Seclending Chain

A Blockchain Experimentation at Banco de Portugal

André Leal, João Rodrigues, Nuno Pereira
Banco de Portugal
October 25, 2020

Abstract

This paper presents the work done by Banco de Portugal (BdP) regarding the experimentation on Distributed Ledger Technologies (DLT), more specifically on blockchain. The main goal of this work is to better understand the differentiating capabilities of this novel technology, the challenges from a Central Banks perspective and learn on how to develop and deploy blockchain based solutions. It also aims to understand how those developments can be further integrated within the current BdP's enterprise technology ecosystem.

Detailed information is provided regarding on how the internal experimentation was conducted, from the identification of a relevant use case named "Securities lending", to its development and final deployment.

For this experiment, BdP used Hyperledger Fabric 1.3 blockchain platform and deployed it on a private DLT environment. A CLI, REST API and Web interface were also developed, being the latter implemented on a low-code platform named OutSystems.

The network was further extended to other National Central Banks (NCB) - De Nederlandsche Bank and Oesterreichische Nationalbank. The challenges faced regarding the extension of the network to those NCB's are also highlighted.

This experimentation increased BdP's knowledge on blockchain technology and distributed peer-to-peer networks. The usage of an Experimentation Framework along with automated scripts resulted on faster and more meaningful results.

Keywords: Blockchain, Distributed Ledger Technology, Securities lending, Hyperledger Fabric, National Central Bank, Banco de Portugal

1 Introduction

The current payment ecosystem, as we know it, is being challenged. The appearance of crypto-assets [1] – being bitcoin the best known example – has proved that it is possible to have a completely fully functional decentralized payment system on a peer-to-peer based model, without the need to have a centralized entity responsible and accountable for managing transactions within the system. Blockchain is a specific implementation of a Distributed Ledger Technology (DLT) enabling the development of distributed solutions, being crypto-assets often supported by this technology.

Bearing in mind the potential of the technology and the appearance of new disrupting business models, especially on Fintech areas, BdP and also other National Central Banks from the Eurosystem, have started to look into this new world of distributed solutions to better understand the implications underneath the technology, as well as its new capabilities, limitations, implications, possible scenarios and use cases.

BdP believes that DLT and related platforms and technologies could be disruptive within the current capabilities of NCBs, especially with functions related to Money Market Operations.

Therefore, it was considered important to follow up on DLT and blockchain trends to raise our basic knowledge on both technical aspects and generic capabilities.

Following the work developed in closed collaboration within the Eurosystem System of Central Banks [2], and motivated to better understand the technology, Banco de Portugal decided to start an internal experimentation with the following proposed goals:

- Extend Banco de Portugal knowledge on blockchain capabilities and related technologies.
- Identify one candidate use case that could take advantage of blockchain technology.
- Create a BdP DLT infrastructure to support the experimentation.
- Explore the development and deployment of smart contracts for the chosen use case.
- Explore the development and deployment of different types of interfaces for further integration with current BdP's ecosystem.
- Assess the extensibility of the use case itself.

Given BdP's low maturity with the associated platforms and technology to be used, it was decided to start with a simple use case that could benefit from blockchain unique capabilities. Specifically, the characteristic of decentralization between distinct Organisations, highlighted the importance to look at a use case that could also be relevant to other NCBs, increasing opportunities for further collaboration. As such, it was decided to harvest a low risk-based business function.

The aim of this paper is to contribute to shorten the learning curve when entering the world of distributed solutions by showing the work developed

[1] Commonly known as crypto-currencies although it does not fulfil currency's definition

[2] The ECB prompted the creation of such collaboration group - EUROChain - as a learning tool for all interested NCBs to explore and share experiences on DLT

by Central Bank of Portugal internally and the results of the extended collaboration with De Nederlandsche Bank and Oesterreichische Nationalbank.

In the end, a section with the takeaways is available, where the main learnings from this experimentation are highlighted. A section with the next steps is also present with relevant challenges that can be further explored.

