship, and history of any given object. It was originally used in the context of works of art to certify that an object was created by the claimed author. Nowadays it is being used in other fields in a similar fashion, namely archaeology, paleontology (FEIGENBAUM et al., 2012), archives, manuscripts, printed books (PEARSON, 1994), science, and computing (TAN et al., 2013).

Data Provenance (also known as Data Lineage or Data Pedigree (ROMEU, 1999)) aims to track the history of a piece of data instead of an object, starting from its original source and accounting for all the transformations that it goes through. Currently information is readily available, but for many applications it is important to make sure that the information comes from a trusted source and that it is unchanged; if something is changed, it is imperative to know by whom and what was changed.

Distributed applications in heterogeneous environments (e.g. the Internet) pose new challenges for provenance services. Issues such as origin trust, privacy control and communication reliability become critical elements in the design of services. For example, in a global system for tracking epidemic outbreaks, all of those aspects are important requirements. All participating regions must join the system, with equal terms and privileges. Therefore, there is no predetermined centralized authority, providing trust in the system. Furthermore, health units need some level of autonomy to allow time for investigating cases before disclosing contamination information to the global network, allowing for privacy control. However, it is also important that regulatory organizations become aware of any new possible cases as soon as possible, providing reliability to the information provided. Finally, local health agents may need to connect to the system from mobile devices or other settings with unreliable connectivity. These requirements can be found in other applications, such as food safety, supply chain, etc.

New technologies can be used to share provenance information with transparency, immutability and reliable distributed access.

Blockchains are distributed immutable ledgers that track and register all transactions

on chains: blocks of information that are tied together. Fog computing consists on the allocation of computational resources close to the edge of the network. Although very similar to the Cloud as it provides data, computer storage, and services to end-users, Fog computing is closer to the edge devices, which leads to better performance and superior user experience. It is important to notice that Fog and Cloud are complimentary as the Fog provides limited resources that can be offloaded to the Cloud.

The work on this dissertation makes use of recent technologies to facilitate the process of managing and sharing provenance information in a distributed environment that is susceptible to network failures. This solution is as generic as possible to be used by anyone that requires a provenance service and not just by a single use case.

The architecture implements a Fog computing infrastructure that enables Blockchain-based data provenance close to edge devices. Clients are able to store provenance records in a private Blockchain, enabling privacy control and providing reliable information to the Cloud about the records from a trusted source. Once the information is ready to be shared it can be disclosed to the Cloud. This approach reduces the complexity of the data provenance system by delegating the task of sharing information to the Blockchain. It also provides good performance since most of the data processing is kept in local Fog clusters.

The main contributions of this dissertation are: (i) a review of the state of art of related topics, (ii) an architecture that allows distributed data provenance to be stored in the local Fog node, producing a proof of authenticity in the Cloud Blockchain, (iii) the implementation of a distributed service to demonstrate the practicality of the solution, and (iv) performance tests and analyses that demonstrate the practicability of this approach.

1.1 OBJECTIVES

1.1.1 GENERAL OBJECTIVE

Develop an architecture that enables distributed data provenance that addresses: (i) decentralized trust, (ii) localized performance and reliability, (iii) local control over data publication.

1.1.2 SPECIFIC OBJECTIVES

- Review the state of art for Provenance, Blockchain, and Fog Computing and how they can be used together

- Design an architecture using Blockchains to provide trust and Fog computing to provide privacy control and reliability
- Implement a prototype of the solution
- Implement a simulation of the architecture that enables tests of network failures
- Test the implementation

2 STATE OF ART/RELATED WORK

2.1 DATA PROVENANCE

To understand the basics of what Data Provenance is, bear in mind the following example: document “D” was generated by author “F” and was reviewed by PhD. “R”. After this review a new document “RD” was derived from document “D”. In this example, document “D” is the subject of the Data Provenance. All of the actions that the document was submitted to (generation, review, derivation) are listed and associated with its authors, and finally a new document “RD” was generated and associated with the source document.

Data provenance has a wide range of application scenarios. (GIL et al., 2013) argue that *different people may have different perspectives on provenance, and as a result different types of information might be captured in provenance records*. Thus different definitions and characteristics of what provenance covers are defined by different authors.

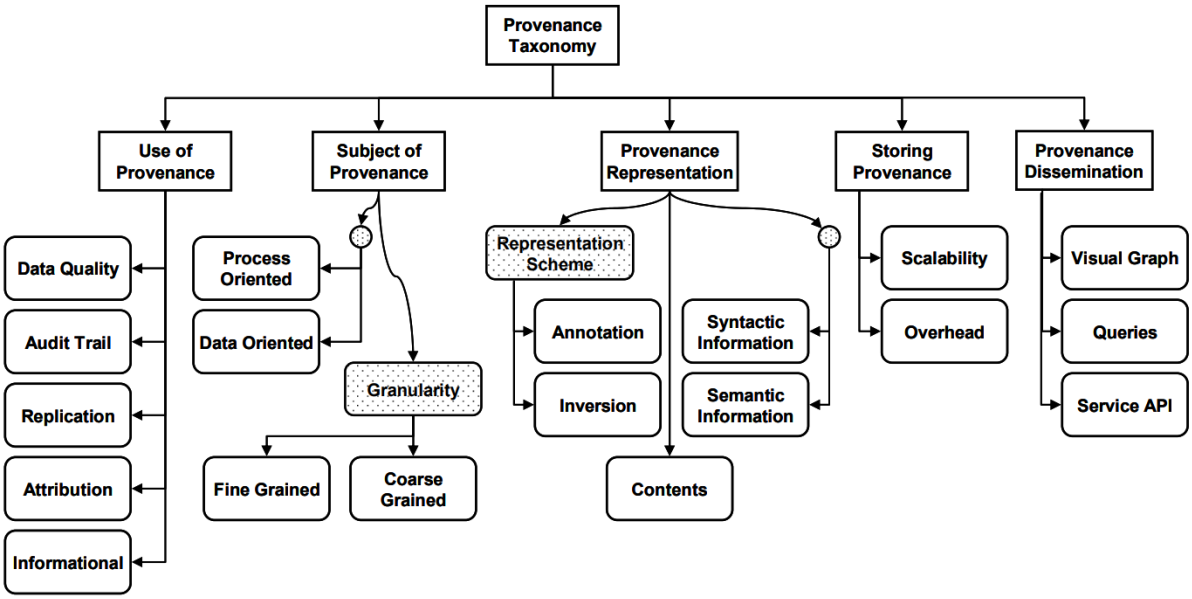
Past research has aimed at describing the characteristics for data provenance. Buneman et al. (2001) introduce the concepts of “why” and “where” provenance aiming mainly at data stored in databases. “Why” provenance refers to the source data that had some influence on the existence of the data, giving an explanation of how the data was obtained. “Where” provenance refers to the location in the source databases from which the data was extracted or modified to get to the actual data.

Simmhan et al. (2005) surveyed many research efforts in data provenance field and created a taxonomy to categorize those efforts. The taxonomy categorize data provenance systems based on characteristics such as why provenance is stored, what they describe, how they represent and store provenance and ways to disseminate it as visualized in Figure 1.

Based on Simmhan et al. (2005) it is possible to see that provenance systems can be built to support a number of uses, having different characteristics and different ways of working. Based on Figure 1 , Provenance can be used for:

- Data quality: ensure the quality and reliability of the data, as one can track the data

Figura 1: Provenance taxonomy



Fonte: Simmhan et al. (2005)

origins, who changed it and what was changed.

- Audit trail: used for audit purposes, as completeness and integrity characteristics ensure that it should not be changed and that must have all action logged to it.
- Replication Recipes: allow a piece of information to be reproduced as it stores the changes applied to it.
- Attribution: as it tracks the author and source of the information, it is used to determine the responsible for that data and enable its citation.
- Informational: used as a source of reference for context discovery and to understand the information.

Bauer e Schreckling (2013) define characteristics and requirements for data provenance used on the Internet of Things (IoT), which is a recent use for provenance.

Those characteristics can also be applied to the internet and any other recent usages for provenance:

- Completeness: Every single action which has ever been performed is gathered.
- Integrity: Data has not been manipulated nor modified by an adversary.

- Availability: The possibility to verify the collected information. In this context, availability is comparable to auditing.
- Confidentiality: The access to the information is only reserved to authorized individuals.
- Efficiency: Provenance mechanisms to collect relevant data should have a reasonable cost.

Other authors may have different definitions for data provenance and those definitions can still be improved, but Bauer e Schreckling (2013) definitions are used as the baseline for data provenance requirements for this work.

2.2 W3C PROV STANDARD

Groth e Moreau (2013), as members of the W3C, present the "Prov Family" (PROV) of documents that define a model and related operations enabling inter-operable interchange of provenance information. Those documents were developed after an extensive research including use case cataloging, requirements elicitation and a literature survey.

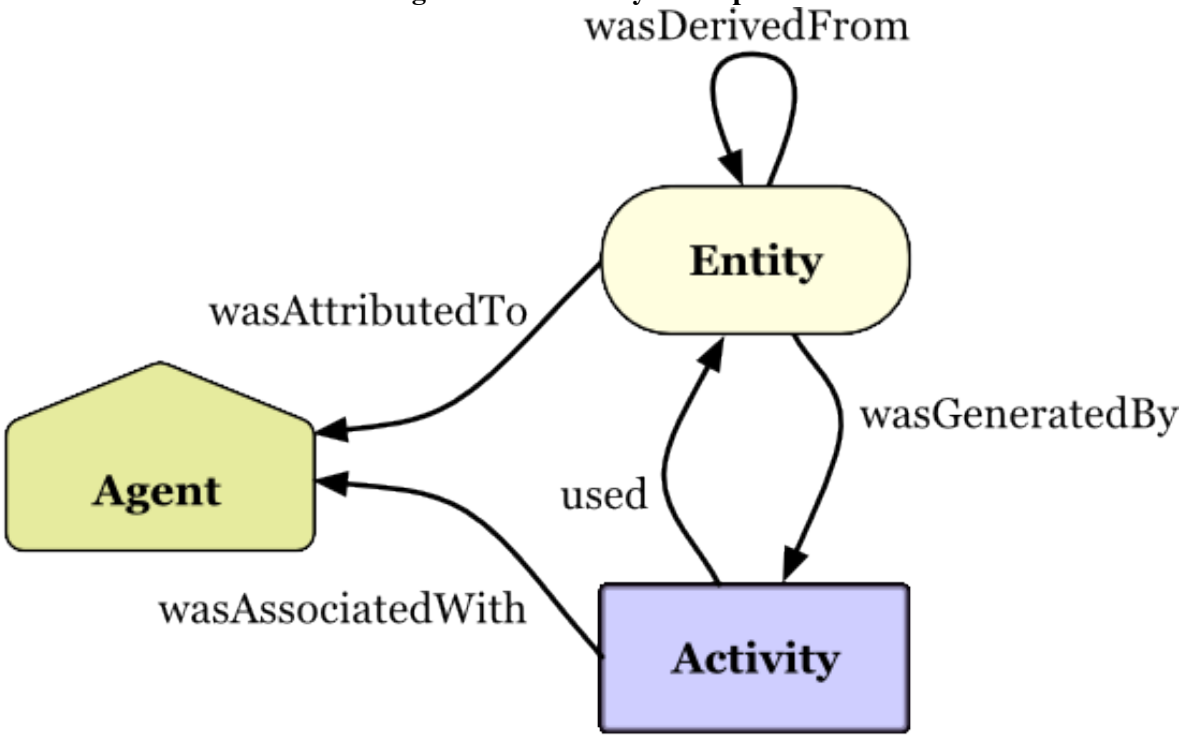
PROV papers consist of twelve documents that can be used as a reference to design provenance representation, especially in heterogeneous environments such as the Web. Programming libraries have been implemented to generate and handle those documents as published in the PROV Implementation Report[1], that is the specification used to represent provenance and generate the documents used in this work.

Figure 2 illustrates the most important concepts of PROV and how they relate to each other:

- **Entities**: the objects that are the target of provenance are entities. Entities can be related to each other and can have many attributes such as version, location, author, etc
- **Activities**: represents actions or processes applied to an entity, for instance: entity X was generated by a review of entity Y (review is the activity).
- **Agents**: one or more agents can be associated with entities and/or activities, and those associations are labelled as roles or responsibilities.

[1] https://www.w3.org/TR/prov-implementations/, visited on 2020-04-13

Figura 2: PROV Key Concepts



Fonte: Belhajjame et al. (2013)

The example from Data Provenance section can be mapped to Prov concepts as follow: provenance document 'D' (entity) was generated (activity) by author 'F' (agent) and was reviewed (activity) by PhD. 'R' (agent). A new document 'RD' (entity) was derived from document 'D' (entity to entity association).

A complete review of W3C PROV documentation can be found on appendix A.

2.3 BLOCKCHAIN FOR DATA TRANSPARENCY AND PROVENANCE

There are several efforts in integrating Blockchain technologies in general purpose information systems. A general concern regards the trade-off between transparency and performance, especially in data-intensive applications. Lautert et al. (2020a) compares Blockchains and database management systems (DBMS), demonstrating the practicability of a Blockchain-based architecture.

Fostered by their flexibility, Blockchains are being used for many different applications. Azaria et al. (2016) propose a Blockchain for medical data access and permission management called MedRec. This solution is a good example where a Blockchain that is available on the market was used as a backend to a complete solution which provides APIs,

security and linking protocols all built relying on the Blockchain as its safe storage.

There are some approaches of using Blockchains as storage systems, mostly using it as a replacement for a database. Gaetani et al. (2017) designs a Blockchain-based database for Cloud environments that aims to be effective in terms of speed by using the "total consensus" algorithm instead of Bitcoins'[2] "proof of work" (GERVAIS et al., 2016). McConaghy et al. (2016) takes a different approach and implements Blockchain capabilities using different instances of the distributed database RethinkDB (WALSH et al., 2009) to store the backlog of transactions and the chain. Both approaches depend on the creation and maintenance of a blockchain network. To prevent this work, Liang et al. (2017) uses the ChainPoint API[3] to store file operations in the Bitcoin Blockchain to provide an auditable track of modifications applied to a file. The downside of using Bitcoin is to have to spend money to handle the mining of the blocks, while in McConaghy et al. (2016) and Gaetani et al. (2017) the network can even be executed in private mode.

Greenspan (2016) cites provenance tracking as one of the main usages for Blockchain and emphasizes how a Blockchain can be more secure than a database to store provenance information. His main concern is that a database will have a system administrator that can tamper with the data, as for a Blockchain the consensus protocol gives the participants a chance to jointly manage the chain and ensure that no single participant can change it without others approval.

Kamath (2018) writes about Walmart's Blockchain solution using IBM Hyperledger developed to track the supply chain (food provenance) of Pork in China and Mangoes in the USA. An advantage is that it reduced the time for tracking mango origins from seven days to 2.2 seconds, as the information is stored in the Blockchain instead of having the information distributed across many locations. This is an example where businesses are using Blockchain to track provenance.

By using a standard protocol and a REST API, the proposed architecture provides a generic provenance tool that could be used by any of the previously cited examples without dealing with all the complexities of integrating a Blockchain with their solutions.

2.3.1 BLOCKCHAIN STORAGE OPTIONS

Information can be stored in a blockchain in any way that is required by the application - the storage form usually is not implemented by the blockchain but by the provenance system.

[2] https://bitcoin.org/bitcoin.pdf, visited on 2020-08-11
[3] https://chainpoint.org/, visited on 2019-06-22

In terms of the actual data in the blockchain, there are some popular options:

- plain: data can be read by anyone that has access to the blockchain, providing transparency and full access to everyone. This dissertation uses the plain approach to facilitate data validation and to provide data transparency, but does not prevent source systems to send enclosed information encrypted, hashed or in the form of pointers;
- encrypted: can be read by the ones that have access to the required key to decode it; if encrypted with a private key, the decryption with a public key can act as a form of signature to ensure that the data was added by a certain party;
- hashed: only a hash of the information is stored in the blockchain, and the original data is stored in another place that can be accessed only by the interested participants. In this case, the blockchain acts as a tool to ensure that the data was not modified by keeping the hash as an immutable source;
- pointers: instead of storing the data, a pointer to it is stored. It can be a restful URL, an IPFS address [4], a SQL query to be executed in a predefined database or anything that the system developers can come up with.

2.4 BLOCKCHAINS AND DATA PROVENANCE

Blockchain can help to keep track of data provenance as it has features that can be applied to information technologies challenges as discussed in section 2.3. In this section we revisit those features and explain how they can be used to provide provenance information.

2.4.1 BLOCKCHAIN FEATURES

Blockchains are distributed ledgers that track all transactions and register them in blocks of information that are linked. To achieve its goal blockchains implement characteristics that can be explored in the context of data provenance, namely:

Immutable information based on the hash of the block: information stored in blockchains is immutable by nature as every block is tracked by a hash. This characteristic helps to ensure that provenance information written to the blockchain can not be changed.

