AuditChain: Blockchain views applied to auditing

Afonso Marques
Instituto Superior Tecnico, Lisbon
afonso.m.marques@tecnico.ulisboa.pt

ABSTRACT

Blockchain, an emerging technology, provides a decentralized, and immutable data storage, replicated across untrusting peers. Contrary to common knowledge, private blockchain stakeholders can have different views on the same distributed ledger, and thus several challenges are raised. This happens because as soon as complexity increases, different representations are possible and this way different business process views are created on the same blockchain. Currently, it is hard to visualize data partitions within a blockchain and perform arbitrary operations on it, for instance merging two blockchain views to conduct a blockchain migration or audit. In this document, we will explain the concept of view and how it can be used for several purposes. We also present the entire process required to obtain the data and subsequently create the blockchain view. We will likewise identify the possible mechanisms and the tool created *BUNGEE*, for this purpose. Subsequently, to the creation of the blockchain view, various stakeholders can prove its internal state, contributing to blockchain interoperability, and enhanced the way audits are performed. This thesis aims to contribute to blockchain interoperability and easing of blockchain audits by leveraging the concept of blockchain view.

KEYWORDS

Blockchain audit, Blockchain View, Interoperability, BUNGEE, AuditView

1 INTRODUCTION

In recent years the use of blockchain technology has increased exponentially, a similar thing has happened with the development of new tools and ways to use it, however, when it comes to the auditing aspect, the same does not happen. Several studies have been done on the advantages and importance of blockchain auditing, although there are not many tools available to do auditing, and those that do exist, many are just theoretical solutions and the rest are very limited in terms of functionalities, being only useful when the subject is Cryptocurrencies. This area of blockchain auditing is a recent topic, however it has already sparked interest from quite a few eager investors to be the first to invest in a pioneering solution.

In this document we will introduce what is blockchain technology and its types, we will also introduce the concept of DLT View. We are going to explain what auditing is, its use cases and also blockchain auditing. We will identify the current problem and a possible pioneering solution for it. We will present all the implementation decisions regarding the tools developed. We will also talk about the reason that led us to use certain technologies over others. It will be presented how each one of them was evaluated and the tests used on each.

Throughout the document we will refer to DLT View several times, however it will be presented as View for simplicity.

2 BACKGROUND AND RELATED WORK

2.1 Blockchain

Blockchain is a technology firstly introduced in 2008 by a person or a group with the pseudonymous *Satoshi Nakamoto* after the realise of bitcoin whitepaper[1]. This technology has (slowly) become one of the most frequently discussed methods for securing data storage and transfer through decentralized, trustless, peer-to-peer systems[16]. The blockchain technology provides a decentralized, open, Byzantine fault tolerance transaction mechanism, and promises to become the infrastructure for a new generation of Internet interaction, including anonymous online payments, remittance, and transaction of digital assets [5].

The idea behind this technology is relatively simple. Blockchain technology is constituted by sequential blocks linked together, thus forming a chain. Each block contains a specific group of information (records), constituted by a timestamp and a hash of the previous block in the chain. The first block is called the genesis block, it differs from the others, since it has no block before and thus no reference.

Hash is a cryptographic digest, produced by a hash function. This function has some important characteristics such as, one way (non-invertible), strong collision resistance (computationally infeasible to find two inputs that give the same hash). These characteristics are important since we validate the integrity of the blocks using the hash. The validation process ensures that the records are valid and therefore the block. After this step, the block will be added if the participants reach a consensus. Consensus is reached by participants of the network by a consensus algorithm.

Participants also called miners compete in order to generate the next accepted block. This competition consists of solving a computationally heavy mathematical problem in the most efficient way possible. During this competition, in case of more than one miner solve the hash puzzle at the same time, the blockchain may "fork" leading to parallel chains. Such a scenario is eventually resolved by the miners picking the longest chain [10, 14]. This competitions is associated with Prof-Of-Work (POW) which is the consensus mechanism for Bitcoin, the first application of the blockchain technology [12]. After the miners finish their tasks (generate the next block), they are rewarded, this reward depends on the blockchain [5]. The problem difficulty varies during the time, it is adjusted in order to keep the block generation pace constant, e.g. More participants means more miners competing for the next block, so more blocks would be generated per unit of time, to avoid this, the difficulty of the problem is increased. Naturally, the opposite happens with fewer participants.