2 Distributed Ledger Technology (DLT)

Distributed Ledger Technology (DLT) is a rapid growing technology where the database – a ledger - is shared across several entities, each one having its own copy [1]. These digital systems are characterized for being immutable as once a transaction is recorded on the ledger it is mathematical infeasible to tamper that given record.

As a distributed technology there is no central point of authority responsible for ensuring the proper system operation. This responsibility is dematerialized in the network as a whole.

2.1 Blockchain vs DLT

The concepts of blockchain and DLT, although incorrectly, are often used interchangeably. DLT represents a broader concept of having a shared and distributed ledger across a network where the typology and implementation of the ledger is out of scope. Blockchain is a specific DLT implementation where the ledger is composed of chains of blocks of data, cryptographically connected [2]. Blockchain is the most known DLT implementation, much due to the amount of crypto-assets that are supported on it. Nevertheless, there are other possible implementations such as Hash Graph based DLT - Distributed Hash table (DHT) [3, 4] or Directed Acyclic Graph (DAG) [5].

Therefore, every blockchain system is a DLT system, but the opposite is not true. This work will focus on blockchain as it was the specific technology used for the experimentation.

2.2 The value of trust

In a world of distributed collaboration, trust is always a central point of discussion. Blockchain technology redefines the concept of trust, allowing new collaborations between Organisations without the need to explicitly set trust relations between them or with other third-party intermediaries [6]. Each Organisation needs only to trust the network as a whole. The trust mechanisms are built around cryptography and mathematical proven algorithms, which ensures that each transaction is correctly executed[3].

2.3 Blockchain vs Database

Blockchain can be seen as a distributed, digital system of records, where the main goal is to store information. Databases, although having the same purpose, are a very different kind of technology [7].

[3]More information available on chapter '3.5 Consensus'

Authority

Blockchain doesn't have a centralized, single point of authority. In a blockchain implementation, the network is responsible for every decision. However, in some blockchain implementations, such us permissioned blockchains, there may be some form of centralization.

Databases, on the other hand, have normally a centralized authority, commonly associated with the notion of an administrator.

Data Handling and Integrity

Databases support *CRUD* operations allowing the information to be updated without ensuring historical traceability. With blockchain all data is typically recorded as unique transaction. Therefore, it is mandatory to read all ledger's records in order to have access to the full system. Blockchain platform do not support neither update or delete operations on existing transactions, being read and write the only ones available. Such model contributes directly for achieving immutability.

Transparency

By default, blockchain offers transparency as the ledger is fully available to every participant. In some implementations the information in the ledger may be encrypted. In Databases, on the other hand, the access to the information is normally granted by a central authority, namely the administrator.

Cost

Blockchain, due to its novelty, has a considerable learning curve, which requires a higher investment when compared to mature technologies like Database systems. In the future the cost of implementation between the two technologies may shorten as it is expected that future offers will evolve to provide blockchain capabilities as a service, thus reducing the actual cost. Nevertheless, the cost will likely be always higher because of the complexity of implementing fully distributed and decentralized systems.

Performance

Blockchain has typically lower performance when compared with Databases, mainly on public blockchains. This is normally related to the specificities of the consensus mechanism used, which tends to be very costly[4].

3 Demystifying Blockchain Concepts

When reading information regarding blockchain, it is common to be exposed to a considerable amount of many different concepts. This chapter summarizes the most frequent ones, allowing newcomers to better understand the use

[4]The cost may be represented in time, processing power, stake, etc.

and applicability of the platforms and technology along with the references through the document.

3.1 Permissioned vs Permissionless Blockchain

Blockchain technology is often associated, sometimes exclusively, as a permissionless network topology. This has been the result of its historical association with bitcoin. However, when Organisations started to look on blockchain concepts from an enterprise perspective, trying to take advantage and benefits from its associated capabilities, a demand for a new topology was born - permissioned network.