Distributed ledger: blockchains are developed to be distributed. The idea of a blockchain is to enable the sharing of the ledger with anyone that is interested in reading the

[4] https://ipfs.io/#how, visited on 2019-08-30

information. There may be security configurations that restrict a blockchain sharing when in the case of a private blockchain, but the most famous blockchains as Bitcoin and Ethereum are public and anyone can get a copy of the whole chain.

Transaction signing: generally blockchains will hold transactions, but to ensure that a transaction was really executed by someone it needs to be signed by this person's private key and can be verified using his public key. Every transaction in a block will be signed.

Free form data storage: the first blockchain (Bitcoin) was created aiming solely to hold transactions and it already had some extra fields that could be used to store any random piece of information. Other blockchains allow their administrators to configure what kind of information it will hold, and even some private blockchains as Hyperledger [5] allow the user to customize the data that is stored in the chain to suit their business needs.

Smart contracts: smart contracts are a way to execute a piece of code that will validate a transaction or execute other tasks using the transaction data. In the original blockchain the idea for smart contracts was to just validate the transaction, but in other implementations it can be used for anything using the data, from validations to code execution. The main problem of using those contracts is that usually there are associated costs, and as it is usually executed in an interpreted language by the blobkchain it may be slow and costly to run.

Consensus mechanism (Proof of Stake): blockchains have many consensus mechanisms, being proof of work (POW) and proof of stake (POS) the most commons. POW is the original consensus algorithm from Bitcoin that has been widely used in other blockchains. In this algorithm the blockchain participants need to spend computational power to calculate a challenge to prove that they worked to find the block, and they are rewarded with something (usually a bit of the crypto currency). Depending on the complexity level of the riddle it may be very expensive in terms of energy required to find the solution: Stoll et al. (2019) says that the carbon emissions produced by Bitcoin sit between the levels produced by the nations of Jordan and Sri Lanka, which is comparable to the level of Kansas City in the US.

POS selects the creator of the next block using various combinations of random selection and wealth (how much or the proportion of money that one have on the system) or age of the participant. So it does not require a lot of computational power neither miners to decide who is the next participant to generate a new block, and the random selection parameters and wealth can be tuned if one own the chain.

There are other consensus algorithms as Proof of Importance (productive activity on

[5] https://www.ibm.com/blockchain/hyperledger, visited on 2019-08-17

the network should be rewarded), Proof of Burn (users need to generated unspent addresses in exchange for a native cryptocurrency), Practical Byzantine Fault Tolerance (participants confirm messages sent to them by running a computation to determine its decision about the message's validity and get to a consensus), Proof of Elapsed Time (ensuring every participant has equal opportunity to produce a block) [6] and many others that appears as the technology advances.

Highly customizable implementations: as most of the blockchains are open source it is easy to simply get the source code and suit it to a specific end – one can change how it works or even just implement a new consensus algorithm and plug it into the blockchain. Besides that the Blockchains already have a lot of configuration options out of the box as explained at configuration pages of Ethereum [7] and Tendermint [8].

Public and private options: it is possible to run a private blockchain, with no requirements to open it to the world. This blockchain can be tamed to a private organization and access can be granted just to the ones that this organization has a contract with.

Applying blockchain to provenance: blockchains can be a great tool to track information in a distributed way ensuring that no one can tamper with the data stored on that. In fact, some big companies already use blockchains to keep track of their supply chain [9] and there are online courses and companies selling blockchain solutions for supply chain [10] [11] – and data provenance can be related to a supply chain solution.

2.5 COMPARING BLOCKCHAIN PROVENANCE IMPLEMENTATION WITH EXISTING SYSTEMS

Pérez et al. (2018) provides a complete and recent systematic review of provenance systems. It reviewed 251 systems and focused on the 25 with most references in the literature. The authors define that a provenance systems must support generating, querying, and storing of provenance information, and defines a taxonomy to compare and classify the provenance system features as shown in Figure 3. This taxonomy is different from Simmhan et al. (2005) as it classifies systems features while the prior classifies the provenance itself.

This taxonomy has 6 main aspects, each one of these aspects has categories that

[6] https://www.kryptographe.com/a-guide-to-blockchain-consensus-mechanisms/, visited on 2019-08-18

[7] https://github.com/ethereum/go-ethereum/wiki/Private-network, visited on 2019-08-20

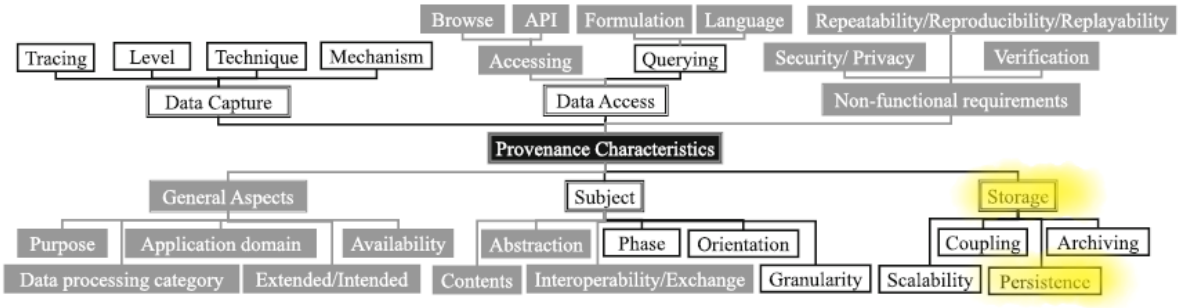
[8] https://tendermint.com/docs/tendermint-core/configuration.html, visited on 2019-08-20

[9] https://www.forbes.com/sites/bernardmarr/2018/03/23/how-blockchain-will-transform-the-supply-chain-and-logistics-industry/, visited on 2019-08-25

[10] https://openledger.info/solutions/blockchain-supply-chain/, visited on 2019-08-25

[11] https://www.udemy.com/blockchain-supply-chain-management/, visited on 2019-08-25

Figura 3: Taxonomy of Provenance Systems Features



Fonte: Pérez et al. (2018)

correspond to concrete aspects to focus on when comparing provenance systems. The work covers systems released in the time span 1993–2014, noticing an increasing interest from 2004 onwards. There is also a large number of connections among the papers defining the systems, showing that provenance systems' developers have been aware of other solutions. A detailed review of the work including a list of all the researched provenance systems can be found at the Supplementary material of the the paper[12]. Of all of those aspects and categories, the Blockchain implementation differentiate itself the most in aspect *Storage*, category *Persistence* (highlighted in yellow in Figure 3): the majority of the 25 main surveyed systems use a relational database as the persistence system, either alone or combined with other storage techniques; 2 of the researched systems use a NOSQL system, 2 do not specify what kind of DB is used, and others use a XML-based format; one of the surveyed system called RAMP has been built as an extension of Hadoop and uses the Hadoop Distributed File System (HDFS) for storing data across multiple machines, but none of them store the information in a Blockchain.

Pérez et al. (2018) also discusses the importance of standardisation of the model for managing provenance. It is highlighted that the absence of using standards for collecting, representing, storing, and querying between the researched system as being a huge problem to promote integration between the solutions. The paper makes clear that using a standard as the W3C PROV contributes to the sharing and use of provenance information, and this is the standard that is being used in the Blockchain approach.

[12] `http://www.unirioja.es/cu/beperev/SupplementaryMaterial.html`, visited on 2020-01-06

2.6 FOG COMPUTING

Common information systems architectures tend to have centralized information in servers or clusters located in the Cloud. Fog computing goes the other way and aims to bring computing resources closer to the edge of the network. The objective of this movement is to improve user experience by making the communication more efficient and eliminating the transfer of irrelevant data to the Cloud. The importance of this architecture has been brought by IoT devices that usually require data to be processed in real time and their need to be continuously connected to the network.

Fog Computing is a recent field of study that was introduced by Cisco (2014). Marín-Tordera et al. (2017) question what a Fog node is and review the trends towards a definition. They end up defining that Fog nodes are distributed Fog computing entities that will enable services close to Edge devices. This dissertation assumes that Fog computing may be observed as an intermediary structure that ranges from car sensors connected to a cellular tower with dedicated hardware to local networks with servers providing services to local devices. It relies on Fog nodes to keep information private to a location and to improve response performance for local clients, specially in cases where the connection to the Cloud is unstable.

Recently many works have been trying to associate Blockchains with Fog computing. Kaur et al. (2019) proposes a lightweight authentication mechanism for Vehicular Fog Computing (VFC) providing distributed authentication and key-exchange scheme using Blockchain to replicate the information among Fog nodes. In this architecture, vehicular Fog data-centers (VFDs) are responsible for provisioning services to users holding a server that is responsible for maintaining the Blockchain for that region.

Jang et al. (2019) proposes a system architecture for Industrial IoT split into Cloud, Fog, and Edge to organically operate the IoT ecosystem. The system aims to solve performance issues by not connecting IoT devices directly to the Blockchain by delegating this task to a Fog server ensuring stability, security and scalability. The Blockchain is responsible for replicating data between different Fog nodes.

Sharma et al. (2017) proposes a secure distributed Fog architecture using software defined networking (SDN) and Blockchain techniques bringing computing resources to the edge of the IoT network. They propose a novel consensus protocol named “Proof-of-Service” that enables Fog nodes to provide services to other Fog nodes using their spare resources.

One characteristic that all contributions have in common is that they all use Blockchains to share information between different Fog nodes. Edge devices have faster access

to information and resources because they are kept close to them.

3 PRIVACY-PRESERVING DATA PROVENANCE USING BLOCKCHAINS

3.1 SECURITY CONSIDERATIONS

The architecture aims to mitigate threats where an attacker wants to mislead auditors that are using the shared information. Such attacker would try to tamper with the information in a document that was not yet released. To mitigate this problem the document hash is shared to the Cloud, so when information is released the authenticity can be confirmed by the final users. This way the privacy of the document is preserved and it is secured against modifications.

The Blockchain accomplishes what is required for the provenance proposed in the use case by protecting the hash and data from modification and enabling the exchange of information between Fog nodes. If data can be added just by specific clients the Blockchain services must be protected by firewalls and URL proxies.

Besides the described scenario, there are many other attacks that can be executed against a Blockchain. For instance, Tendermint blockchain (the blockchain used in this work, further discussed in section 4.1) uses a public/private key pair to identify itself as a validator node; if an attacker gains access to the server host he/she can copy this key and use it to set up a new validator node. Denial of service attacks can bring down a node and prevent the Blockchain from reaching consensus.

Fog, Blockchain and Provenance threats were reviewed but it is out of the scope of this work to evaluate and mitigate these forms of attack. The review of related work can be found on appendix C.

3.2 USE CASE AND PROPOSED ARCHITECTURE

The usage scenario is inspired by a hypothetical global disease outbreak surveillance system. Outbreak surveillance is a distributed effort, crossing political boundaries and requiring trust and reliability in the most diverse situations. In this scenario, local health units investigate and report contamination cases to make regulatory organizations aware of possible outbreaks.

Local networks, especially in field surveys, may be unreliable. Furthermore, health workers need some autonomy to manage suspicious cases that need further investigations or tests to be confirmed. However, a proof that a case is under investigation has to be registered and published globally in a timely manner.

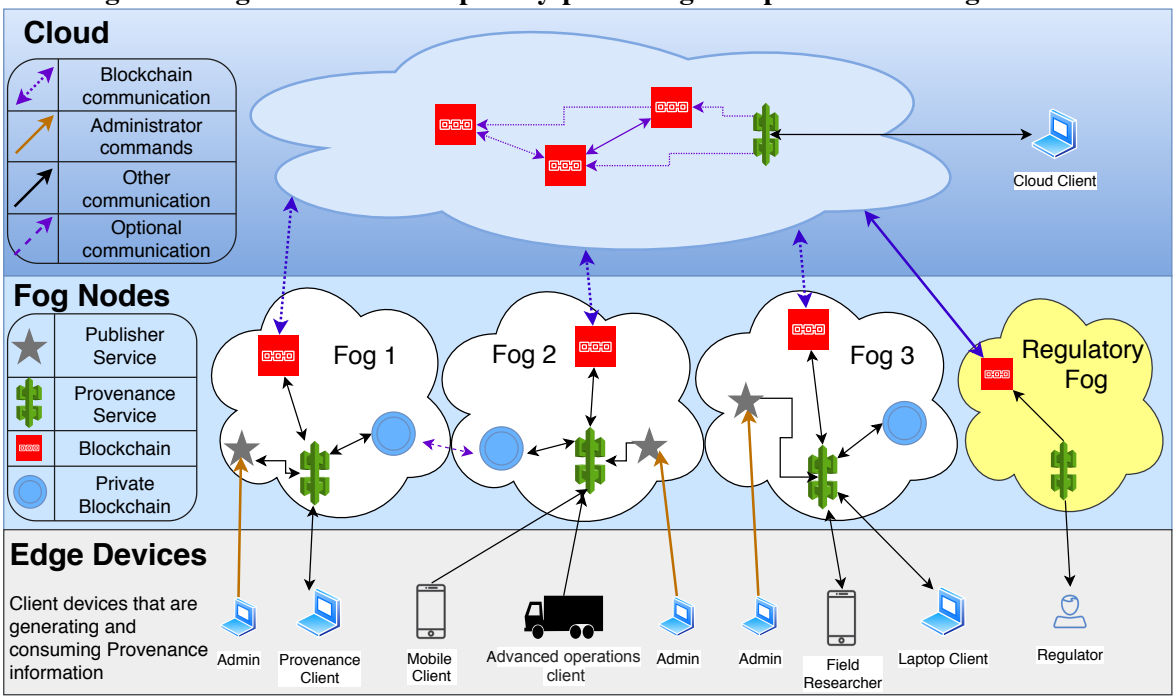
The use case scenario suggests a distributed, two level architecture supporting both localized autonomy and globalized trust. Figure 4 shows the main components of the proposed architecture split in 3 layers: the Cloud represents a multi-party international organization responsible for overseeing trends, Fog nodes represent local health units responsible for daily surveillance of cases while Edge Devices are the end users of the services available in the Fog nodes.

The proposed architecture enables data provenance for clients executing in the Fog clusters, allowing information sharing in 2 steps: first, the information is privately stored in the Fog node protecting the information and a W3C Prov document containing the proof of authenticity is sent to the Cloud Blockchain notifying that data has been generated. In the second step, data is shared as an extension of the original document and it can be verified using the proof of authenticity already present in the Cloud, preventing attacks that would change the data in the Fog node. This architecture can be used in any scenario that requires real time creation tracking but data needs to be kept private for a certain period of time, for instance:

- the information needs to be verified in the Fog node before sharing to prevent premature conclusions (e.g. ongoing investigations);
- a research has been finished but is not ready to be presented to the public, but the researcher needs to prove that it was finished at that time;
- it would take a long time to share the data with the Cloud, so first a proof of authenticity is shared to provide the timestamp and the payload is uploaded at a later time (i.e. when a better internet link is available, link is not being used etc.);
- the provenance information is just used in the Fog node and is shared only by request from others.

The architecture works over 2 layers of Blockchains: Fog nodes and Cloud. Figure 4 defines Fog nodes 1,2 and 3 having each one a private Blockchain (blue circle) built for a specific local health unit. However, when the data is stored in the private Blockchain, a proof of authenticity is also registered in the Cloud layer by the “Provenance Service” (located in the Fog nodes) providing trust to the data stored locally. When data is reviewed and can be disclosed, the

Figura 4: Fog architecture for privacy-preserving data provenance using blockchains



Fonte: The author

system administrator publishes the information to the Cloud Blockchain using the "Publisher Service" (star icon). Blockchain communication between nodes happens through the Cloud and is represented by dotted lines.

The architecture requires the following actors:

- System administrator (laptop labeled "Admin"): issues instructions to the Publisher service to disclose a provenance document.
- Edge devices: they are the ones generating and consuming information and are represented as cellphones, routers and laptops communicating with Fog nodes.
- Regulator: organization that consumes the generated provenance information.
- Cloud client (laptop in Cloud layer): a standalone client that accesses the information available in the Cloud using a provenance service instance.

To enable the architecture, the following services are proposed:

- Provenance service (service available in every Fog node): it is a service that handles 2 tasks: a) reads data from the Cloud and private Blockchains and provide it as a provenance

document; b) receives provenance documents from the devices in the edge layer, storing the documents in the private Blockchain and sharing the proof of authenticity to the Cloud Blockchain. Each Fog node has its own instance of this service. The Cloud has its own copy of the service, which always stores the information in the public Blockchain.

- Publisher service (star icon in Fog nodes 1,2 and 3): service that receives instructions from the system administrator (orange arrows) to share the information from the private Blockchain. It makes use of the provenance service to handle this task.

- Private Blockchain (blue circle inside Fog nodes 1, 2 and 3 in pink): stores the local private data. A Blockchain is used to store local information over any other type of storage to have a standard storage engine in the architecture; it also allows sharing of information between Fog nodes in cases where an institution has more than one Fog cluster, as outlined between nodes 1 and 2 by the purple dotted line.