Blockchain was plenty of advantages compared with a centralized technology, however is not untouchable. There are a few

[1]https://bitcoin.org/bitcoin.pdf

known attacks, such as: Distributed denial of service (*DDoS*), *51%-Attack*, *Race attack*, *Sybil attack*, among many others.

The first attack, *DDoS* is hard to perform on the blockchain network because of their nature, however is not impossible. When this attack is performed, the intention is to bring down a server by consuming all its processing resources with numerous requests. DDoS attackers aim to disconnect a network's mining pools, e-wallets, crypto exchanges, and other financial services. A blockchain can also be hacked with DDoS at its application layer using DDoS botnets.

An attack scenario against the consensus mechanism is called the *"51% attack."* In this scenario, a group of miners, controlling a majority (51%) of the total hash power of the network, conspire to attack. With the ability to mine most blocks, attacking miners can spawn deliberate bifurcations in Blockchain, generate double-spend transactions, or perform denial of service attacks (DoS) against specific addresses or transactions. A bifurcation attack or double-spend attack is an attack where the attacker causes already confirmed blocks to be invalidated by bifurcating a level below them, with a later reconvergence in an alternate chain. With enough power, an attacker can invalidate six or more blocks in a sequence, invalidating transactions that were previously considered immutable (with six acknowledgments). Note that double spending can only be done on the attacker's transactions, for which the attacker can produce a valid signature. Making a double spend of the transaction itself is profitable when, by invalidating a transaction, the attacker can receive an irreversible payment or product without having to pay for it[8].

Race attack happen when an attacker sends two conflicting transactions in rapid succession into the network. This type of attack is relatively easy to implement in PoW-based blockchains. Merchants who accepts a payment immediately with 55 "0/unconfirmed" are exposed to the transaction being reversed. There are some possible countermeasures: "Listening period", "Inserting observers" and "Forwarding double spending attempts" [13].

In a public blockchain we can have a *sybil attack* where one hostile peer create lots of fake identities in order to defraud the system to break its trust and redundancy mechanism.[19].

2.1.1 *Public or Permissionless Blockchain.*

Permissionless means that anyone may join or leave the network at will. Public means that anyone, in principle, may propose a new state of the ledger[9]. Permissionless blockchain networks are decentralized ledger platforms open to anyone publishing blocks, without needing permission from any authority. Permissionless blockchain platforms are often open source software, freely available to anyone who wishes to download them. Since anyone has the right to publish blocks, this results in the property that anyone can read the blockchain as well as issue transactions on the blockchain (through including those transactions within published blocks). Any blockchain network user within a permissionless blockchain network can read and write to the ledger. Since permissionless blockchain networks are open to all to participate, malicious users may attempt to publish blocks in a way that subverts the system. To prevent this, permissionless blockchain networks often utilize a multiparty agreement or 'consensus' system that requires users to expend or maintain resources when attempting to publish blocks. This prevents malicious users from easily subverting the system. Examples of such consensus models include proof-of-work and proof-of-stake methods. The consensus systems in permissionless blockchain networks usually promote non-malicious behavior through rewarding the publishers of protocol-conforming blocks with a native cryptocurrency[18].

Proof-of-stake (PoS) aims to replace the way of achieving consensus in a distributed system. Instead of solving the Proof-of-Work, the node which generates a block has to provide a proof that it has access to a certain amount of coins before being accepted by the network. Generating a block involves sending coins to oneself, which proves the ownership. The required amount of coins (also called target) is specified by the network through a difficulty adjustment process similar to *PoW* that ensures an approximate, constant block time.[17]

2.1.2 *Private or Permissioned Blockchain.*

Similarly to the public blockchain, this one is also constituted by connected blocks, however it has some particularities in what concerns its members and the actions they can perform. It is considered private or permissioned since its participants are restricted to a set of permitted actions. External participants cannot submit or participate in transaction validation process[15].

In this type of blockchain users are added by a person/authority (be it centralized or decentralized), and it is not possible to join spontaneously. This type of blockchain is usually used when all its participants/organizations are known, but not all of them can be trusted. It is also used by organizations that need to more tightly control and protect their blockchain. These characteristics pose the problem of blockchain users having to trust on a single entity

Permissioned blockchain networks may thus allow anyone to read the blockchain or they may restrict read access to authorized individuals. This blockchain may be instantiated and maintained using closed or open source software. The consensus model can be determined by the organizations, based on how much they trust one another.Beyond trust, permissioned blockchain networks provide transparency and insight that may help better inform business decisions and hold misbehaving parties accountable. This can explicitly include auditing and oversight entities making audits a constant occurrence versus a periodic event[17].