Public and private networks are often used interchangeably as synonyms for permissionless and permissioned networks respectively [8]. On a permissionless network, any identity can participate on the network. This means that anyone can access the ledger, submit and validate transactions on the network. In order to keep the network running, the use of a token is required (e.g.: Ethereum [9] has the ETH token) as a form of incentive to the community that is using and maintaining it.

Permissioned networks, although using the same set of capabilities, only allow participation from known identities. Only those identities are able to access to the ledger and execute operations. This specific characteristic makes this network topology the preferable choice for Organisations trying to leverage blockchain as an enterprise solution, where distinct but known entities create a distributed ecosystem, sharing then a distributed responsibility.

When compared to a permissionless network, the greatest advantage of a permissioned network is performance. Because the network is restricted to known Organisations, it is possible to use simpler consensus mechanisms as part of a Byzantine Fault Tolerance (BFT) solution to still run a safe and tamper-proof ledger in the absence of any central control. As such, these network topologies are less distributed.

A permissioned network is essentially a blockchain topology with some definition of trust by design. This makes the network simpler and more performer, but with a greater risk of compromising immutability.

Some of the most known activity sectors and business models that may benefit from this technology capabilities are the supply chain sector [10], the financial sector (e.g.: JPCoin [11]) and payment systems [12]. Other use cases that could leverage on blockchain technology are related to “Digital Identity” [13], “Reference Data” (e.g.: MADRE[5]) and "Gaming" (e.g.: Cryptokitties[6]), among others.

3.2 Network, Nodes, Organisations and Peers

In its simple design, a DLT network is a combined set of connected nodes sharing a common ledger and a smart contract implementation for enforcing the agreed business rules.

[5] `https://www.banque-france.fr/en/banque-de-france/about-banque-de-france/le-lab-banque-de-france/banque-de-france-labs-achievements`

[6] https://www.cryptokitties.co/

A node is a single point on the network that is associated to a given entity or Organisation. Depending on the role, a network node may be exclusively allowed to read the information on the ledger or be able to submit and accept new transactions.

On some blockchain implementations the notion of a peer might exist. In those implementations each Organisation is composed by one or more peers, which are no more than nodes related with the same entity. Those nodes are deployed and operate in parallel in order to improve the performance and resiliency of the network.

3.3 Ledger

The ledger is the digital system of records of a DLT network. On blockchain - one of the most common ledger implementations - the ledger is composed by a sequence of blocks of transactions chained together through the use of hashing functions[7]. In order to make the chain work, each block needs to have a pointer to the identifier of the next block.

3.4 Smart contracts

Smart contracts are a digital piece of code that implements the business rules by which the DLT network should run. The name came from the dematerialization of contracts on a digital and smart way, transposing the physical contractual obligations into the DLT network [1]. The contracts represent all the functions that can be executed – from the requirements to the end results.

Since every node on the network runs the same smart contract, each transaction is enforced to be executed accordingly. Such behaviour assures that it is impossible for the network to propose or accept transactions that do not respect the running smart contract.

3.5 Consensus

Since a DLT network does not have any kind of intermediary responsible for ensuring the proper execution of the network and deciding if any given transaction is valid, the execution of any smart contract must be accepted by the remaining entities on the network, according to a specific policy. If the policy is fulfilled than a consensus is achieved between those entities – and that specific transaction is considered valid and broadcasted to all nodes.

Consensus algorithms ensure that transactions submitted to the network are executed on every node of the network, on the same exact order. Several algorithms can be used to implement consensus, but there are significant differences between permissioned and permissionless networks [8].

Permissionless Networks

The most known permissionless distributed networks' consensus algorithms are Proof of Work (POW) [14] and Proof of Stake (POS) [15].