- Blockchain (red square icons): responsible for sharing the information between Fog nodes and anyone else interested in the information in the Cloud. It provides reliability to any information added to the chain, protecting the trust provided by the hash shared by the Fog nodes.

- Cloud: the Cloud is used by the Fog node blockchain validators to communicate between themselves. It has a provenance service instance to store published data allowing public access to documents.

- Regulatory Fog (last and yellow Fog node): the regulatory Fog node differs from the other Fog nodes as it is read only, thus not requiring the Publisher Service. As an option to having their own Fog node, Regulators could access the information directly from the "Provenance Service" in the Cloud.

Contextualizing the described elements in the use case, when a health agent visits a locality he/she collects information from an individual that is suspect of being contaminated. Using an edge device such as a smartphone, the agent sends the information to the local Provenance service. The provenance service stores the data on the private Blockchain and sends a proof of authenticity to the public Blockchain making regulators aware of the activity. Agents in the health unit would then use the information from the private Blockchain to investigate the case. After a conclusion has been reached, the system administrator instructs the Publisher Service to share the actual data to the Cloud Blockchain, making it available to the regulators and the general public.

3.3 INTERFACE DEFINITION

The provenance features were designed over well established standards. The reference implementation is the ProvStore service. ProvStore provides a service that can be used as a repository to store and manage documents on the W3C Prov standard. Huynh e Moreau (2014) describe the implementation and explain the features including provenance views that focus on specific characteristics such as Data Flow, Process Flow and responsibility. The service provides a Web Interface and a Rest API to allow users to use it programmatically.

ProvStore is available in the Open Provenance umbrella[1], and anyone can create a free account to use their services. There are client implementations such as the Python library provstore-api [2] that can be used to access the Rest API and manage W3C PROV documents.

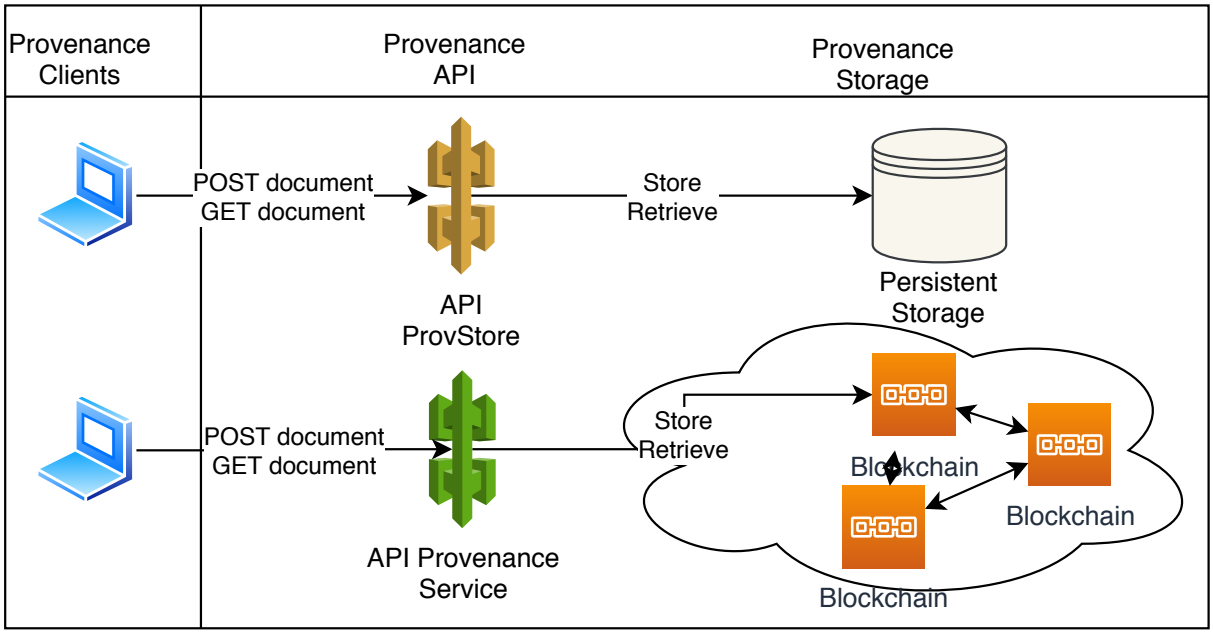
Three REST endpoints provided by ProvStore were chosen to be implemented in the architecture. In figure 5 there is a comparison between ProvStore and the implementation: the upper flow illustrates a request to ProvStore storing the provenance data in a persistent storage, and the lower flow illustrates a request to the service in green that stores the information in a Blockchain. Both requests are sent from an API client and have the same methods signatures and response making the calls interchangeable, meaning that one can point to any of the APIs just changing the server address. The 3 REST endpoints implemented are:

- POST method to store documents. It receives a document key and the document as json, returning some information about the processing, store time and a generated id;
- GET method that searches for a document id given the document name;
- GET method that retrieves a document by id.

A complete review of ProvStore API can be found at appendix B.

[1] https://openprovenance.org/store/, visited on 2019-09-07
[2] https://github.com/millar/provstore-api, visited on 2019-09-10

Figura 5: Comparison between ProvStore and Blockchain-based provenance API calls



Fonte: The author

4 EXPERIMENTS

4.1 IMPLEMENTATION

The Blockchain implementation used was Tendermint which has a REST API interface that allows operations to store and retrieve data from the Blockchain. It uses the Tendermint consensus protocol that is a byzantine fault tolerance (BFT) Proof of Stake protocol that never forks the Blockchain if it has the presence of at least 2/3 of the nodes (KWON, 2014). As a result, to ensure that the Blockchain is not forked, Tendermint stops generating blocks until it has 2/3 nodes available again.

Tendermint allows for 3 different processing methods to store data thought the REST API:

- fully asynchronous: the API call returns right after receiving the information without doing any validations to the request data;
- asynchronous: the API call returns after the transaction is verified by the node and is validated to be inserted in a block;
- synchronous: returns after the transaction is committed to a block or a timeout/network error is reached.

Using the synchronous option guarantees to the client that the data is securely stored in a block and it is the safest way to use the Blockchain, but it can take seconds to release the client. Asynchronous is a good option for a quick answer but the transaction will not be committed and reach other node until a later time. This dissertation makes use of synchronous and asynchronous methods. Full asynchronous method is not being used as there is no guarantee that the data will be stored in the Blockchain.

The ProvStore Blockchain implementation was developed using the Flask lightweight web application framework[1]. The REST requests to Tendermint API were implemented using

[1] https://palletsprojects.com/p/flask/, visited on 2019-09-10

python's "requests" library instead of a dedicated client library.

4.2 TESTING SCENARIOS

Test scenarios were developed and executed to benchmark the performance of the application. Those tests were performed by sending requests containing a W3C Prov document of around 1kb as a payload to the API. Network packets were dropped in some scenarios to understand how the architecture works under unreliable connectivity.

Lautert et al. (2020b) compares the performance of the proposed architecture executing in a single machine using virtualization to simulate Fog and Cloud architectures. The work concludes that "Fog infrastructure is very important in reducing communication latency with the edge devices". This result helped to design the tests executed in this dissertation.

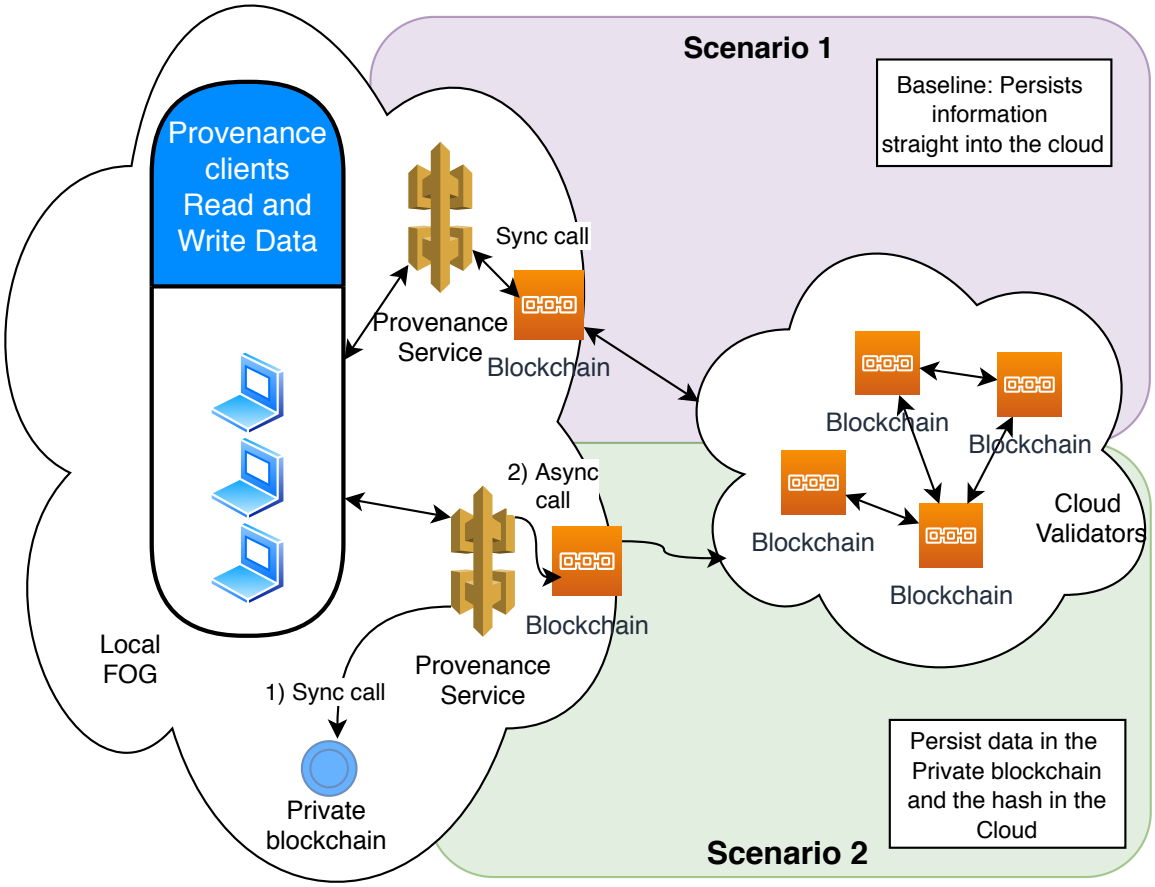
The tests are focused on network failures considering that Fog and Blockchain nodes are secured and can be trusted. First the configurations are executed without any failures or interventions to provide a baseline, followed by tests that represent unstable networks in different interfaces of the system.

As one of the advantages of this architecture is storing data in a local Blockchain, a comparison is proposed against an implementation that stores data directly on the Cloud. Figure 6 illustrates these scenarios.

- In scenario 1 (Cloud), Provenance service communicates only with the local instance of the Cloud Blockchain. To ensure that the data is stored, it sends a synchronous call that waits for confirmation of the insertion in a valid block (after which the client releases the connection).
- In scenario 2 (Fog proposed architecture - FOG), first the Provenance service sends the data to the private Blockchain (blue circle) in a synchronous way. Upon receiving confirmation, the service sends the information's hash to the local instance of the Cloud Blockchain and waits for confirmation that the data was accepted, but does not wait for block generation. After this call it releases the client. The architecture can execute this second call asynchronously because the data is already secured in the local Blockchain. In the eventuality of the hash not getting included in the Cloud Blockchain it just needs to be calculated and sent again.

The private Blockchain is a single Tendermint instance running on Docker, while the Cloud Blockchain has 4 Tendermint validator node instances that are executed using Docker

Figura 6: Test scenarios overview



Fonte: The author

containers running in different machines. Those machines are globally distributed as in figure 7, being:

- 1) Curitiba, Brazil (the client and the provenance service are running in this machine).
- 2) Frankfurt, Germany
- 3) New Jersey, USA
- 4) Tokyo, Japan

To test the backends under different loads, 3 throughputs were defined: 1 request per second (1 rps), 20 requests per second (20rps) and 40 requests per second (40 rps). Each request consists of adding a document and retrieving it to validate the operation. In the 1 request

Figura 7: Blockchain nodes locations

Fonte: The author

per second, the testing application starts a thread with 1 request every 1 second (whether the previous request finished or not), until it reaches the maximum request number of 1000 requests. In this case, the minimum runtime for this test would be 1000 seconds (around 16.7 minutes). For the same amount of 1000 requests, 20 requests per second takes at least 50 seconds and 40 requests per second at least 25 seconds.

If all of the requests take less than one second, when a new round of tests starts all of the previous requests are finished. Otherwise they form a bottleneck and the number of waiting/blocked requests will increase. Taking the 40 requests per second as an example, in second 0 the test executes 40 requests. Considering a case where 20 threads finish in second 0, then in second 1 the next 40 requests are executed, meaning that now there are 60 requests waiting for response, and so on until the 1000 requests limit (1000r) is reached and the application will just wait for everything to finish.

To simulate network errors, random packets were dropped during execution. The idea behind those dropped packets is to simulate devices that may be in a location with an unstable network connection, or a Cloud with bad connectivity. The first round of tests were executed without errors, and after that the scenarios were run with network packet drop rates of 20%, 40%, 60% and 80%. The packages were dropped using iptables[2] tool. The base command line used to drop the package was:

```
1 iptables -I INPUT -p tcp --dport $PORT -m statistic --mode random --
    probability $LOSS -j DROP
```

Listing 4.1: iptables statistic drop command

Each one of the combinations (backend and requests per second) was executed 10 times in a round robin fashion, meaning that each combination was invoked 10.000 times. The test execution combinatorial formula is presented below, with each category explained on table 1:

$$10 * ((Cloud + Fog) * (1rps + 20rps + 40rps) * (0\% + 20\% + 40\% + 60\% + 80\%) * (faultyserver) * 1000r)$$

Tabela 1: Formula components

Formula component	Description
Cloud / Fog	the type of environment being tested
1 - 40rps	the number of Requests per second
0 - 80%	Packet dropping percentage
faulty server	Packets being dropped at the Service or at the Blockchain
1000r	1000 requests

Fonte: The author

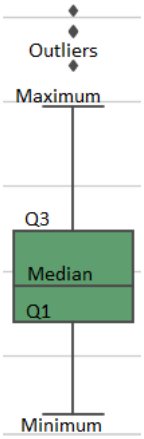
The main machine running the client, server and one Blockchain node has 12 cores and 16GB of RAM. The other 3 nodes are dual core CPUs with 2 GB of RAM, all of them dedicated to the tests. Before every test execution, all of the data was removed and a cool down period was performed to ensure that the system load of one test would not affect the next. To make sure that the machine was not overloaded the system load was monitored during the execution and all machines were under a small load.

[2]https://git.netfilter.org/iptables/, visited on 2020-08-28

4.3 RESULTS

This section presents the results of the tests separated by test cases. The majority of the results are illustrated using Box Plot diagrams. Box plots are useful as they provide a visual summary of the data and makes easy to quickly identify mean values, the dispersion of the data set, outliers and signs of skewness. It consists of two parts: a box and a set of whiskers. Figure 8 demonstrates the parts of a Box Plot: the lowest line is the minimum of the data set and the highest line is the maximum of the data set. The box is drawn from the first quartile (Q1 - 25% of the data) to third quartile (Q3 - 75% of the data) with a horizontal line drawn in the middle to denote the median. The points outside of the minimum and maximum are values that are numerically distant from the rest of the data, and are called outliers.