2.2 Audit

Before we explain what auditing is, it is important to note that there are several types of auditing [11]. When most people think about audit, they imagine IRS agents arriving in an unannounced way, calculators in hand, ready to analyze countless boxes of receipts, invoices and bills. That is one type of audit, but it is certainly not the only one. Most audits have nothing to do with tax day and none of them are unannounced.

Traded companies are required by the securities and exchange commission to validate their financial position with an audit[6], and although they're not legally required to, privately held companies often perform audits at the request of banks, investors, and other key stakeholders. The goal is to ensure investors and other stakeholders that their cashflows balance sheets and profit and loss statements aren't materially misstated.

Public or private companies that go through an audit have the opportunity to gain valuable insight into how their businesses are performing[7].

The word audit means, to evaluate. Auditors are the ones who evaluate where money / information is coming from, where it's going, and what it's doing each step of the way.

Auditors need to think critically in order to understand what business decision drive the transactions. They also have to verify if what the company states is correct. During the audit process, they compare the documents the company uses in their day-to-day operations against what they have recorded in their financial statements.

Audit engagement can take between couple mouths till one year, depending on the size of the client and the complexity of the project.

2.3 Blockchain Audit

Blockchain auditing has the same objective as auditing, but is done differently, taking advantage of the characteristics of the blockchain[2]. As explained before, blockchain is essentially a type of database known as a distributed ledger that is decentralized with no central administrator. When a "user" saves a transaction in the database it is timestamped and saved in a block. Each "user" keeps a copy of the distributed ledger, the data is replicated and synchronized between all copies of the ledger in real-time. Since the ledger is both shared and encrypted, an attacker who wants to change the data contained in it, would have to simultaneously hack each node of the network and overcame the encryption[1], even if the attacker succeeded, users would be able to identify which data was tampered with, thus eliminating one of the disadvantages of traditional systems, single point of failure. This high level of security is one of the main attractions of the blockchain. It is increasingly being used for a range of functions where it is important to transmit data securely, particularly within the transaction-based financial services industry. For example block blockchain technology can be used to process payments, to create verifiable audit trails and to register digital assets such as stocks and bonds. In fact blockchain as the potential to change the way business in every sector operate. Once data stored in distributed ledgers is continually updated it offers finance team the possibility of real-time[4] reporting to both management and external auditors. This could free up auditor to offer more value-added services to their clients, such as strategic planning and support with wider business decisions. Like any technology, this one also has some drawbacks including the cost of implementation, privacy issues, lack of agreed standards and a limited scalability of blockchains.

2.4 Blockchain View

This concept of blockchain view is relatively new and there is very little related work. At this moment there are not many documents that use this concept yet, however we believe that in the future it will be widely used since it brings clarifications that were missing until today.

Blockchain views are tools that promote data portability once they enable blockchain interoperability. A view is an intermediary standardized data format not dependent on a specific blockchain (agnostic) that can be translated across blockchains. Views are self describing because they are represented via a multi-hash. They denote the commitment (e.g., via accumulators or Merkle trees) of a state in a particular time from a stakeholder centric perspective [3].

3 HYPERLEDGER FABRIC

Hyperledger Fabric[2] is an enterprise-grade, distributed ledger platform that offers modularity and versatility for a broad set of industry use cases. The modular architecture for Hyperledger Fabric accommodates the diversity of enterprise use cases through plug and play components, such as consensus, privacy and membership services.

On Hyperledger network, only parties directly related to the transaction deal are updated on the ledger, thus maintaining privacy and confidentiality. Hyperledger primarily leverages the concept of *channels* to ensure privacy and confidentiality. A *channel* is a virtual blockchain network which sits on top of a physical blockchain network and has its own access rules. They employ their own mechanism for transaction ordering and thus ensure scalability, thereby allowing effective ordering along with separation of data.

Chaincode[3] is a program, written in Go, node.js, or Java that implements a prescribed interface. It runs in a secured Docker container isolated from the endorsing peer process. Chaincode initializes and manages ledger state through transactions submitted by applications. Typically, it handles business logic agreed to by members of the network, so it may be considered as a “smart contract”. State created by a chaincode is scoped exclusively to that chaincode and can't be accessed directly by another one. However, within the same network, given the appropriate permission a chaincode may invoke another to access its state. To deploy a chaincode, a network admin must install the chaincode onto target peers and then invoke an orderer to instantiate the chaincode onto a specific channel. While instantiating the chaincode, an admin can define an endorsement policy to the chaincode.