[7] More information in Chapter '3.6 Cryptography'

Proof of Work is the consensus mechanism implemented on bitcoin. In short, this consensus algorithm demands that, for a given block of transactions to be accepted by the network and thus chained to the existing ledger, some entity must first solve a very computational demanding problem. This process is known as mining. Only then the block can be sent to the remaining network as being the new truth. The network can easily validate the new block by checking if the mathematical problem was correctly solved. This consensus algorithm assumes that for a given block to be added to the ledger, a specific amount of computational work needed to be done. The same is true when someone wants to update the information of an already submitted block. Such task will require the update not only of that block but also the whole chain of blocks, thus requiring more work to be done. Therefore, older blocks will be in longer chains, making it harder to change the truth.

On Proof of Stake the concept is similar to the Proof of Work, but instead of processing power, a stake is used. This means that the bigger the stake ("coin" of the system) an entity has on the network, the higher its validation power is.

Permissioned Networks

On a Permissioned network topology, any possible violation on the network can be resolved with contractual obligations that already exist outside of the network [1]. Although applying consensus algorithms on permissioned networks could be seen as a redundant feature, as each node is known and authorized in advanced, this mechanism can still be used to increase the resilience of the network.

Simpler consensus mechanisms can be employed, such as the Practical Byzantine Fault Tolerance (PBFT) [16]. This algorithm is commonly used on distributed systems to improve their resilience assuming the presence of f malicious nodes, as long as the whole system is of size $3f + 1$.

3.6 Cryptography

One of the intrinsic characteristics of blockchain is immutability [8]. An immutable DLT network assures that whenever a transaction is written to the ledger, chained to a block and further disseminated through the network, it becomes mathematically infeasible to change it.

Cryptography is then essential to ensure the security of the network and also its immutability. To achieve that, DLT systems use a specific cryptographic technique called Hashing.

Cryptographic hashing [17] transforms any given type of information, no matter its size, into a fixed length unique identifier. One characteristic is that even the slightest change on the input information - for instance adding a comma - results on a totally different unique identifier as the end result. Another characteristic is irreversibility, which means that although it is very easy to apply a hash function and produce the unique identifier - resulting on a fast validation as well - the reverse operation is computational infeasible.

[8] Although considered immutable (tamper-proof), blockchain networks can still be tampered. The hardness for achieving such result is what grants the immutable condition to blockchain platforms (See more in '3.7 Double Spending and the 51% attack')

On blockchain each block has a unique identifier as the result of applying the hash function on its own information. Each block also has the hash identifier of the previous block, thus creating a chain reference. If the data on a given block is tampered, its unique identifier would become completely different, breaking the chain sequence, as the following block would have an invalid reference to its precedent block. Therefore, for the tampering to be accepted, the intended block needs to be changed along with every single block that follows the whole chain.

Other techniques like encryption and signatures are used to further protect the confidentiality of the information inside the blocks. It is also used to identify the entities (nodes) that can access the network.

3.7 Double Spending and the 51% attack

One of the most known attacks on blockchain networks, both permissioned and permissionless, is known as the *Double Spending* attack. As the assets are digitized and thus dematerialized on a blockchain network, it is possible to try to use the same digital asset on two separated transactions. This attack is mitigated by combining the characteristics of immutability of the ledger, the use of strong consensus algorithm and the execution of smart contracts.

Using proof of work for instance, it is mathematical infeasible for the same Organisation to be able to compute two concurrent transactions[9] faster than the time needed to generate new blocks by the remaining network nodes. This means that it is very unlikely that a given entity could change an already accepted block and future ones[10] faster than the remaining network.

The 51% attack simply states that if a given entity or group of entities has in their possession 51% of the processing power of the network, then the probabilities to decide the next block to be written to the network are higher. If this entity or group is maleficent, the successful execution of a double spending attack is more likely. With proof of Stake algorithms this attack can also be achievable by having 51% of the stake of the entire network.