Figura 8: Parts of a Box Plot



Fonte: The author

4.3.1 BASELINE TESTS

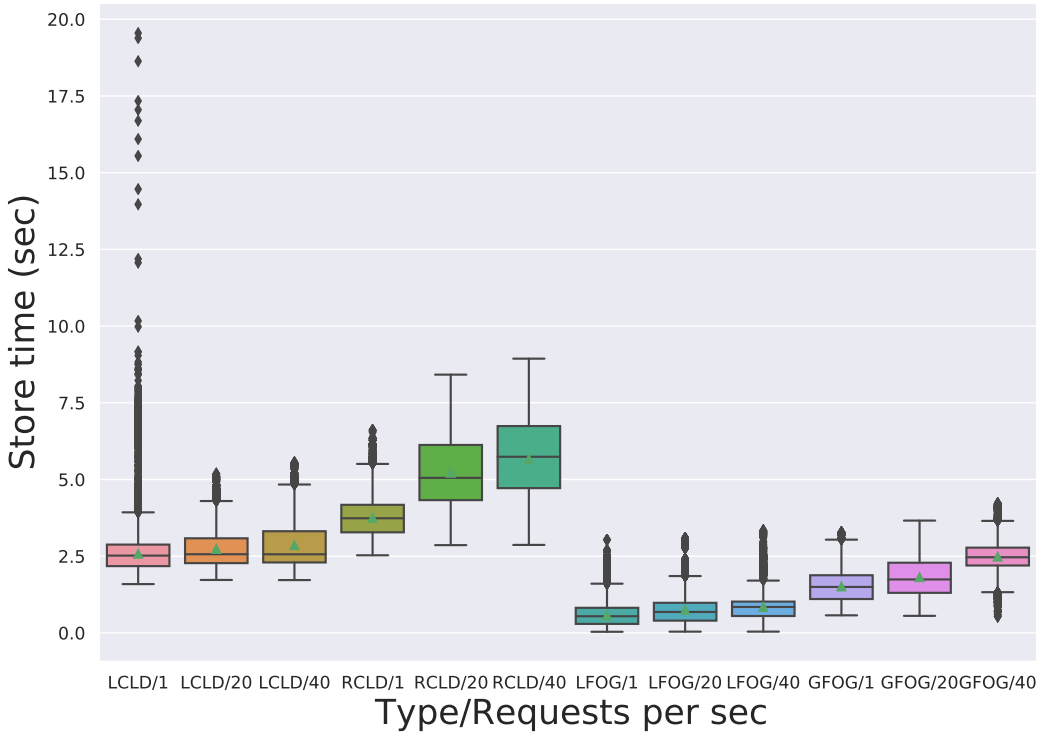
The first test compares the results of:

- LCLD - a local Cloud with 4 nodes
- RCLD - a Cloud distributed globally
- LFOG - the Fog environment with the remote nodes running locally
- GFOG - the Fog environment with the remote nodes distributed globally

No errors were simulated in those tests. The results on the Provenance service side show that using the synchronous option and having a single Tendermint node in the private

Blockchain substantially increased the response speed. In figure 9 the times for the Cloud are slower than the times for the Fog model. This can be explained by the amount of time it takes to reach consensus among the 4 Blockchain nodes when compared to a single node. However, when looking at the retrieval times, they are almost the same as seen in figure 10.

Figura 9: Blockchain storage time when no errors are being simulated



Fonte: The author

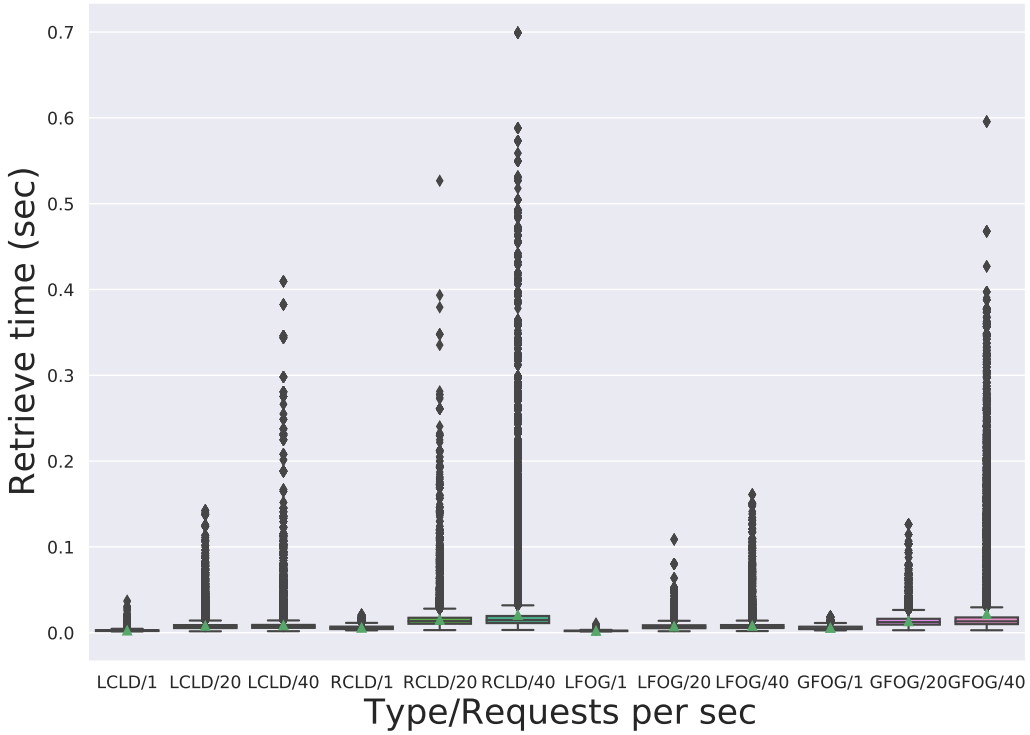
As the Cloud scenario takes more time to store data, the number of blocked threads on the client side was significantly higher than the number of blocked threads for Fog scenario as seen in figure 11. This means that clients using our architecture would spend less time blocked waiting for the provenance service to confirm that the information was stored successfully than an architecture that stores everything on a shared Blockchain.

For the remainder of the tests, only the Real Cloud and Real Fog settings were executed.

4.3.2 FOG EDGE CLIENTS WITH NETWORK ISSUES

In this scenario, the Fog test environment was executed with random errors being raised between the edge and the Fog provenance server as illustrated by the red section with

Figura 10: Blockchain retrieval time when no errors are being simulated



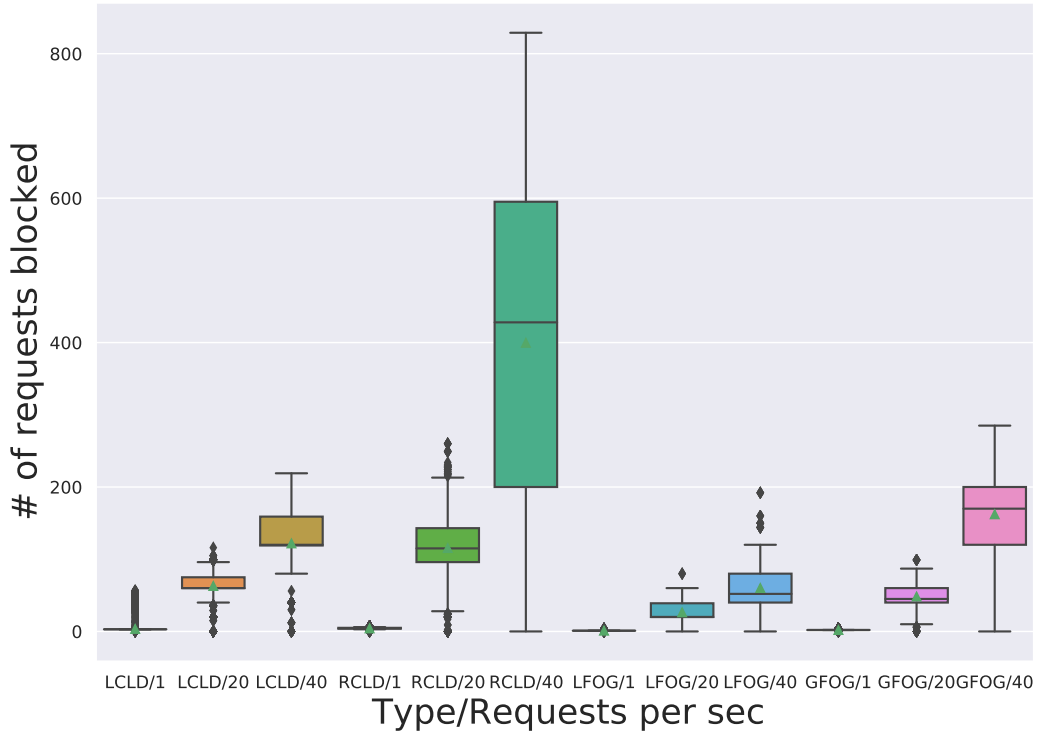
Fonte: The author

an exclamation mark in figure 12.

The number of blocked requests on the client side was similar the the number of blocked requests from the test without error as seen in figure 13. There was, however, one main difference: as the simulated packet loss percentage raises, more requests fail. One interesting information is that when the number of dropped packets rises from 60% to 80% the number of blocked requests rises faster when compared to the rise from 40% to 60% drop. In table 2 this information is highlighted in blue and yellow: when the packet drop rate is at 60% the error percentage of client requests is around 9%. But when packet drop rate goes to 80% this number raises to around 73%. This implies that the mean of requests blocked raises proportionally.

Figures 14 and 15 analyze the communication between the service and the Blockchain to store and retrieve the information on the server side. The response times are at a level similar to the 1 request per second when there are no errors. When the drop rate in the service increases, fewer requests are completing thus fewer requests are reaching the blockchain and the communication gets faster.

Figura 11: Comparison of number of Client Blocked Threads when no errors are being simulated



Fonte: The author

Tabela 2: Dropped packets X Request Errors when Edge clients are facing network issues for Fog environment

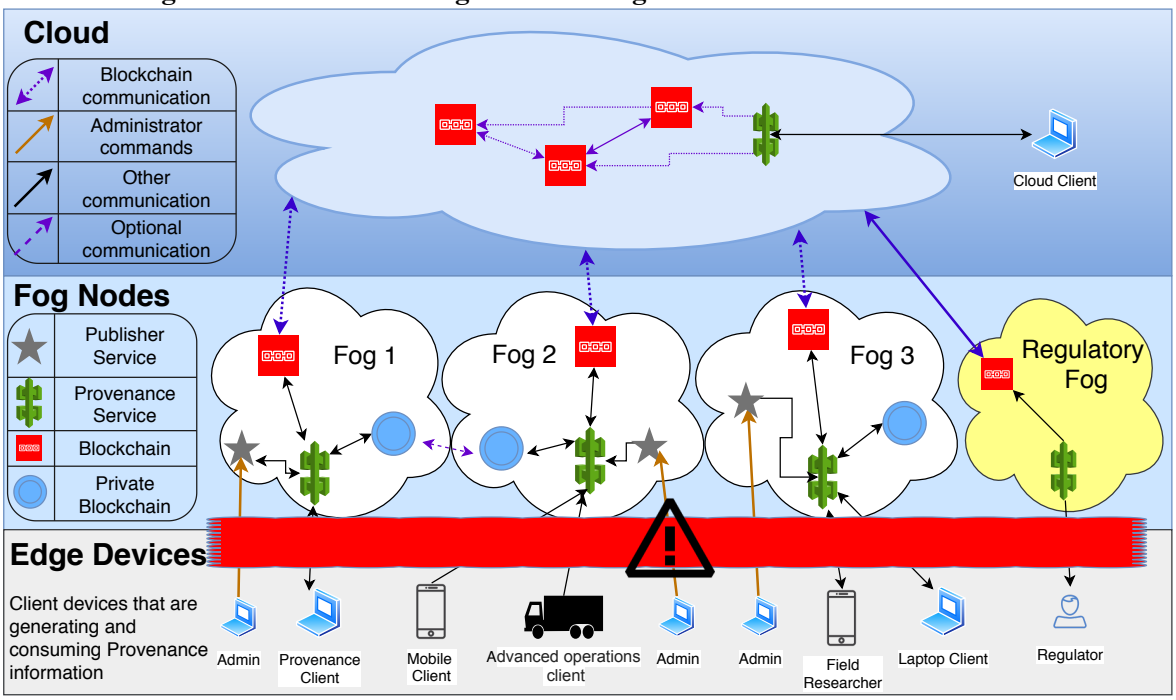
Request per second	Packet Loss	Blocked requests mean	Error percentage
1	40%	7.82	0.5871%
1	60%	46.39	9.3565%
1	80%	269.17	73.0049%
20	40%	44.05	0.59%
20	60%	64.56	9.58%
20	80%	192.64	74.6498%
40	40%	49.17	0.57%
40	60%	61.41	9.2%
40	80%	185.63	73.7956%

Fonte: The author

4.3.3 CLOUD BLOCKCHAIN NODE HAVING NETWORK ISSUES

In this scenario we do not have the Fog node private Blockchain and all information sent by the client is stored directly in the Cloud Blockchain having random errors raised between

Figura 12: Location of Fog clients having network issues on the architecture



Fonte: The author

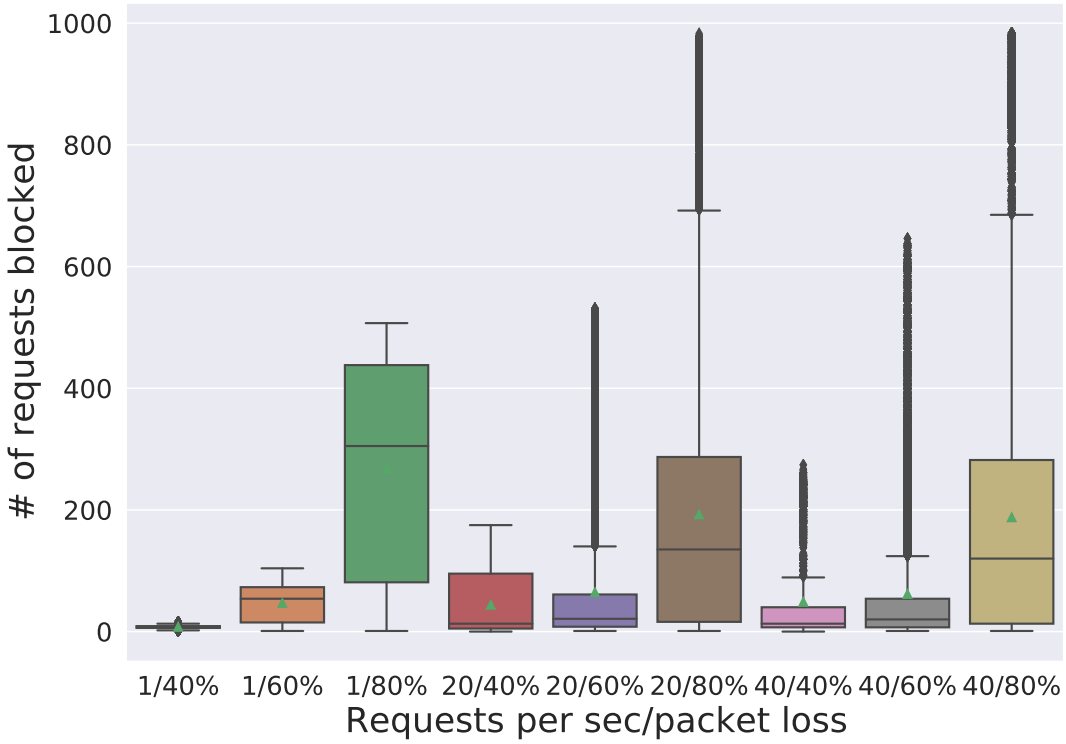
the Blockchain in the provenance server and the other Blockchain validators in the Cloud, as denoted by the red section with an exclamation mark in figure 16. Those errors were simulated by dropping packets in the Tendermint communication port between Blockchains. As stated before, this communication is done in a synchronized way so the clients will wait until the data is written to the Blockchain.

The dropped packets played a significant role in this test as they affected Tendermint consensus protocol and triggered interesting results. Table 3 shows the number of dropped packets and the number of client requests that failed. The tests were executed only with 1 and 20 requests per second (with 20 requests per second all the requests failed).

The total failures occurred because Tendermint protocol classifies the network problems of the Blockchain node as 'misbehave'[3] and punishes the node by removing it from the validators pool for a calculated period of time. As the requests sent by the provenance service are synchronous, when the node is removed from the validator pool it returns a failure. This happens because the call can only return successfully if the node is able to commit the information to the Blockchain.

[3]https://docs.tendermint.com/master/spec/consensus/light-client/accountability.html, visited on 2020-07-17

Figura 13: Number of Client Blocked Requests when Edge clients are facing network issues for Fog environment



Fonte: The author

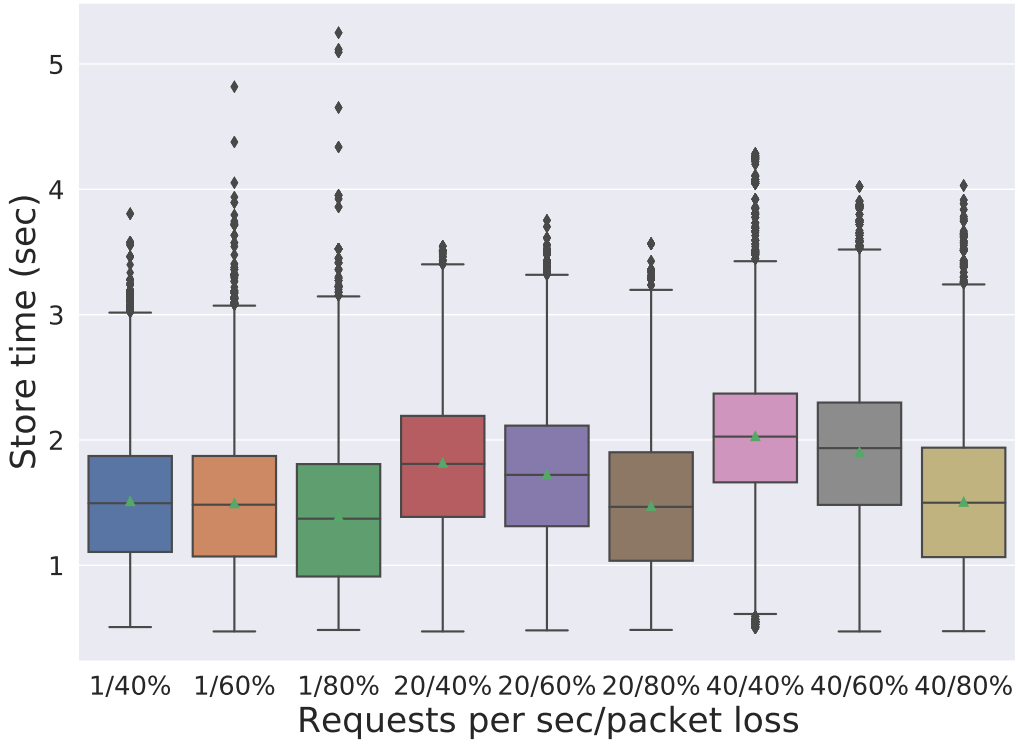
Tabela 3: Request failures for Cloud Blockchain node having network issues

Requests per second	Drop rate	Total Requests	Total requests failures
1	20%	1000	4
1	40%	1000	645
1	60%	1000	1000
1	80%	1000	1000
20	20%	1000	1000
20	40%	1000	1000
20	60%	1000	1000
20	80%	1000	1000

Fonte: The author

Figure 17 shows the number of blocked requests for this case. The requests remain blocked while the Blockchain is accepted as a validator, but when it is removed from the pool they fail and the next requests fail immediately. For the tests with no data shown, the requests were not counted as blocked because all of them failed right after being executed without blocking.