Endorsement policy defines which peers need to agree on the results of a transaction before the transaction can be added onto ledgers of all peers on the channel.

Peer is a blockchain node that stores all transactions on a joining channel. Each peer can join one or more channels as required. However, the storage for different channels on the same peer would be separate. Therefore, an organization can ensure that confidential information would be shared to only permitted participants on a certain channel.

Orderer is one of the key elements employed in the the Fabric consensus mechanism. Orderer is a service responsible for ordering transactions, creating a new block of ordered transactions, and distributing a newly created block to all peers on a relevant channel.

Certificate Authority or CA is responsible for managing user certificates such as user registration, user enrollment, user revocation. More specifically, Hyperledger Fabric is a permissioned blockchain network that uses an X.509 standard certificate to represent permissions, roles, and attributes to each user.

Client is considered to be an application that interacts with Fabric blockchain network. *World State* maintains the current state of variables for each specific chaincode.

[2]https://www.hyperledger.org/use/fabric

[3]https://hyperledger-fabric.readthedocs.io/en/release-1.3/chaincode.html

Hyperledger Fabric currently can support two types of World State database including *LevelDB* and *CouchDB*. LevelDB is a fast key-value storage library written at Google that provides an ordered mapping from string keys to string values. CouchDB is a JSON-based database supporting rich querying operations based on JSON objects.

4 HYPERLEDGER CACTUS

Hyperledger Cactus[4] is a framework for integrating distributed ledgers. This tool was build under Hyperledger ecosystem aiming to provide decentralized, secure and adaptable integration between blockchain networks. One of the reasons why Cactus was created was the need to suppress an existing lack of frameworks/tools that would allow the integration of different environments, technologies and systems within a company or even among several companies, thus facilitating business. This framework is relatively new, however it already provides a pluggable architecture that enables the execution of ledger operations across blockchains, allowing users to make integration between different blockchains.

5 BUNGEE

BUNGEE (Blockchain UNifier view GEnErator) is a view generator, a system composed of four software components that creates, and processes views from a set of views, each one from a specific stakeholder. BUNGEE constructs views from a set of states coming from an underlying blockchain, giving them as input to a controller. The controller component can process views (including merging views or other processing after a view is merged). BUNGEE abstracts both analysts and developers from the underlying data structures from blockchains, and allow them to rationalize about the collected data from a business perspective[3].

In order to build the View, the process was divided into two main parts: taking a snapshot of the blockchain according to a specific participant p and constructing the View considering the desired time interval. Two important definitions that we must keep in mind in order to understand the BUNGEE tool are View and Snapshot.

blockchain View offers a stakeholder-centric, generalizable, self-describing commitment to the state of a blockchain, allowing for representing states from different blockchains in a standardized way.

Snapshot is the state of a system in our case blockchain at a particular point in time.

6 VIEW

This concept of blockchain view is relatively new and there is very little related work. At this moment there are not many documents that use this concept yet, however we believe that in the future it will be widely used since it brings clarifications that were missing until today.

Blockchain views are tools that promote data portability once they enable blockchain interoperability. A view is an intermediary standardized data format not dependent on a specific blockchain (agnostic) that can be translated across blockchains. Views are self describing because they are represented via a multi-hash. They denote the commitment of a state in a particular time from a stakeholder centric perspective [3].

7 AUDITVIEW

After we have developed BUNGEE, we decided that it would be interesting to have the possibility to analyze the views without the necessity of inspecting the JSON file. Therefore, we created a web app in order to make it easier and more user-friendly to analyze the View. Before we developed the software, several functional and non-functional requirements were identified:

(1) ***Data Integrity***, it should provide a service to detect unauthorized modification of data.
(2) ***Usability***, the tool should be easy to operate by people with advanced computer knowledge, but also beginners.
(3) ***Identifiability***, it should be possible to identify who audited the view.
(4) ***Non-repudiation***, it should have a property that prevents an entity from denying previous commitments or actions.
(5) ***Date Verifiability***, it should provide the possibility to verify creation date.