4 Blockchain Platforms

As with any other technology, a proper platform is required to develop suitable solutions. This experimentation was developed using the IBM Hyperledger Fabric platform. The choice is mainly related to BdP's experience on previous collaborations within the ESCB community and better alignment with enterprise solutions by leveraging on a permissioned network topology.

This section will dive into Hyperledger Fabric concepts and components. A short context will be given on other platforms available on the market.

[9] A given asset is spent on the first and second block. Since the blocks were compute by the same entity, it can choose to "replace" the first block with the second, resulting in a chain fork and thus, spending the same asset twice

[10] In order to have the longest chain of the network

4.1 Hyperledger Fabric

Hyperledger is a multi-project hosted by the Linux Foundation with contributions from IBM. It supports the development of open source and collaborative blockchain platforms and tools.

One of the available projects is called Hyperledger Fabric, a permissioned blockchain platform supported on a modular architecture [18]. It is designed to be a versatile solution supporting pluggable implementations for its various components and deployable through docker containers.

An Hyperledger Fabric network is composed by the following key components [19]: Organisations, Orderer service, ledger, World State View, channels, chaincode, consensus algorithm and Certificate Authorities. In order to interact with the network one can also use the available SDKs.

Organisations and Peers

Organisations are the entities responsible for creating the blockchain network. They are the ones who manage and contribute with resources, such as participants entities and *peer nodes* to run the network [11]. being this a permissioned network, each Organisation and all its participants are well known.

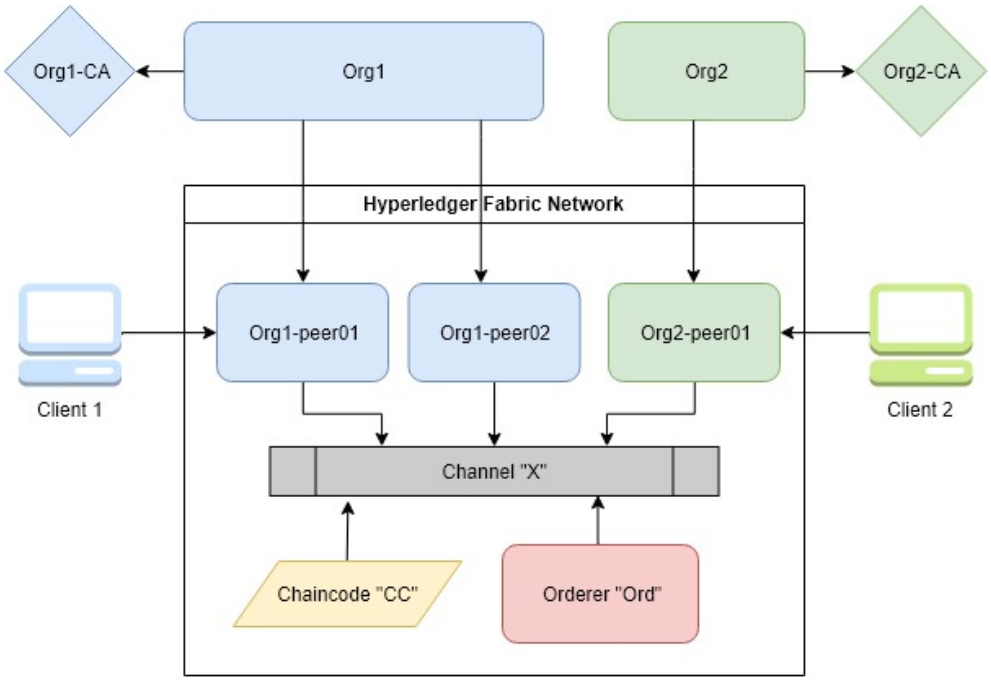

Figure 1: Hyperledger Fabric Architecture (simple view)

Peers, also called nodes [12], belong to the Organisations and are the ones who connect all the various Organisations within it. They are also the ones who run the chaincode and store the ledger. Each Organisation may have

[11] `https://hyperledger-fabric.readthedocs.io/en/release-2.2/peers/peers.html#peers-and-organizations`

[12] On this document, peer and node are used interchangeably

more than one peer for performance or resilience reasons. Some peers are able to validate and endorse transactions. In those situations, they are called Endorsing Peers.