Figura 14: Server storage when client has connection errors for Fog environment



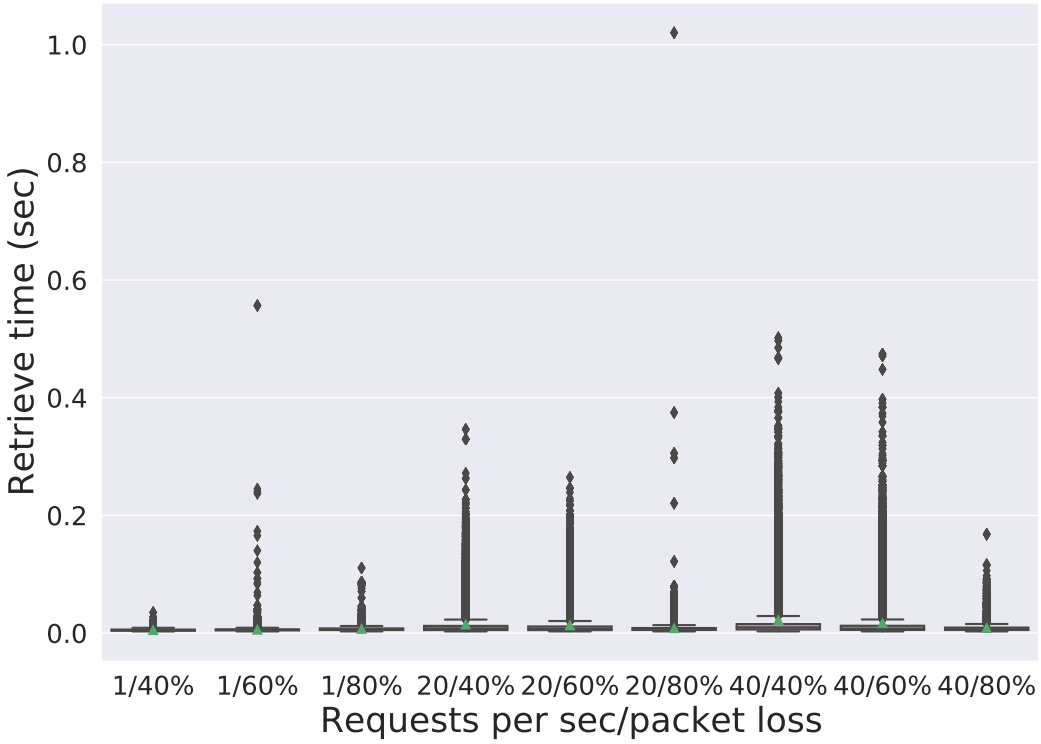
Fonte: The author

4.3.4 FOG NODE/BLOCKCHAIN NODE HAVING NETWORK ISSUES

In this scenario the Fog node has errors being raised between the Blockchain instance in the Fog node and the other Cloud Blockchain nodes, as illustrated by the red section with an exclamation mark in figure 18. In this test the clients do not suffer any problems as the data is stored in the Fog node private Blockchain and the hash is sent asynchronously to the Cloud. The Cloud suffers the same problem as the Cloud only client, but the main difference of the asynchronous client is that it does not return errors: in the asynchronous option Tendermint stores the requests locally and when it manages to negotiate the block it flushes everything. As the data is stored in the local private Blockchain without errors, the numbers are basically the same ones as when no errors are being simulated as shown in figure 19 for blocked threads, store time in figure 20 and retrieval in figure 21.

Further details about Tendermint error handling behaviour can be seen on appendix D.

Figura 15: Server retrieval when client has connection errors for Fog environment



Fonte: The author

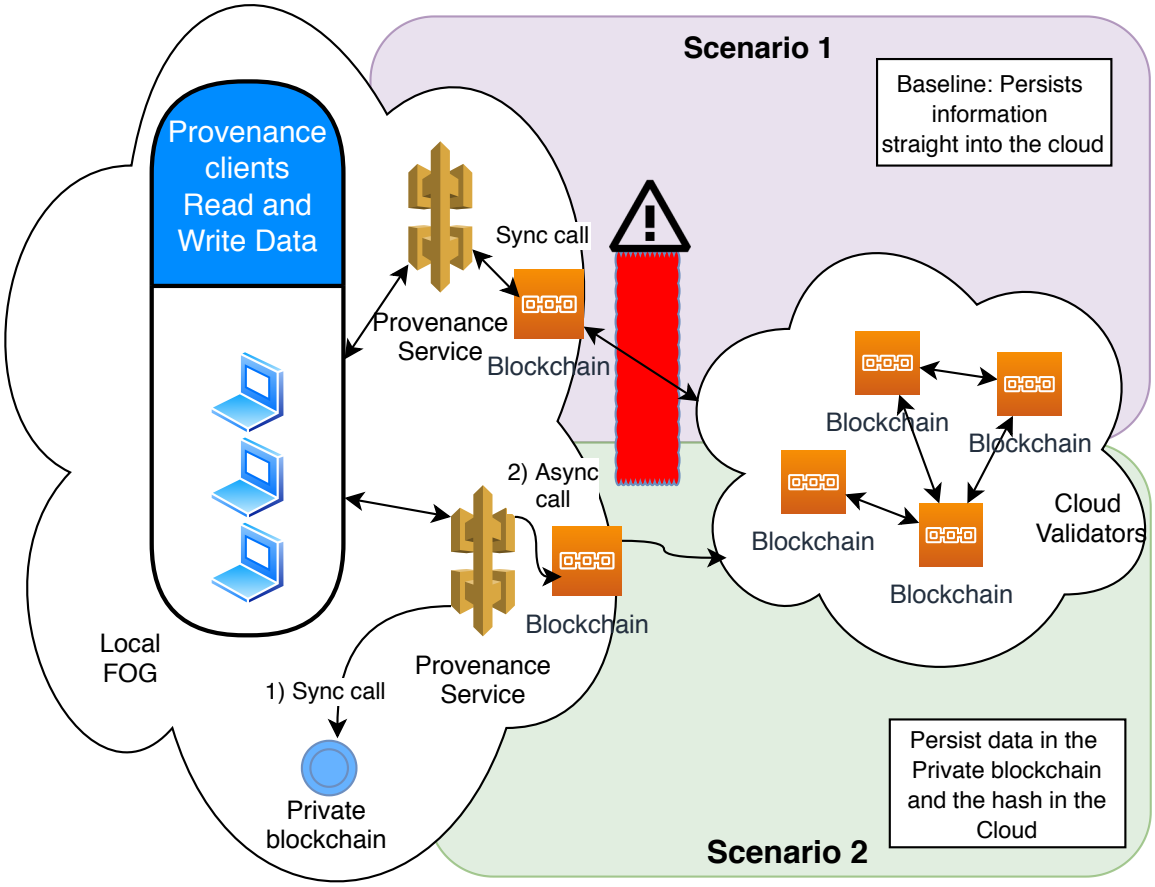
4.4 TEST RESULTS REVIEW

The tests executions demonstrated that Fog architecture is faster and more reliable than directly storing information straight into the Cloud Blockchain, mainly due to the time taken by the Blockchain protocol to reach consensus.

When network errors were simulated between the clients and the provenance service, TCP/IP protocol handled packet loss and it was demonstrated how the number of failures changed as the packet loss percentage increased, being this a common behaviour in both architectures.

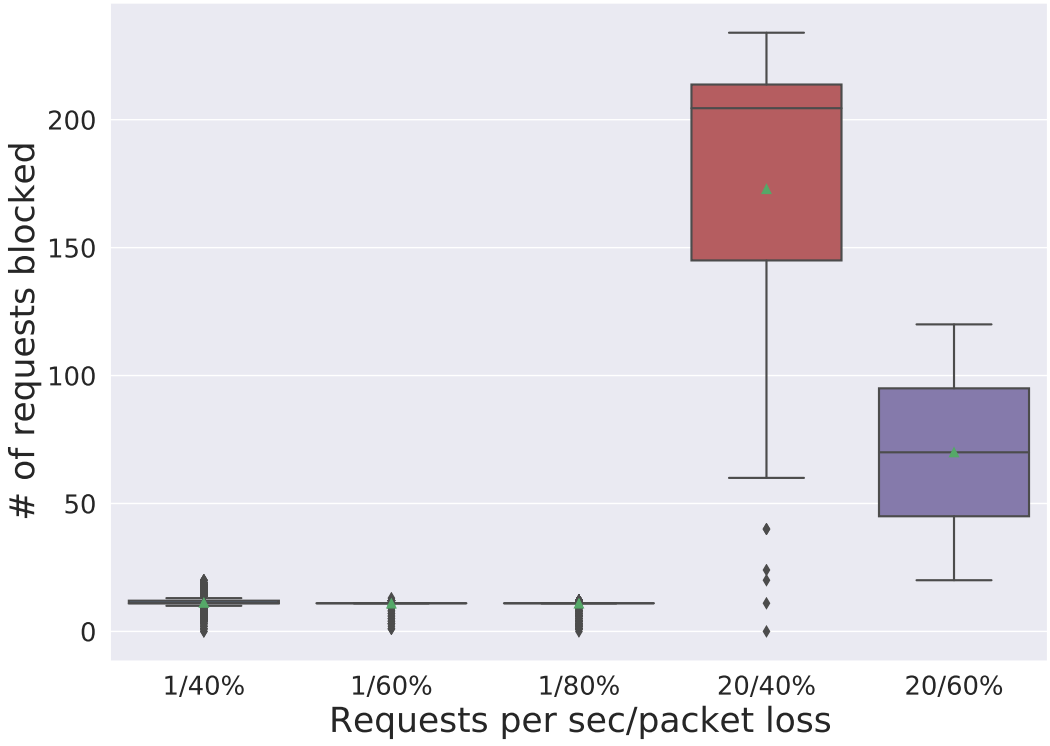
Finally, when errors were raised in the communication with the Cloud the main advantage of the Fog architecture appeared: while the usual approach of storing information directly into the Cloud Blockchain failed to complete even under light loads, Fog architecture managed to keep the same experience to the users as when no failures were simulated.

Figura 16: Location of Cloud blockchain network issues on the test architecture



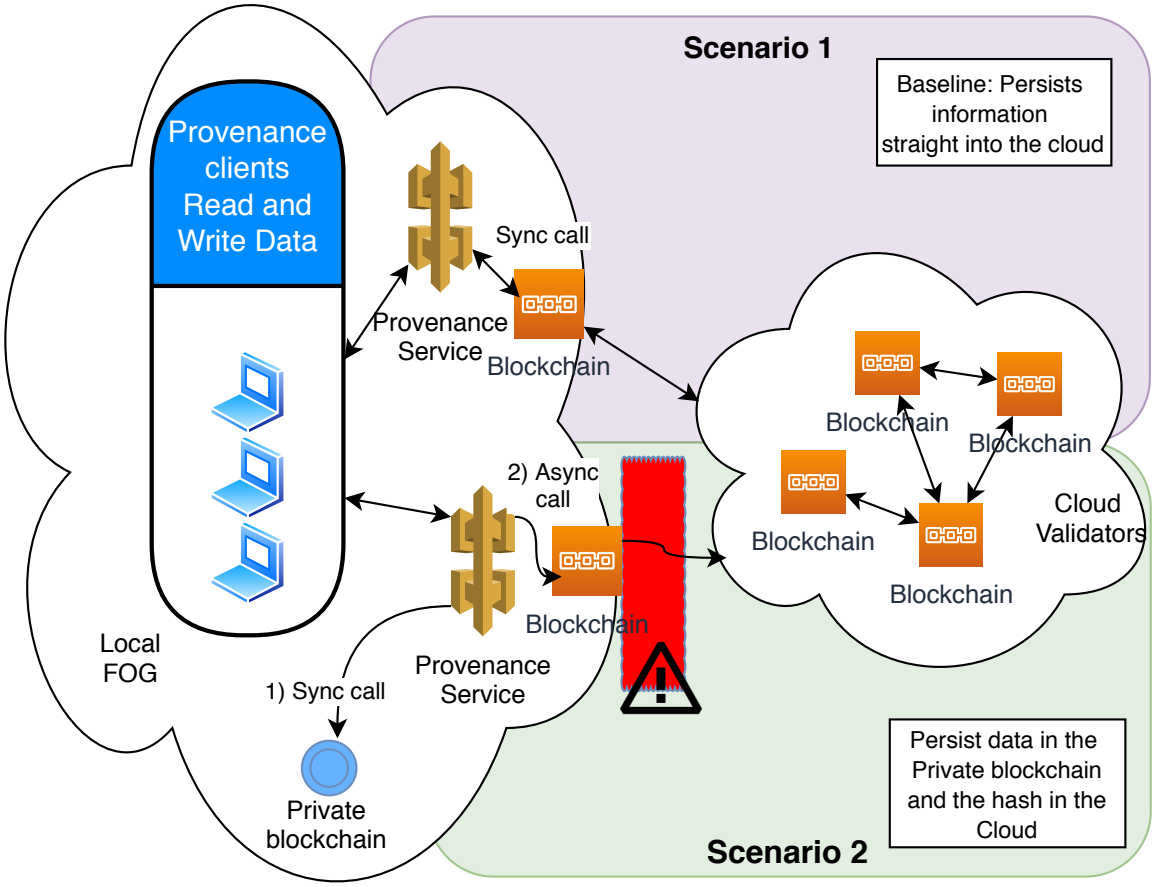
Fonte: The author

Figura 17: Number of Client Blocked Requests when Cloud Blockchain is having network errors