Let's start by exploring data integrity, given that if data is changed intentionally or unintentionally, the audit loses its value, and is no longer valid. In order to ensure the data integrity, each auditor has access to a set of asymmetric RSA 2048 bit keys, generated at the same time as the auditor account is created. These pair of keys allow each of them to sign the audit with their private key, thus taking advantage of the properties offered by the asymmetric keys. In order to create the signature for the audit file, the data is first hashed using SHA-256, then the signature is created and later included in the file.

Subsequently to the creation of an audit, it is possible to validate its integrity. To do this, we will only need the author's public key and the audit file. In order to do this validation, AuditView reconstructs the audit to memory, generates the corresponding hash and verifies the signature using the public key. Once AuditView have the result, it will show to the user if the audit still valid or has been tampered with. Regarding the audit view, we identify the assets that were audited, since AuditView gives the possibility to select specific assets without having to analyze all of them. Further to the View, we also present the id, hash and signature.

Concerning the auditing part, we have some pertinent information, such as the date and time of the audit creation. There are also the auditor's information, name and email. This information can be changed in the future and more information can be included, such as the organization the auditor belongs to, phone number, among others. One feature of AuditView is to verify the View integrity, therefore, before the audit is performed, the auditor is informed if the view is valid or if it was tampered with. In both cases we offer the possibility to perform the audit, however in the final file we display if the view is considered valid.

8 EVALUATION

This section will present the tests performed on each developed tool and the outcame of these tests. In certain cases it is common to compare the developed solution with previously existing solutions, however, in our case, since we are working at the frontier of

[4]https://www.hyperledger.org/use/cactus

knowledge, it is not possible to take advantage of this comparison, since there are no tools to compare yet, thus making the evaluation even more challenging.

9 BUNGEE EVALUATION

In order to evaluate BUNGEE, two types of tests were conducted, namely unit tests and performance tests. It is important to highlight that the obtained results were influenced by the performance of the machine used, so if the same tests were executed on a different machine, the values obtained would be slightly different.

The machine used as the following specs: Memory Ram 16Gb, Processor Intel® Core™ i7-7700HQ CPU @ 2.80GHz × 8, Graphics Card GeForce GTX 1050 Mobile.

For a tangible data set, 30 tests were performed for each number of transactions (10, 100, 500). From these thirty tests, ten of them were discarded in order to eliminate the outliers and have more consistent data. This step is particularly important since outliers are problematic. They represent measurement errors and/or processing errors and therefore impairing the consistency of the data.

Initially, we started by using the Hyperledger Caliper tool, which is a blockchain benchmark tool. Which allows users to measure the performance of a blockchain implementation with a set of predefined use cases. Hyperledger Caliper produce reports containing a number of performance indicators to serve as a reference, having the capability to analyze the Hyperledger Fabric blockchain. However, we soon realized that it was not feasible since the integration with Hyperledger Cactus is still very limited, so we had to find another strategy to solve this challenge.

To overcome this challenge we decided to measure how long the most important functions took to perform. For this purpose timestamps were placed throughout the code in order to record the initial time when the function started and the final time when it ended. After we had all these values, it was only necessary to compute the elapsed time, subtracting from the final time the initial time.

At the end of all tests being executed, the outliers were then removed. To this end, we start by calculating the first and third quartile (*Q1* and *Q3*). Then evaluate the interquartile range, which involves the subtraction between *Q3* and *Q1*, *IQR = Q3 - Q1*. Once we have the IQR, we calculate the lower bound (*Q1 − 1.5*IQR*) and the upper bound (*Q3 + 1.5*IQR*). Lastly we remove all the values below the lower bound and above the upper bound, thus obtaining a more homogeneous dataset of values.

In the sequence we will introduce three figures that represent the times required to perform the most important functions of BUNGEE. It is important to take the scale into consideration when these are analyzed, since they are all different. This decision was made since the execution times of the functions have significantly different ranges of values, therefore it was necessary to change the scale to have a better representation.

The figure 1 represents the chart of time versus number of transactions that each of the functions interacting with the blockchain requires. In order to make the values more perceptible, they were placed in the table 1.