Consensus

Consensus is the algorithm which ensures that each node on the network agrees and has the same understanding of what is the correct order of the submitted transactions. On Hyperledger Fabric the consensus mechanism is coupled to the Orderer service typology.

Orderer

On permissionless ledger networks, like bitcoin for example, any node can participate to establish consensus. A probabilistic algorithm is used to ensure that the ledger will always be consistent in some point in the future using mathematical functions. This type of algorithm allows the appearance of ledger divergences (also known as forks) because different nodes might have different views of the accepted order of transactions[13].

On Hyperledger Fabric, the Orderer [20] is the service component that is responsible for ensuring that the order of the transactions is the same across all nodes of the network. The service is run by a node in the network that has this special role. With this type of implementation, the network achieves higher performance without the existence of ledger forks.

There are four known implementations for the Orderer Service [14]:

- **Solo**: Orderer service is running on a single centralized node. This default implementation is typically used only in development scenarios. For production, a higher availability solution must be employed.
- **Raft**: Crash fault tolerant based on Raft protocol[15] [21]. It is easier to implement than a Kafka implementation.
- **Kafka**: Similar to RAFT-based ordering but with a harder implementation and management.
- **Byzantine Fault tolerant**: A BFT consensus algorithm is still under work by the Hyperledger Fabric team [16]. Nevertheless, the modular architecture of the platform allows the integration of custom implementations as for instance the BFT-SMART [16]. Although more resilient, this implementation is still less performant.

With its modular architecture, any of these ordering services implementations can be plugged on the network without affecting the remaining components configuration.

[13] On such scenario, the longest chain is the one that ends up being accepted by the network

[14] Solo and Kafka implementation have been deprecated since v2.x of Hyperledger Fabric (`https://hyperledger-fabric.readthedocs.io/en/release-2.2/orderer/ordering_service.html#ordering-service-implementations`)

[15] https://raft.github.io/

[16] `https://hyperledger-fabric.readthedocs.io/en/latest/Fabric-FAQ.html?#bft`

Ledger and World State View

Hyperledger Fabric's ledger is a blockchain structure where all submitted transactions are registered. To avoid the need for each node to go through the whole chain in order to check the current state of the ledger, a database called World State View is deployed alongside with the current state of the ledger.

This database is not immutable, being used as a simply cache system. Therefore, it is common to have the World State View values updated frequently, based on the transactions submitted to the network. On the ledger, on the other way, each transaction is always registered as a new one, without changing any past transaction.

For the World State View, two implementations are available: LevelDB and CouchDB. The latter is the most common choice as it supports rich queries improving the smart contract performance and capabilities. Each node has its own World State View implementation along with its blockchain ledger.

Chaincode

Chaincode is the name associated to a Smart Contract implementation on Hyperledger Fabric. Chaincode can be written in Go, Nodejs or Java. A chaincode runs on each node with the responsibility to handle business logic by exposing the operations available on the network.

Channels

Channel is a concept on Hyperledger Fabric that is used to represent a private communication between a set of network nodes.

A network can have as many channels as needed, being the transactions on a channel only visible to its members. Each channel has a specific chaincode and the associated ledger. With this architecture each node can be a member of several distinct channels, having, as a result, a copy of each channel's ledger.

Certificate Authorities

In order to interact with the network a valid identity is required. Certificate Authority (CA) is the service that typically runs on each node and is used to generate and manage those digital identities.

Each identity consists on a digital signed certificate that was issued by this service. A CA is responsible for operations over these certificates like issuance, enrolment, renewal and revocation.