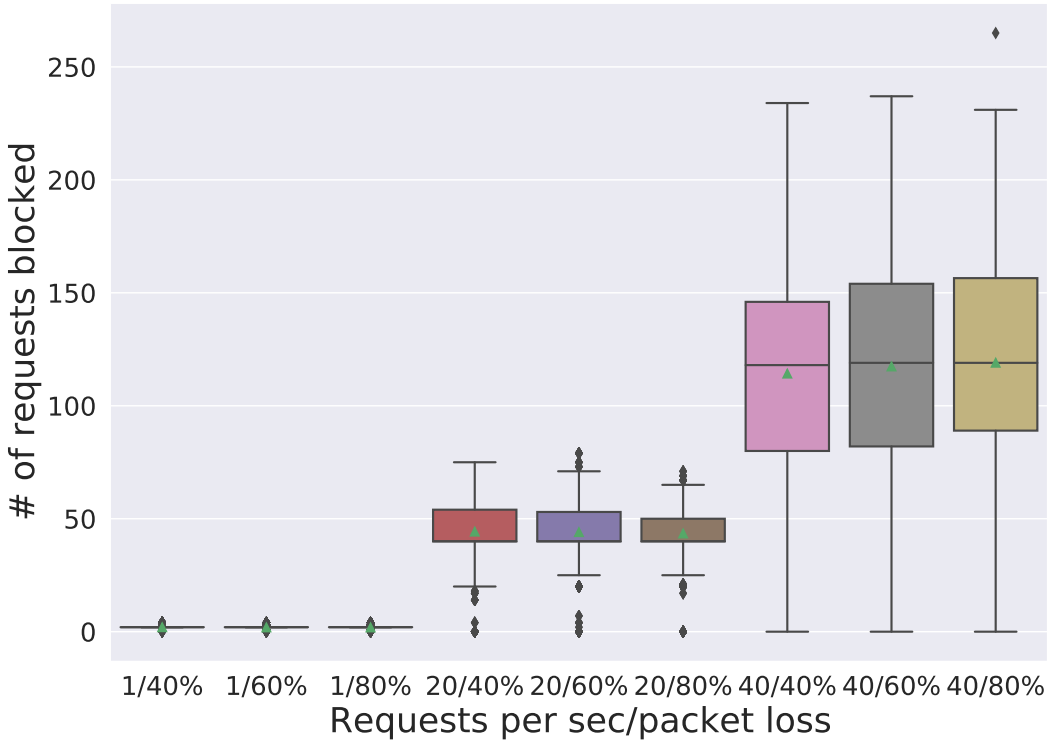
Fonte: The author

Figura 18: Location of Fog blockchain network issues on the test architecture



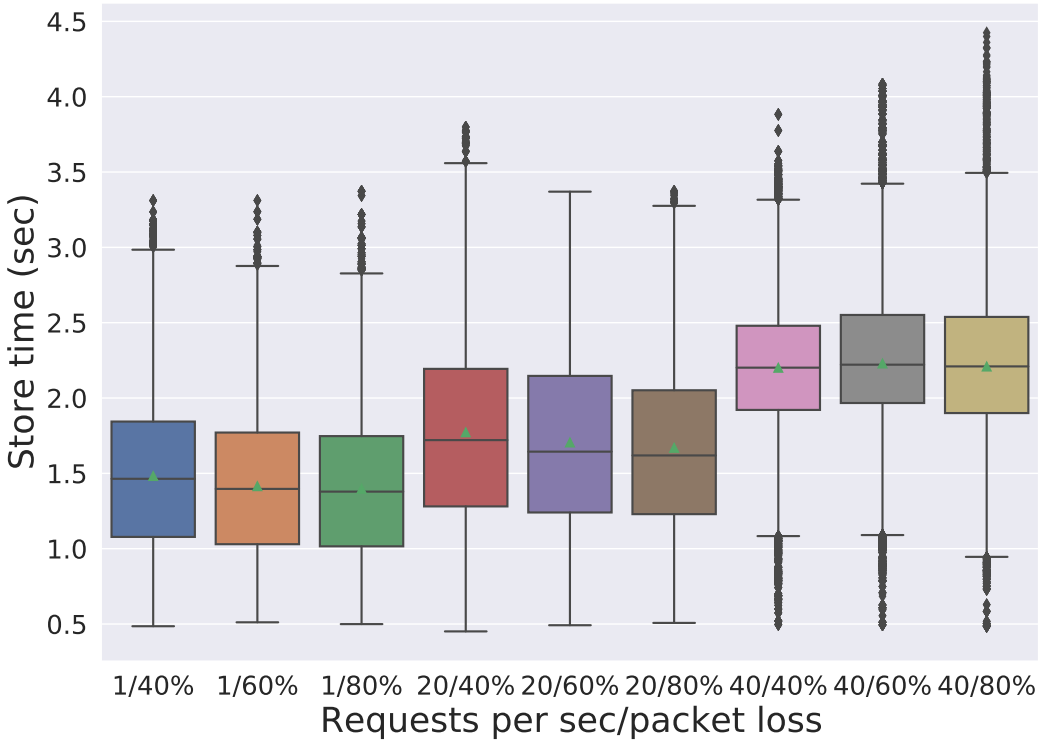
Fonte: The author

Figura 19: Number of Client Blocked Requests when Fog node Blockchain node is having network errors



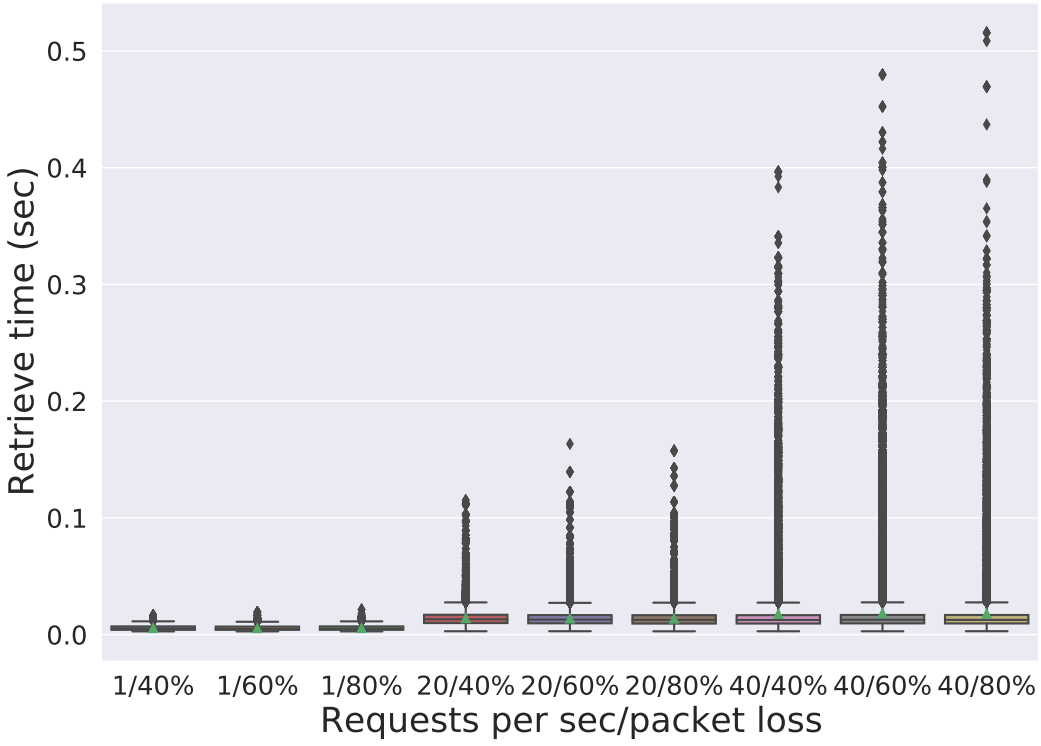
Fonte: The author

Figura 20: Server storage time when Fog node Blockchain is having network errors



Fonte: The author

Figura 21: Server retrieval time when Fog node Blockchain is having network errors



Fonte: The author

5 CONCLUSION

Provenance can be applied to many areas of computing and information systems. The popularity of pervasive and distributed applications and the ever-increasing need for real time information sharing with low latency have introduced a huge demand for resources provisioning closer to the network edge. Provenance in Cloud/Fog/Edge computing infrastructures is a relevant topic for modern applications, including not only the real time oriented ones, but also those focused on hard to reach areas.

This dissertation reviewed the requirements of a Provenance System by using Blockchain to securely store and share information among different Fog nodes. An architecture that can be used to manage provenance information alternating between private and public data has many applications, with disease outbreak surveillance highlighted as an use case. Our proposed architecture employs services to orchestrate Fog computing nodes, allowing the implementation of W3C Prov standard with the advantages enabled by the integration of Fog computing and Blockchain.

Several tests were performed to demonstrate the practicability of the architecture. The introduction of the Fog layer improves performance when compared to an architecture that stores information directly in a distributed Blockchain. This demonstrates that Fog infrastructure is important in reducing communication latency with the edge devices and minimizing issues when communication with other Fog nodes is unstable. The use of Blockchain itself minimized the complexity of sharing data between Fog nodes, and the consensus protocol based on Proof of Stake handled all the network issues that were simulated, meeting performance expectations.

Network errors were simulated and monitored to understand how the Blockchain behaves and how it mitigates network issues. As future work a complete network disconnection could be explored and the Fog nodes could be prepared to handle full network disconnection, providing Provenance services autonomously until they are back online. This kind of feature would be useful in disaster situations or field research in remote areas where no network is

available or the aggregated value of having a connection is too expensive. Besides network problems, Bizantine failures may occur and were not explored in this dissertation. Future work could cover situations were nodes can not be trusted and are behaving in malicious ways.

5.1 PUBLICATIONS

The first results of the research compares Blockchain and Database performance and was accepted at SBRC 2020 WBlockchain Workshop:

Lautert, Filipe and Pigatto, Daniel F. and Gomes-Jr, Luiz. Blockchain-based Data Provenance. In: XXXVIII Simpósio Brasileiro de Redes de Computação (SBRC) - Wblockchain Workshop, 2020

Building upon the first paper and now employing a Fog computing architecture, a second paper was published at IEEE ISCC 2020 8th Workshop on Communications in Critical Embedded Systems (WoCCES):

Lautert, Filipe and Pigatto, Daniel F. and Gomes-Jr, Luiz. A fog architecture for privacy-preserving data provenance using blockchains. In: IEEE ISCC - 8th Workshop on Communications in Critical Embedded Systems (WoCCES), 2020

REFERÊNCIAS

ARUNKUMAR, S.; MUPPIDI, S. **Secure your blockchain solutions**. [S.l.], 2019. Disponível em: <https://developer.ibm.com/technologies/blockchain/articles/how-to-secure-blockchain-solutions/>.

AZARIA, A. et al. Medrec: Using blockchain for medical data access and permission management. In: IEEE. **2016 2nd International Conference on Open and Big Data (OBD)**. [S.l.], 2016. p. 25–30.

BAUER, S.; SCHRECKLING, D. Data provenance in the internet of things. In: **EU Project COMPOSE, Conference Seminar**. [S.l.: s.n.], 2013.

BELHAJJAME, K. et al. Prov-dm: The prov data model. **W3C Recommendation**, 2013.

BUNEMAN, P.; KHANNA, S.; WANG-CHIEW, T. Why and where: A characterization of data provenance. In: SPRINGER. **International conference on database theory**. [S.l.], 2001. p. 316–330.

CISCO, R. C. R. f. P. R. **Fog Computing, Ecosystem, Architecture and Applications, Cisco**. [S.l.], 2014. Disponível em: <http://research.cisco.com/researchHASHrfp-2013078>.

FEIGENBAUM, G.; REIST, I.; REIST, I. J. **Provenance: An alternate history of art**. [S.l.]: Getty Publications, 2012.

GADELHA, L. M. et al. Towards a threat model for provenance in e-science. In: SPRINGER. **International Provenance and Annotation Workshop**. [S.l.], 2010. p. 277–279.

GAETANI, E. et al. Blockchain-based database to ensure data integrity in cloud computing environments. 2017.

GERVAIS, A. et al. On the security and performance of proof of work blockchains. In: **Proceedings of the 2016 ACM SIGSAC conference on computer and communications security**. [S.l.: s.n.], 2016. p. 3–16.

GIL, Y. et al. Prov model primer. **W3C Working Group Note**, w3c, 2013.

GREENSPAN, G. **Four genuine blockchain use cases**. [S.l.], 2016. v. 10. Disponível em: <https://www.multichain.com/blog/2016/05/four-genuine-blockchain-use-cases>.

GROTH, P.; MOREAU, L. Prov-overview. an overview of the prov family of documents. w3c working group note note-prov-overview-20130430. **World Wide Web Consortium**, 2013.

HITZLER, P. et al. Owl 2 web ontology language primer. **W3C recommendation**, World Wide Web Consortium (W3C), v. 27, n. 1, p. 123, 2009.

HUYNH, T. D. et al. The prov-json serialization. World Wide Web Consortium, 2013.

HUYNH, T. D.; MOREAU, L. Provstore: a public provenance repository. In: **5th International Provenance and Annotation Workshop (IPAW'14)**. [s.n.], 2014. Disponível em: <https://eprints.soton.ac.uk/365509/>.

JANG, S.-H. et al. Fog computing architecture based blockchain for industrial iot. In: SPRINGER. **International Conference on Computational Science**. [S.l.], 2019. p. 593–606.

KAMATH, R. Food traceability on blockchain: Walmart's pork and mango pilots with ibm. **The Journal of the British Blockchain Association**, The British Blockchain Association, v. 1, n. 1, p. 3712, 2018.

KAUR, K. et al. Blockchain-based lightweight authentication mechanism for vehicular fog infrastructure. In: IEEE. **2019 IEEE International Conference on Communications Workshops (ICC Workshops)**. [S.l.], 2019. p. 1–6.

KWON, J. Tendermint: Consensus without mining. **Draft v. 0.6, fall**, v. 1, p. 11, 2014.

LAUTERT, F.; PIGATTO, D. F.; GOMES-JR., L. Blockchain-based data provenance. SBRC Wblockchain Workshop, 2020.

LAUTERT, F.; PIGATTO, D. F.; GOMES-JR., L. A fog architecture for privacy-preserving data provenance using blockchains. IEEE ISCC - 8th Workshop on Communications in Critical Embedded Systems (WoCCES), 2020.

LIANG, X. et al. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In: IEEE PRESS. **Proceedings of the 17th IEEE/ACM international symposium on cluster, cloud and grid computing**. [S.l.], 2017. p. 468–477.

MARÍN-TORDERA, E. et al. Do we all really know what a fog node is? current trends towards an open definition. **Computer Communications**, Elsevier, v. 109, p. 117–130, 2017.

MCCONAGHY, T. et al. Bigchaindb: a scalable blockchain database. **white paper, BigChainDB**, 2016.

Mukherjee, M. et al. Security and privacy in fog computing: Challenges. **IEEE Access**, v. 5, p. 19293–19304, 2017.

PEARSON, D. **Provenance research in book history: a handbook**. [S.l.]: British library London, 1994.

PÉREZ, B.; RUBIO, J.; SÁENZ-ADÁN, C. A systematic review of provenance systems. **Knowledge and Information Systems**, Springer, v. 57, n. 3, p. 495–543, 2018.

POSTON, H. **Threat Modeling for the Blockchain**. [S.l.], 2019. Disponível em: <https://www.howardposton.com/blog/threat-modeling-for-the-blockchain>.

ROMEU, J. L. Data quality and pedigree. **Material Ease**, 1999.

SHARMA, P. K.; CHEN, M.-Y.; PARK, J. H. A software defined fog node based distributed blockchain cloud architecture for iot. **Ieee Access**, IEEE, v. 6, p. 115–124, 2017.

SIMMHAN, Y. L.; PLALE, B.; GANNON, D. A survey of data provenance techniques. **Computer Science Department, Indiana University, Bloomington IN**, v. 47405, p. 69, 2005.

Stojmenovic, I.; Wen, S. The fog computing paradigm: Scenarios and security issues. In: **2014 Federated Conference on Computer Science and Information Systems**. [S.l.: s.n.], 2014. p. 1–8.

STOLL, C.; KLAAßEN, L.; GALLERSDöRFER, U. The carbon footprint of bitcoin. **Joule**, v. 3, p. 1647–1661, 2019.

Suhail, S. et al. Introducing secure provenance in iot: Requirements and challenges. In: **2016 International Workshop on Secure Internet of Things (SIoT)**. [S.l.: s.n.], 2016. p. 39–46.

TAN, Y. S.; KO, R. K.; HOLMES, G. Security and data accountability in distributed systems: A provenance survey. In: IEEE. **2013 IEEE 10th International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing**. [S.l.], 2013. p. 1571–1578.

WALSH, L.; AKHMECHET, V.; GLUKHOVSKY, M. Rethinkdb-rethinking database storage. **Hexagram 49, Inc.**, 2009.

WEIBEL, S. L.; KOCH, T. The dublin core metadata initiative. **D-lib magazine**, v. 6, n. 12, p. 1082–9873, 2000.

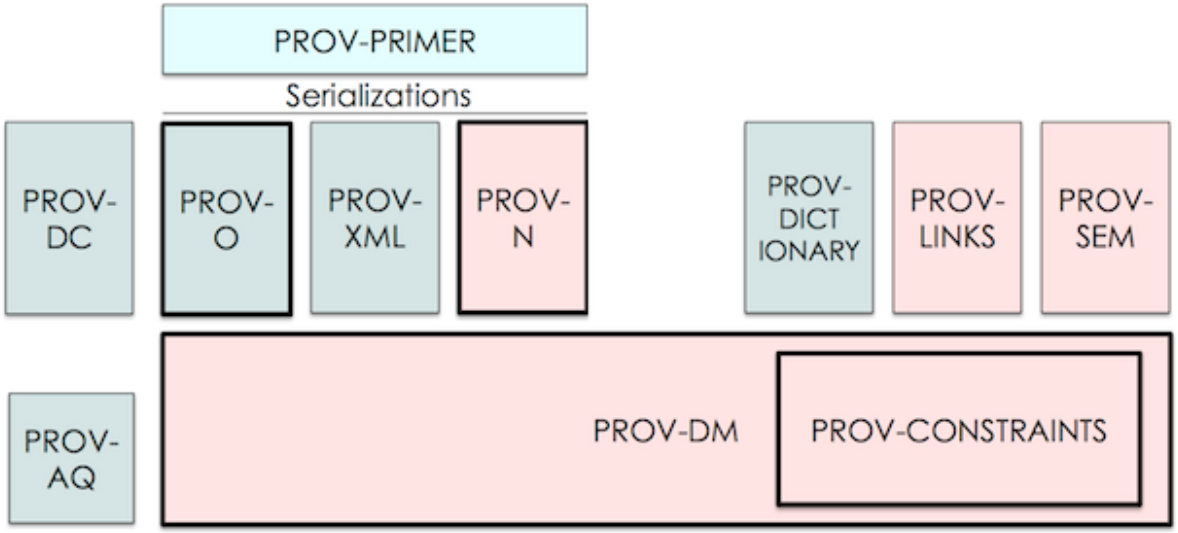
APÊNDICE A – W3C PROV STANDARD

Simmhan et al. (2005) uses the characteristic of how provenance is represented as one of the keys to classify the systems. Groth e Moreau (2013) as part of the W3C defines a set of documents called the “Prov Family” that was developed after an extensive research including use case cataloging, requirements elicitation and a literature survey, which the last version was released in 30 April 2013. The PROV Documentation is a set of 12 documents that aims to defines a model and related operations to enable the inter-operable interchange of provenance information that can be used as a reference to design provenance representation, especially in heterogeneous environments such as the Web. Those documents are based on 8 recommendations that a provenance framework should support, being them:

- the core concepts of identifying an object, attributing the object to person or entity, and representing processing steps;
- accessing provenance-related information expressed in other standards;
- accessing provenance;
- the provenance of provenance;
- reproducibility;
- versioning;
- representing procedures;
- representing derivation.