The following figure portrays the time required by the *GenerateLedgerStates* function to process the number of transactions

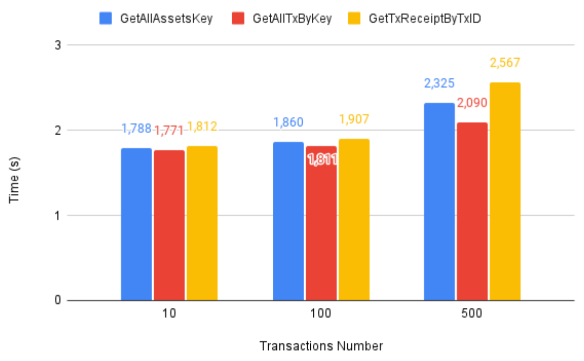

Figure 1: Number of seconds used by the functions: GetAllAssetsKey, GetAllTxByKey, GetTxReceiptByTxID

Tx Number	GetAllAssetsKey	GetAllTxByKey	GetTxReceiptByTxID
10	1,788347s	1,771108s	1,812183s
100	1,860473s	1,811004s	1,906563s
500	2,325358s	2,090279s	2,567222s

Table 1: GetAllAssetsKey, GetAllTxByKey, GetTxReceiptByTxID Values

selected. As we can see, the value is higher than the functions shown in the figure 2, this happens because the function *GenerateLedgerStates* invokes the three previous functions. This means that the time it takes is the sum of the previous times, plus the processing time of the remaining function.

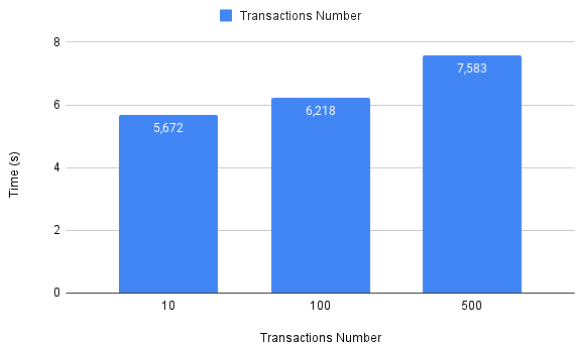

Figure 2: Number of seconds used by the function: GenerateLedgerStates

Tx Number	GenerateLedgerStates
10	5,671638s
100	6,218040s
500	7,582860s

Table 2: GenerateLedgerStates

Lastly, we have the figure 3, which, as the previous ones, represents the time spent by the functions *GenerateSnapshot* and *GenerateView* in detriment of the number of transactions.

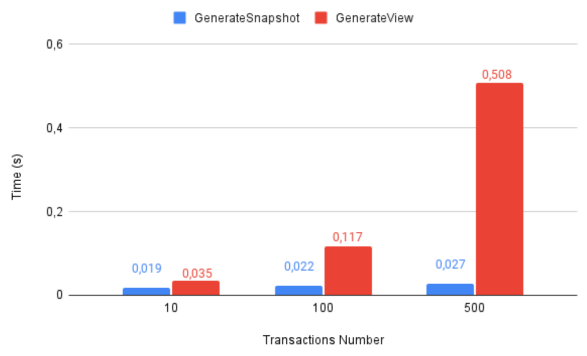

Figure 3: Number of seconds used by the functions: GenerateSnapshot, GenerateView

Tx Number	GenerateSnapshot	GenerateView
10	0,018515s	0,035067s
100	0,021907s	0,116664s
500	0,027118s	0,507847s

Table 3: GenerateSnapshot, GenerateView Values

Regarding the size of the views a file with 10 transactions will be about 21.1 kB, with 100 transactions it will be 202.8 kB and with 500 transactions it will be about 1Mb.

Our initial goal was to use a real use-case, however, since it was not possible to obtain a real use-case from a company that uses the technology, due to the sensitivity of the data, we created a use-case as close to reality as possible. This consisted of exchanging assets between peers. These had the possibility to sell and purchase assets. This way, it was possible to have data about asset owner's exchange inside our View. That being said, the values presented above not only vary with the change of the computer used to run the tests, but also vary with the change of the use-case. The values presented do not include the test ledger creation time nor the insertion of transactions, since in a production context these actions would not be performed by BUNGEE but by the peers in the blockchain.

It is important to note that BUNGEE was developed to meet the definition provided in the paper [3], which was submitted to the Journal of Parallel and Distributed Computing.During the development, some limitations were found and, consequently, solutions to overcome them were provided, thus helping to improve the paper.

10 AUDITVIEW EVALUATION

As a means of testing AuditView we have used three types of methods, these were testing using low fidelity prototype[5], user testing with the tool completely developed and unit testing.