Software Development Kits (SDKs)

Hyperledger Fabric has two SDKs available: Nodejs and Java. It is possible to leverage on these SDKs to further develop applications that will interact with the blockchain network on behalf of the users. Operations could range from channel creation and chaincode installation to invoking transactions and querying the ledger, among others.

4.2 Other blockchain Platforms

In this chapter a small overview is given on two well-known blockchain platforms: R3 Corda and Ethereum.

R3 Corda is a permissioned blockchain platform designed specifically for the development of enterprise financial solutions. Ethereum on the other hand is a permissionless blockchain platform. It was one of the first permissionless blockchain platforms allowing the development of solutions on top of its currency - Ether.

R3 Corda

Corda is an open source platform for developing DLT solutions owned by R3 [22, 23]. Just as Hyperledger Fabric, this platform allows the implementation of permissioned ledger networks with a focus on the enterprise environment, more specifically on financial area. On Corda the consensus mechanism is different from the one used on Hyperledger as it only requires the acceptance from the parts evolved on the transaction.

Although referred as a blockchain implementation, it is still not clear if it fits the description as its implementation is quite distinct from a typical blockchain system. For instance, transactions on Corda are validated in real time by the entities involved, and the resulted transaction is not broadcast to the whole network as it is shared on a need to know basis. On the other hand, and in the same way as in a blockchain system, transactions are cryptographically chained. In this specific implementation each transaction is chained to all transactions it depends on.

Ethereum

Ethereum is a well-known permissionless ledger network that also achieves consensus by using Proof of Work algorithm [17]. This platform implements the Ether crypto-asset, that can be used to send payments between members of the network [9, 23].

More than a payment system, Ethereum allows to build new solutions on top of its blockchain technology, making it a distinct permissionless blockchain solution when compared to bitcoin. The new business cases implemented on Ethereum platform use the ETH crypto-asset as a payment currency. Given its ability to support the development of new blockchain permissionless ledgers, Ethereum was one of the main platforms used for the creation of new projects and the appearance of Initial Coin Offers (ICO) [21].

5 Experimentation Framework

Being this the first internal experimentation performed on blockchain leaded by Banco de Portugal, a framework was designed to validate and ensure the achievement of the intended results. The framework was designed as a six stage process with clear deliverables on each stage.

[17] Planning on implementing Proof of Stake consensus by 2020

Stage 1: Experimentation Goal

To achieve a more focused experimentation initiative, it is important to have a clear vision and understanding on what the intended goal is. This creates a focus on the team itself and limits the chances of a broader dispersion of related activities, which can result on failing to reach conclusions.

Deliverables

1. Vision for experimentation.
2. Main achievements to be pursued.

Stage 2: Use Case

For the experimentation to have a solid start with the needed sponsorship, it is important to identify a relevant use case. Such use case must be able to take advantage on blockchain's distinct characteristics.

Deliverables

1. Identification and clarification on candidate use cases.
2. Team and roles responsible for it.

Stage 3: Business Model

As in any business activity, a clear communication from the business requirements to the final implementation is very important. A clear vision on what business processes have to be implemented is crucial. In collaboration with the business team, and for each life-cycle captured from the use case, the main states and transitions must be identified.

Deliverables

1. A machine-state diagram where the business processes are identified.

Stage 4: Chaincode Development

Development of the chaincode based on the machine-state diagram delivered on the previous state. A chaincode operation shall be implemented for each state transition, which represents a business process. Each state must represent a given state of the asset on the ledger.

Deliverables

1. Chaincode for the Business processes identified.

Stage 5: Infrastructure

As in any experimentation it is important to ensure a dynamic and flexible environment. This mindset allows the team to surpass and dynamical adapt to new challenges that may and will occur during the experimentation. Therefore,

it is required to ensure that a proper infrastructure environment is available with all the permissions and needed capabilities. On this stage it is relevant to already have a clear idea of the experimentation goal so the environment can be configured accordingly.