The documents are organized as show in figure 22. The colors represents the target audience, being light blue for Users that needs to understand PROV, green for Developers that will develop applications to create and consume provenance, and red for Advanced audience that may create validators, new serializations or other advanced features.

Figura 22: The organization of prov documents by use



Fonte: Groth e Moreau (2013)

Huynh et al. (2013) submitted a last document to the Committee specifying the Json serialization for prov documents, but that work item has not been added to the Working Group's original charter and the group ended without including this serialization. Besides not being in the standard it is supported by provenance frameworks such as Python provit[1] and Java ProvToolbox[2], to cite examples.

This dissertation reviews the "Users" and "Developers" documents as the functionalities included in them are the recommended to have a Provenance system working. It is important to note that most of those documents implementations is already covered in frameworks, so this is not being implemented again but just reviewed for the sake of understanding. The referred documents are:

- PROV Model Primer (PROV-PRIMER) is the entry point to PROV offering an introduction to the provenance data model
- PROV-O defines a light-weight OWL2 ontology for the provenance data model (PROV-DM). This is intended for the Linked Data and Semantic Web community.
- PROV-XML defines an XML schema for the provenance data model (PROV-DM). This is intended for developers who need a native XML serialization of the PROV data model.

[1] `https://pypi.org/project/provit/`, visited on 2020-01-12
[2] `https://lucmoreau.github.io/ProvToolbox/`, visited on 2020-01-12

- PROV-AQ Provenance Access and Query (PROV-AQ) specifies headers, queries and other extensions to HTTP protocol that are aimed to retrieve provenance information of a HTTP resource. It is an interesting idea but this pattern is not seen around web servers and demands metadata configuration that requires application programming.

- PROV-DC defines a mapping between Dublin Core and PROV-O. The Dublin Core Metadata Initiative (DCMI) (WEIBEL; KOCH, 2000) is a core metadata vocabulary for simple and generic resource descriptions.

- Provenance Dictionary (PROV-Dictionary) describes extensions to facilitate the modeling of provenance for dictionaries – logical structures consisting of key-entity pairs.

In the next sections PROV basic concepts and entities are explained in further details.

A.1 PROV MODEL PRIMER

PROV-PRIMER is an introduction to PROV and explains many concepts that will be used and revised on the other documents. At figure 23 it illustrates the most important concepts of PROV and how they relate to each other:

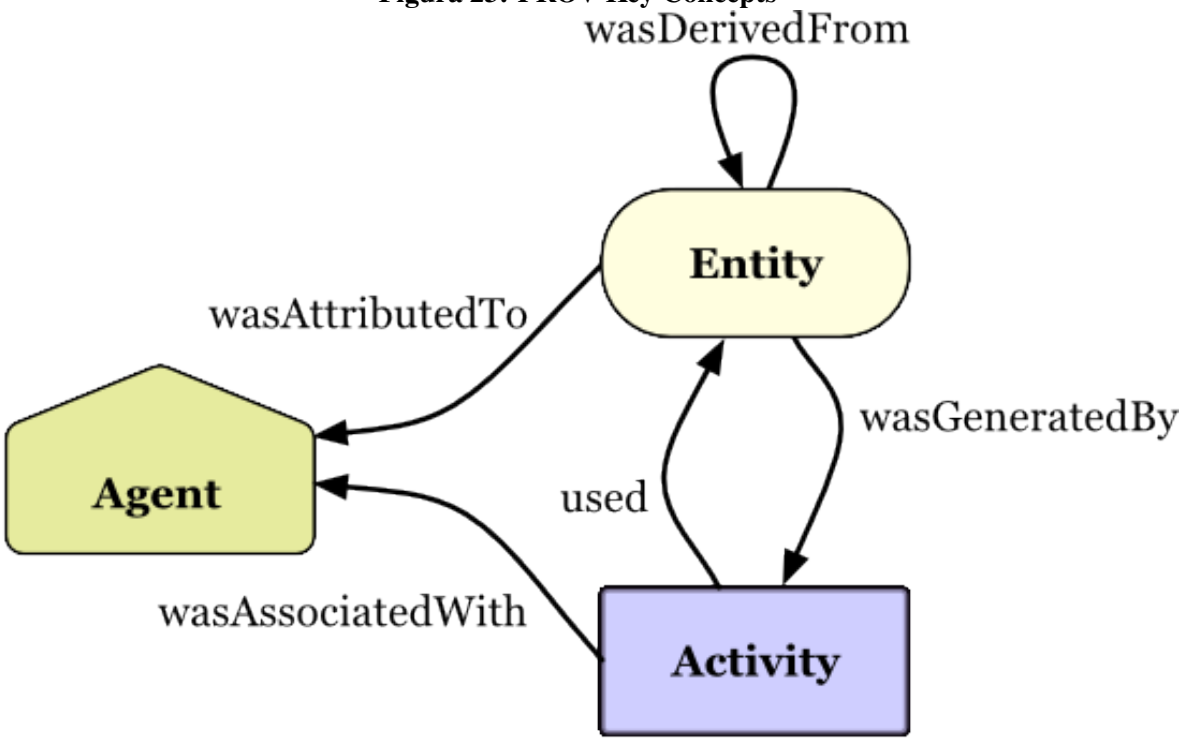
- **Entities**: the objects that are the target of provenance are entities. In the case of "Data Provenance", "Data" would be the entity. Entities can be related to each other: for instance a document points to a reference entity. And entities can have many attributes such as version, location, author, etc

- **Activities**: explains how entities are created and how their attributes were changed, becoming new entities. They represent actions or processes applied to an entity, for instance: entity X was generated by a review of entity Y, then the review is the activity. An activity may use more than one entity. In the previous example, the review of entity X may use references from document Z to perform the action.

- **Agents**: one or more agents can be associated with entities and/or activities, and those associations are labelled as roles or responsibilities. As in the previous example, the review of Document X was associated with an agent that is the reviewer, while the original document was attributed to it's author. An agent can be anything that can have responsibility for the activity or entity, being it a person, a software, an organization, or anything else that can be responsible.

The document also briefs other elements such as plans, time, entities specializations and more advanced concepts not covered in this appendix.

Figura 23: PROV Key Concepts



Fonte: Belhajjame et al. (2013)

A.1.1 PROV DATA MODEL AND REPRESENTATIONS

Belhajjame et al. (2013) specifies the PROV Data Model (PROV-DM) as the conceptual data model for the PROV specification. It is an extensive document that is split in 6 components covering from the basics of creating entities and activities to the most complex tasks as creating bundles to represent provenance of provenance and structuring it all together. As it is a conceptual model, other documents specify how to implement it.

PROV-O (PROV Ontology) expresses PROV-DM using the OWL2 Web Ontology Language (another W3C standard defined by Hitzler et al. (2009)), whereas PROV-XML provides a XML schema representing it, allowing instances of the PROV data model to be serialized in XML. In the same way, PROV-JSON provides a JSON representation for the PROV Data Model.

To illustrate how those notations represents the same information, consider the following example: a document named “History of provenance” is composed of a dataset (dataset1), a region list (regionList) and an illustration (illustration1) created based on the

dataset and region list. Listing A.1 shows this example in Ontology notation, listing A.2 in XML notations and listing A.3 using Json format.

In the context of this work, information can be extracted from those notations to generate metadata in the blockchain. For instance, a document can reference the block that the source documents are stored, this way speeding up the document linkage and search.

```
# Entities
exn:article      a prov:Entity ;
                 dcterms:title "History of provenance" .
exg:dataset1     a prov:Entity .
exc:regionList   a prov:Entity .
exc:illustration a prov:Entity .

# Activities
exc:illustrate1 a prov:Activity .

# Usage and Generation
exc:illustrate1  prov:used           exg:dataset1 ;
                 prov:used           exc:regionList .
exc:illustration prov:wasGeneratedBy exc:illustrate1 .
```

Listing A.1: PROV-O notation

```
<prov:document xmlns:prov="http://www.w3.org/ns/prov#" xmlns:ex="http://
    example.org/" xmlns:dct="http://purl.org/dc/terms/" xmlns:foaf="http://
    xmlns.com/foaf/0.1/">
    <!-- Entities -->
    <prov:entity prov:id="exn:article">
        <dct:title>History of provenance</dct:title>
    </prov:entity>
    <prov:entity prov:id="exg:dataset1"/>
    <prov:entity prov:id="exc:regionList"/>
    <prov:entity prov:id="exc:illustration1"/>
    <!-- Activities -->
    <prov:activity prov:id="exc:illustrate1"/>
    <!-- Usage and Generation -->
    <prov:used>
        <prov:activity prov:ref="exc:illustrate1"/>
        <prov:entity prov:ref="ex:dataset1"/>
    </prov:used>
```

```
    <prov:used>
        <prov:activity prov:ref="exc:illustrate1"/>
        <prov:entity prov:ref="exc:regionList"/>
    </prov:used>
    <prov:wasGeneratedBy>
        <prov:entity prov:ref="exc:illustration1"/>
        <prov:activity prov:ref="exc:illustrate1"/>
    </prov:wasGeneratedBy>
</prov:document>
```

Listing A.2: PROV-XML notation

```
{
    "entity": {
        "exn:article": {
            "dcterms:title" : "History of provenance"
        },
        "exg:dataset1": { },
        "exc:regionList": { },
        "exc:illustration": { }
    },
    "activity": {
        "exc:illustrate1": { }
    },
    "used": {
        "_:u1": {
          "prov:activity": "illustrate1",
          "prov:entity": "dataset1"
        },
        "_:u2": {
          "prov:activity": "illustrate1",
          "prov:entity": "regionList"
        }
    },
    "wasGeneratedBy": {
        "_:wGB1": {
            "prov:activity": "exc:illustrate1",
            "prov:entity": "exc:illustration"
```

```
27                }
28          }
29  }
```

Listing A.3: PROV-Json notation

A.2 PROV IMPLEMENTATIONS

Many libraries and tools have been implemented to validate, generate and handle PROC documents as available on the PROV Implementation Report[3], so this specification has been used to represent provenance and generate the documents utilized in this work.

[3] `https://www.w3.org/TR/prov-implementations/`, visited on 2020-04-13

APÊNDICE B – PROVSTORE API

The REST endpoint provided by ProvStore has 8 methods, most of them protected with Oauth authentication. They are:

- **GET /store/api/v0/documents/** - Retrieve a list of public documents and other documents visible to the user if authenticated. It has a number of parameters as pagination options (offset, limit), ordering and filters by metadata fields: created_at, document_name, id, owner, public, views_count.

- **GET /store/api/v0/documents/:id/** - method that retrieves information about the document by his id. A Json representation of the metadata returned by ProvStore call "GET /store/api/v0/documents/1/" is shown in listing B.1. The listing method will return a similar response, just adding the pagination information.

- **GET /store/api/v0/documents/:id(/flattened)(/view/:view).:format** - Retrieve the raw body of the document with the given ID in the requested format. The formats supported by ProvStore are Json, PROV ontology (provn), Svg, Trig, Ttl and XML . If "/flattened" is added to the request it will bring all bundled (associated) documents in the response. A sample request to method "GET /store/api/v0/documents/1.json" is illustrated in listing B.2: it is possible to see that it is a PROV document in Json format as described in appendix A.1.1 and that the metadata returned by the previous GET methods is not obtained from the document but it is a ProvStore metadata.

- **POST /store/api/v0/documents/** - Store documents for authenticated users. It receives a document name and the document in PROV Json format or an url to the actual document, returning some information about the processing, the store time and the generated id as show in listing B.3.

- **GET /store/api/v0/documents/:id/bundles/** - List the bundles of a document. It is similar to the document listing method, the main diference is that it just list documents associated with the provided document ID.

- **POST /store/api/v0/documents/:id/bundles/** - Similar to the store document method, but this one adds a document as a bundle to document identified by :id. Authentication is required.

- **DELETE /store/api/v0/documents/:id/** - Delete the document associated with this id. After this command the documetn can not be retrieved anymore. Authentication is required.

- **GET /store/api/v0/documents/:doc_id/bundles/:bundle_id(.:format)** - View a specific bundle of a document. It is similar to the retrieval of raw body of a document.

```
{
    "created_at": "2013-08-14T13:10:34.636600",
    "document_name": "first document",
    "id": 1,
    "public": false,
    "resource_uri": "/store/api/v0/documents/1/",
    "url": "",
    "views_count": 15
}
```

Listing B.1: ProvStore Json Metadata

```
{
    "wasDerivedFrom": {
        "_:id2": {
            "prov:usedEntity": "hg:Overview.html",
            "prov:generatedEntity": "w3:WD-prov-dm-20111215",
            "prov:usage": "ex:recordu",
            "prov:generation": "ex:recordg",
            "prov:activity": "ex:rcp"
        }
    },
    "wasAssociatedWith": {
        "_:id3": {
            "prov:agent": "ex:webmaster",
            "prov:activity": "ex:rcp"
        }
```

```
16      },
17      "used": {
18          "ex:recordu": {
19              "prov:entity": "hg:Overview.html", "prov:activity":
                    "ex:rcp"
20          }
21      },
22      "entity": {
23          "w3:WD-prov-dm-20111215": {
24              "prov:type": {
25                  "type": "python:utf8", "$": "html4"
26              }
27          },
28          "hg:Overview.html": {
29              "prov:type": {
30                  "type": "python:utf8", "$": "file in hg"
31              }
32          }
33      },
34      "prefix": {
35          "rec": "http://example.org/record",
36          "w3": "http://www.w3.org/TR/2011/",
37          "hg": "http://dvcs.w3.org/hg/prov/raw-file/9628aaff6e20
              /model/releases/WD-prov-dm-20111215/",
38          "ex": "http://example.org/"
39      },
40      "activity": {
41          "ex:rcp": {
42              "prov:type": {
43                  "type": "python:utf8",
44                  "$": "copy directory"
45              }
46          }
47      },
48      "wasGeneratedBy": {
```

```
49          "ex:recordg": {
50              "prov:entity": "w3:WD-prov-dm-20111215",
51              "prov:activity": "ex:rcp"
52          }
53      }
54  }
```

Listing B.2: ProvStore Json Raw Response

```
1
2   {
3       "content": {
4           the document just stored
5       },
6       "created_at": "2013-08-19T14:54:23.614795",
7       "document_name": "myDocument",
8       "id": 33,
9       "public": true,
10      "rec_id": "myDocument",
11      "resource_uri": "/store/api/v0/documents/33/",
12      "url": null,
13      "views_count": 0
14  }
```

Listing B.3: ProvStore Json Raw Response

APÊNDICE C – THREAT MODEL REFERENCES

There are various studies about threat models for blockchain and Fog scenarios. To find a case that would cover all the specifics of each one of them would be complicated, but by studying the specific cases it is possible to see that all of them have in common concerns about business processes and IT rules. Provenance, Blockchain and Fog threats were reviewed to better understand how they are being covered.

C.1 PROVENANCE

Data provenance threat models mainly focus on integrity and confidentiality of the data: provenance must not be changed since it's inception and confidential information needs to remain as is. As provenance has many use cases, the models and attacks focus on the underlying structure that stores it to manipulate the information. So for provenance data stored in a database, we would review database attacks, and for Fog and Blockchain the focus needs to be in those two infrastructures.

Gadelha et al. (2010) describe an preliminary work to trace security requirements for provenance systems in the context of e-Science, and propose some security controls to fulfill them. Asymmetric cryptography is used to achieve confidentiality and checksums are used for integrity, while other works referenced by the article uses role-based access control techniques. To secure provenance information a system named Kairos (a security architecture for provenance developed by the authors in another article) is proposed and reviewed.

Suhail et al. (2016) identifies possible ways to integrate secure provenance in IoT based on security issues faced in this technology and other resource constraints. It raises a number of challenges for provenance such as integrity, confidentiality, privacy, access controls and freshness.