[5]Low-fidelity (lo-fi) prototyping is a quick and easy way to translate high-level design concepts into tangible and testable artifacts. The first and most important role of lo-fi prototypes is to check and test functionality rather than the visual appearance of the product.

To perform the tests with the users, we started by understanding who would be a potential user of the tool, therefore we recruited six people. During recruitment, it was explained what the tests would be comprised of. The recruited people are computer engineering students, their ages range between twenty-one and twenty-three years old, and they are proficient when it comes to computer skills. The tests were divided into two sessions, the first session consisted of: Presentation of the tool, its purpose, and interaction tests using a low fidelity prototype. The second session involved testing the fully developed tool. For both sessions, a script of directives was made so that the users would know what was required and thus perform the actions.

In the first session we started by presenting why the tool was developed, then we presented the functionalities we intended to develop, and finally we did the usability test using the low fidelity prototype. This prototype was developed in *Figma*[6] which is an interface design tool. This test showed that after registration it was important to give feedback to the user in order them to verify that the actions had been carried out successfully and not redirect directly to the main page, we realized that we should change the way we presented the two main functions to be more intuitive to use, and in the end we realized that it was important to have a navigation bar to facilitate navigation through the tool.

The second user test took place some time later, when the tool was already fully developed and the previous feedback incorporated. We could conclude that there was no difficulty in carrying out the requests made in the script for the users. Therefore, we can consider that the first session was very important once it made the tool easier and more intuitive to use, and this was confirmed in the second session.

These tests introduced some degree of complexity since it was necessary to take into account the availability of these six people, the development of the functionalities, and the stipulated deadlines.

Finally, we have the unit tests, these were the simplest to do since there was no need to manage logistics as in the previous tests. In order to perform the tests, we have used the *unittest* framework, which was originally inspired by JUnit. This tool gave us the possibility to make test cases, thus verifying the specific response to a particular set of inputs. Besides the unit tests, suit tests were also performed, consisting of a collection of test cases executed together. This way, all the functionalities presented were tested, and some problems were detected and subsequently rectified. These unit tests were performed between the first and second session with the users, therefore allowing us to present a stable version of our tool.

11 FUTURE WORK

As we can see throughout this document, this work paves the way to appealing future works.

One of the functionalities to be developed in BUNGEE would be the View Merging part, which would be used when there are two views that we want to see in a single (unified) view. This functionality could be invoked on the fly after a view has been generated, or even when two previously generated views already existed.

[6]https://www.figma.com/

With the code that is already developed it won't be hard to develop this functionality, the steps needed would be only six and two of these were optional. In order to develop this function, BUNGEE would have to provide the ability to upload existing views and validate their integrity or the ability to retrieve them. After having the two views in memory, it would be necessary to reconstruct them so that they could be represented by the classes that constitute them, thus creating an extended state. The following step would be to merge the classes in order to have a unified view. After having the view unified, it would be necessary to do the signature of the view. As optional steps, we would have the persistent publication in the blockchain and its proof. At last, the unified view would be returned to the client.

It would also be interesting to test BUNGEE using other blockchains such as Ethereum. This change of blockchain would not introduce much resistance since all the logic is already developed. It would only be necessary to change the plugin and possibly rename some functions in the smart contract to comply with the nomenclature used in the Ethereum blockchain.

In the future, if we decide to develop something to complement BUNGEE, it would be interesting to have a tool that could validate each of the transactions returned in the snapshot section, and verify the transactions related to an asset and its endorsements.

12 CONCLUSION

Blockchain audit is a rapidly advancing research topic. Nevertheless, state-of-the-art tools are still limited and can only be used with cryptocurrencies. In this document, we describe how can we visualize data partitions within a blockchain and perform arbitrary operations on it, for instance generating blockchain views to conduct a blockchain audit.

We introduced state-of-the-art for fields like blockchain (private and public), auditing, blockchain auditing even though it does not have an extensive repository of articles and scientific papers.

We proposed a solution where we give the possibility to auditors, cybersecurity experts, and in general, developers to have audits facilitated because different data partitions can be analyzed from a specific angle. We also discussed what would be the advantages and disadvantages of our solution.

We believe that we have contributed to the state-of-the-art in blockchain audit and blockchain interoperability by generally develop a tool which allows users to use blockchain views and also the possibility for stakeholders to audit the blockchain in an easier, faster and more automatic way. In the future we expect to see auditors, cybersecurity experts, and in general, developers using the view concept to enhance audits, utilizing BUNGEE in this way. During development, we realized that there is a pronounced learning curve, especially for those who are not experienced with blockchain related technologies.

This is a much broader area than we might think, however, after settling this curve, tool development becomes very straightforward. Something important and very positive to highlight is the proximity of the maintainers of all the Hyperledger projects to the developers.

In conclusion, as we can see, this theme is undoubtedly very innovative, and can be applied to real cases, bringing great benefits that until now did not exist.

REFERENCES

[1] Ashar Ahmad, Muhammad Saad, and Aziz Mohaisen. 2019. Secure and transparent audit logs with BlockAudit. *Journal of network and computer applications* 145 (2019), 102406.
[2] AA Baev, VS Levina, AV Reut, AA Svidler, IA Kharitonov, and VV Grigor'ev. 2020. Blockchain technology in accounting and auditing. *Accounting. Analysis. Auditing* 7, 1 (2020), 69–79.
[3] Rafael Belchior, Limaris Torres, Jonas Pfannschmid, André Vasconcelos, and Miguel Correia. 2022. Is My Perspective Better Than Yours? Blockchain Interoperability with Views. (2022).
[4] Derrick Bonyuet. 2020. Overview and impact of blockchain on auditing. *International Journal of Digital Accounting Research* 20 (2020), 31–43.
[5] David S Evans. 2014. Economic aspects of Bitcoin and other decentralized public-ledger currency platforms. *University of Chicago Coase-Sandor Institute for Law & Economics Research Paper* 685 (2014).
[6] Marshall A Geiger and Kanan Raghunandan. 2001. Bankruptcies, audit reports, and the reform act. *Auditing: A Journal of Practice & Theory* 20, 1 (2001), 187–195.
[7] Ann Hylton. 2002. A KM initiative is unlikely to succeed without a knowledge audit. *Knowledge Board.[Consulta: 15 febrero 2009]* (2002).
[8] Emanuel Ferreira Jesus, Vanessa RL Chicarino, Célio VN De Albuquerque, and Antônio A de A Rocha. 2018. A survey of how to use blockchain to secure internet of things and the stalker attack. *Security and Communication Networks* 2018 (2018).
[9] Tommy Koens and Erik Poll. 2018. What blockchain alternative do you need? In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 113–129.
[10] Roy Lai and David LEE Kuo Chuen. 2018. Chapter 7 - Blockchain – From Public to Private. In *Handbook of Blockchain, Digital Finance, and Inclusion, Volume 2*, David Lee Kuo Chuen and Robert Deng (Eds.). Academic Press, 145–177. https://doi.org/10.1016/B978-0-12-812282-2.00007-3
[11] John Mullarkey. 1984. Case for the structured audit. (1984).
[12] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf.
[13] Nidhee Rathod and Dilip Motwani. 2018. Security threats on Blockchain and its countermeasures. *Int. Res. J. Eng. Technol* 5, 11 (2018), 1636–1642.
[14] Janusz J. Sikorski, Joy Haughton, and Markus Kraft. 2017. Blockchain technology in the chemical industry: Machine-to-machine electricity market. *Applied Energy* 195 (2017), 234–246. https://doi.org/10.1016/j.apenergy.2017.03.039
[15] Siamak Solat, P. Calvez, and Farid Naït-Abdesselam. 2020. Permissioned vs. Permissionless Blockchain: How and Why There Is Only One Right Choice. *Journal of Software* 16 (12 2020), 95 – 106. https://doi.org/10.17706/jsw.16.3.95-106
[16] Paul J. Taylor, Tooska Dargahi, Ali Dehghantanha, Reza M. Parizi, and Kim-Kwang Raymond Choo. 2020. A systematic literature review of blockchain cyber security. *Digital Communications and Networks* 6, 2 (2020), 147–156. https://doi.org/10.1016/j.dcan.2019.01.005
[17] Pavel Vasin. 2014. Blackcoin's proof-of-stake protocol v2. *URL: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper. pdf* 71 (2014).
[18] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. 2019. Blockchain technology overview. *arXiv preprint arXiv:1906.11078* (2019).
[19] Shijie Zhang and Jong-Hyouk Lee. 2019. Double-Spending With a Sybil Attack in the Bitcoin Decentralized Network. *IEEE Transactions on Industrial Informatics* 15, 10 (2019), 5715–5722. https://doi.org/10.1109/TII.2019.2921566