C.2 BLOCKCHAIN

AS blockchains usage is on the rise, researchers are worried about their security specially because they need to be trusted over public networks.

Poston (2019) maps blockchain security threats to STRIDE[1], a well-known threat model developed by Microsoft, to create an effective threat model for the blockchain. Each letter in the STRIDE acronym is designed to refer to one of the most common threats in cybersecurity: **S**poofing, **T**ampering, **R**epudiation, **I**nformation **D**isclosure and **E**levated Privileges.

To have a better discussion it splits the blockchain ecosystem into various levels: cryptographic primitives, data structures, protocols (consensus & block creation), infrastructure (nodes and network) and advanced (smart contracts and blockchains extensions). It maps all the possible attacks into a matrix for each one of those components in figure 24.

Arunkumar e Muppidi (2019) proposes a blockchain security reference architecture that can be applied across blockchain projects and solutions for various industry use cases. It highlight some examples were companies lost millions of dollars because they have been hacked by code exploitation, stolen keys or desktop invasions. The blockchain security levels are reviewed (public, private, permissionless and permissioned) and other risks are reviewed such as business and governance, process and technologies. Those risks are reviewed in detail, as for instance for application security: *"The blockchain itself is immutable and tamper-proof, but the applications that leverage the network pose challenges at every level"*.

A Threat model for blockchains is proposed in figure 25. It classifies the threats as:

- **New threat landscape** - threats that are specific to blockchain.
- **Conventional threats that take on new meaning** - conventional threats that gains new meanings with the introduction of blockchain
- **Conventional threat management** - business-as-usual threats that needs to be addressed for any solution.

It also defines Security controls that are unique to blockchain as:

- Treat the underlying infrastructure of the blockchain solution as critical infrastructure.
- Partition and adopt best practices for namespacing to regulate access.

[1] `https://en.wikipedia.org/wiki/STRIDE_(security)`, visited on 2020-06-04

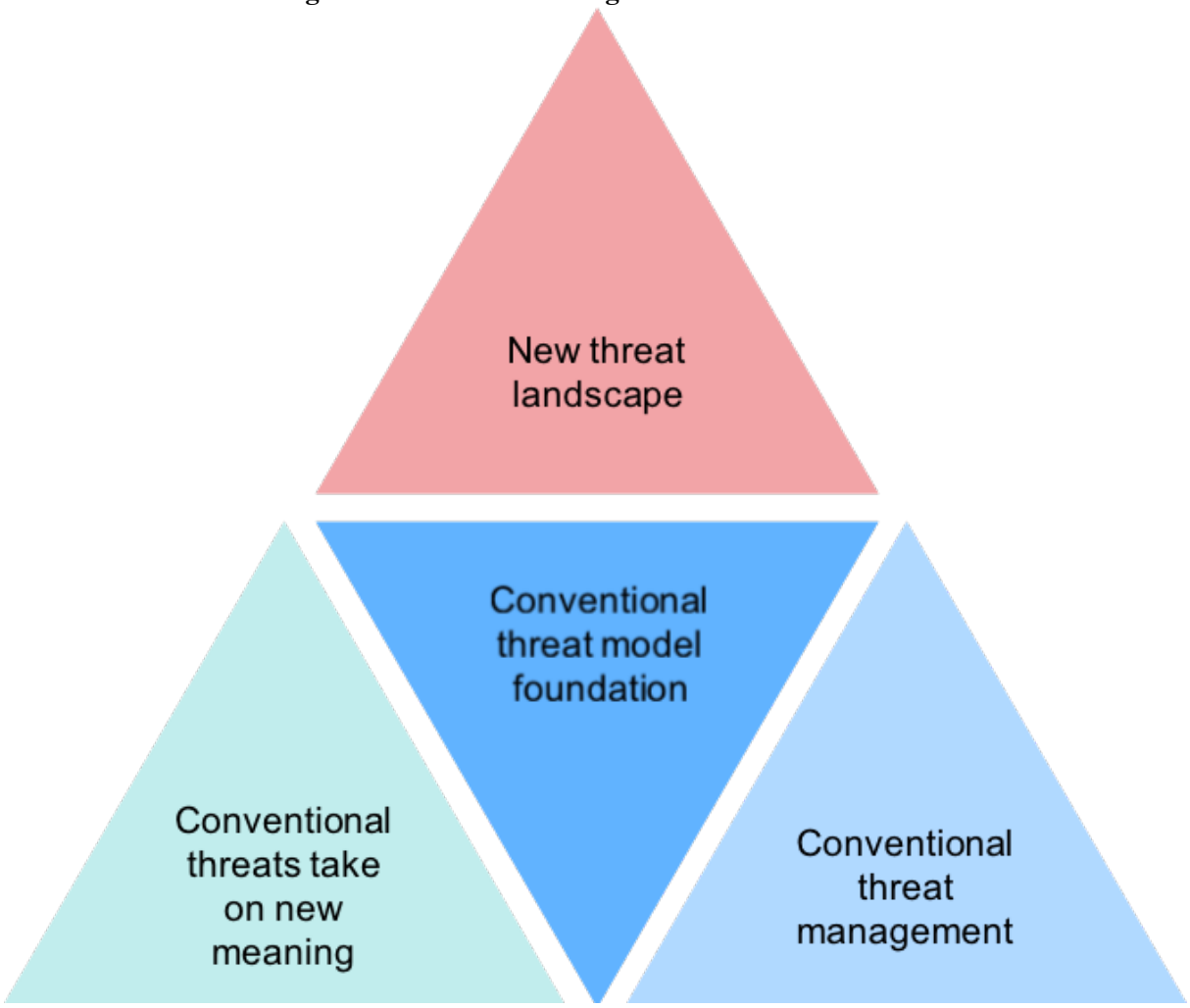
Figura 24: Threat modeling for Blockchain Matrix

		Spoofing	Tampering	Repudiation	Information Disclosure	Denial of Service	Elevated Privileges		
							Account	Blockchain	Smart Contract
Fundamentals	Cryptographic Primitives	Private Key Phishing Shor's Algorithm	Grover's Algorithm		Private Key Shor's Algorithm		Private Key Shor's Algorithm		
	Data Structure		Transaction Malleability						
Protocol	Consensus		51% Long-Range Nothing at Stake	51% Long-Range		51% Artificial Difficulty Increases Long-Range		51% Long-Range Selfish Mining SPV Mining	
	Block Creation		Frontrunning			Transaction Flooding		SPV Mining	
Infrastructure	Nodes	Malware	Malware		Malware	Failure to Update Malware		MSP Misconfig	
	Network		Eclipse/Routing Network Design		Network Design	Eclipse/Routing Network Design Physical Attacks PoS DoS MSP DoS		Eclipse/Routing	
Advanced	Smart Contracts	Delegatecall	Arithmetic Bad Randomness Reentrancy Short Addresses Timestamp Dependence Unchecked Returns			Access Control Out of Gas			Access Control Delegatecall
	Blockchain Extensions	Insecure APIs					Insecure APIs		

Fonte: Poston (2019)

- Define and enforce the appropriate endorsement policies based on business contracts.
- Enforce identity and access controls to access the blockchain solution and data.
- Enforce the hardware security module (HSM).
- Use a privileged access management (PAM) solution for escalated actions.
- Use API security best practices to safeguard API-based transactions.
- Leverage a secrets-store for both application and privileged access.
- Adopt a data classification approach to safeguard data/information.

Figura 25: Threat modeling for Blockchain solution

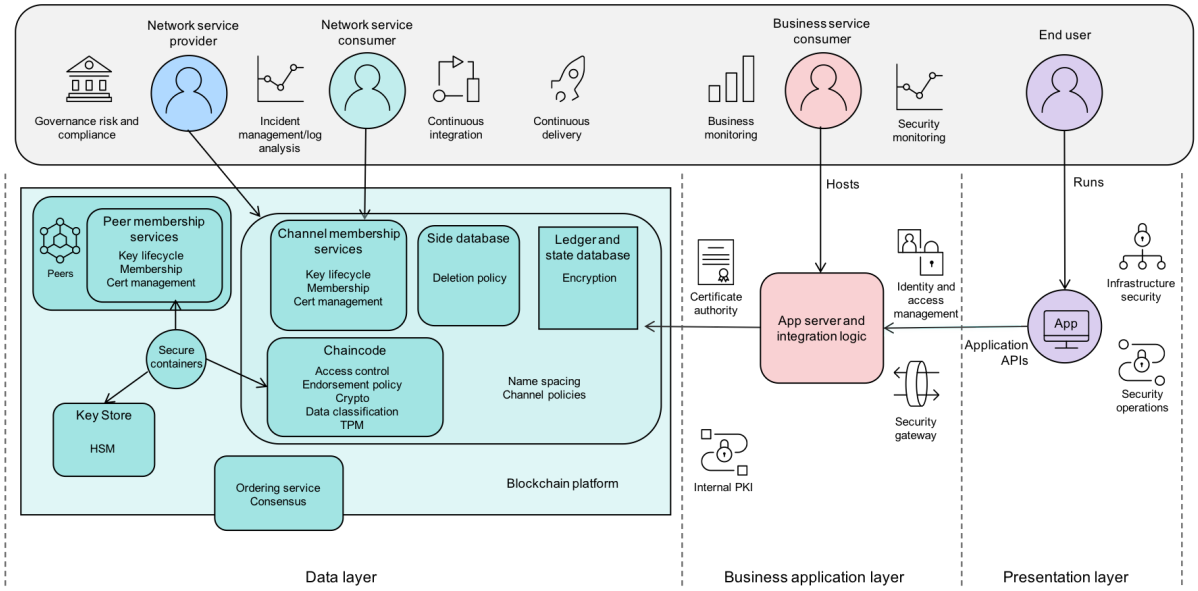


Fonte: Arunkumar e Muppidi (2019)

- Use privacy-preserving technologies for sensitive information.
- Protect applications from vulnerabilities and safeguard data.
- Enforce access control in smart contracts.
- Leverage Trusted Platform Modules (TPMs) for sensitive code execution.
- Secure communications both internally and externally.

Summarizing all security controls for blockchain leads to to the blockchain security reference model illustrated in figure 26 , which can be applied across all blockchain solutions.

Figura 26: Blockchain security reference model



Fonte: Arunkumar e Muppidi (2019)

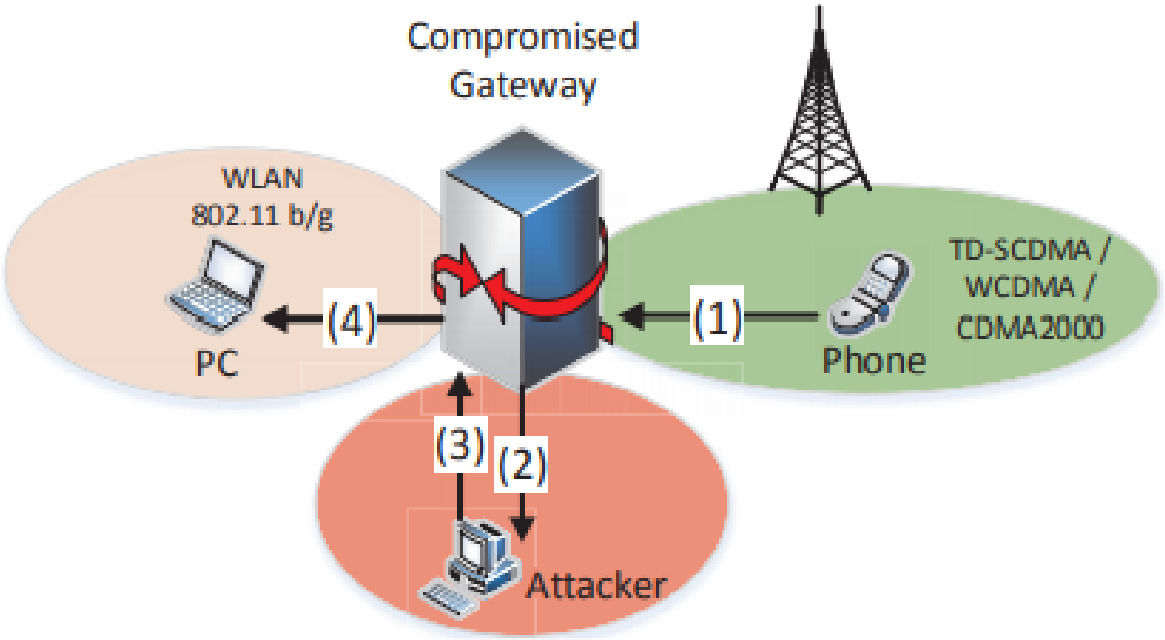
C.3 FOG

As the Fog is a “mini-cloud” away from the Cloud, it may suffer from attacks similar as the Cloud would suffer and it may suffer attacks that would be executed against IoT devices. It is mostly vulnerable to network attacks, and the STRIDE model can also be applied in this case.

Stojmenovic e Wen (2014) highlights that there are many security solutions for Cloud that may not fit well for Fog computing, specially because Fog works at the edge of the network and may face many threats which do not exist in a cloud controlled environment. The work focus in security issues that are created by authentication at different levels of gateways that facilitates Man-in-the-middle attack. It demonstrates how those attacks can be achieved by compromising a Fog gateway which serves the Fog device as illustrated in figure 27. By compromising this gateway the attacker gains access to the communication between the phone (1) and the PC (4).

Mukherjee et al. (2017) review the challenges for security and privacy in Fog Computing and enumerates them as trust, authentication, secure communications, end user privacy and malicious attacks. It reviews other articles to get the conclusion that *“security and privacy issues are well-studied in cloud computing, however, all of them are not suitable for fog computing ”* and *“many new security and privacy threats arise that were not present in*

Figura 27: The hijacked communication in Fog (e.g. from phone to PC)



Fonte: Stojmenovic e Wen (2014)

centrally-managed cloud computing" .

APÊNDICE D – TENDERMINT ERROR HANDLING

When errors were simulated in one of the four Blockchain nodes executing in the Cloud, Tendermint treated it as an erratic node and removed it from consensus.

Listing D.1 shows the end of the log from the Cloud server running in Tokyo during the 40 threads per second execution and failure rate of 80%: from 23:26:15 to 23:27:08 no blocks were generated due to connection errors. But at the end of the test when the iptables rule was dropped all the remaining 1892 transactions were sent (highlighted in green). As a comparison, listing D.2 show the results using 1 request per second with a 20% failure rate: in this case the blocks are generated almost every two seconds (15:53:23, 15:53:26, 15:53:28, 15:53:30) while for listing D.1 they stopped being generated.

For the sake of completeness, Tendermint has a parameter called "Mempool.size"that defines the maximum number of transactions that it can keep on the local Tendermint node without synchronizing with the other blocks. By default it is 5.000, but it was overflowing in the 40 threads per second case, so it was raised to 10.000 to allocate all the pending messages.

```
1 I[2020-07-15|23:26:15.945] Executed block   module=state height=3137
      validTxs=0 invalidTxs=0
2 I[2020-07-15|23:26:15.948] Committed state module=state height=3137 txs=0
      appHash=AABC020000000000
3 I[2020-07-15|23:27:08.562] Executed block   module=state height=3138
      validTxs=11 invalidTxs=0
4 I[2020-07-15|23:27:08.565] Committed state module=state height=3138 txs=11
      appHash=C0BC020000000000
5 'I[2020-07-15|23:27:15.628] Executed block   module=state height=3139
      validTxs=1892 invalidTxs=0   '
6 'I[2020-07-15|23:27:15.649] Committed state module=state height=3139 txs
      =1892 appHash=88DA020000000000    '
7 I[2020-07-15|23:27:17.168] Executed block   module=state height=3140
      validTxs=0 invalidTxs=0
8 I[2020-07-15|23:27:17.171] Committed state module=state height=3140 txs=0
      appHash=88DA020000000000
```

Listing D.1: Tendermint logs

```
1 I[2020-07-16|15:53:23.032] Executed block module=state height=17014
      validTxs=6 invalidTxs=0
2 I[2020-07-16|15:53:23.034] Committed state module=state height=17014 txs=6
      appHash=D6D91E0000000000
3 I[2020-07-16|15:53:26.682] Executed block module=state height=17015
      validTxs=4 invalidTxs=0
4 I[2020-07-16|15:53:26.684] Committed state module=state height=17015 txs=4
      appHash=DED91E0000000000
5 I[2020-07-16|15:53:28.230] Executed block module=state height=17016
      validTxs=8 invalidTxs=0
6 I[2020-07-16|15:53:28.232] Committed state module=state height=17016 txs=8
      appHash=EED91E0000000000
7 I[2020-07-16|15:53:30.012] Executed block module=state height=17017
      validTxs=0 invalidTxs=0
8 I[2020-07-16|15:53:30.014] Committed state module=state height=17017 txs=0
      appHash=EED91E0000000000
```

Listing D.2: Tendermint logs with good connection