Deliverables

1. Identification, deployment and configuration of a DLT infrastructure environment.

Stage 6: Integration

The final stage in this framework is related to the implementation of interfaces required for integrating the resulted experimentation with BdP's current enterprise ecosystem. For instance, it may imply the development of user interfaces or integration with already existing systems.

Deliverables

1. Identification of the interfaces required for the intended integration.
2. Development of such interfaces.

6 Experimentation Goal

6.1 Goal

Explore and learn more about the distinctive characteristics of blockchain technology when applied to a Banco de Portugal internal use case.

6.2 Vision

Allow Banco de Portugal to be able to develop, deploy and manage blockchain platforms and solutions autonomously. Also be able to deploy scalable distributed environments for future use cases.

6.3 Proposed Achievements

- Development of a fully functional blockchain solution based on a real use case.
- Development of interfaces to:
  - Interact with the network.
  - Integrate with internal Platforms – e.g.: using OutSystems to build a web application as a front-end.
  - Visualize the state of the network on real time.
- Extend the network to other NCBs.

7 The use case: Securities lending

This experimentation was built upon a business case with requirements that could take advantage of blockchain's unique capabilities such us immutability and the distribution of the responsibility across a network.

7.1 What is Securities lending

The Eurosystem monetary policy implementation has several instruments available, of which, relevant for this use case, are the asset purchases programmes, like APP and PEPP [24]. Under these programmes, securities are purchased by all National Central Banks (NCBs) and by the ECB. They are responsible for publishing the list of securities they own that are available for lending to financial institutions.

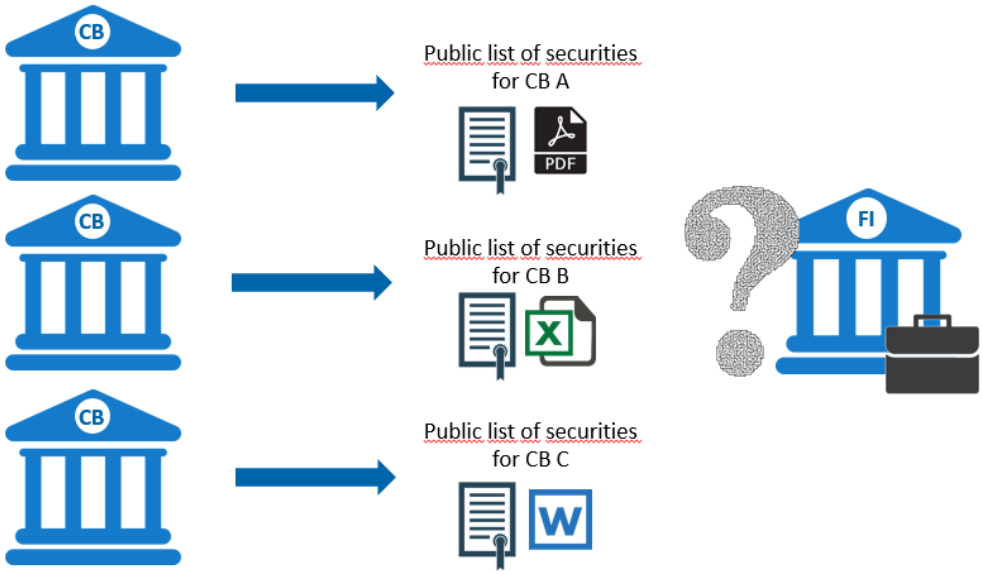

Figure 2: Securities lending As/Is Model (simple view)

This decentralized model has some specific drawbacks:

- For instance, each NCB is free to choose the preferable format to publish the list of securities available for lending. The nonexistence of a standard results in a poor user experience for end users.
- There is no single point of access for the end users to search for intended securities. Such decentralization implies that users may have to perform several searches through different NCBs to find the required security.
- Real time updates are difficult to handle as the lists are currently being updated on a weekly basis. That means that even if a security is available on the list, it may already be lent to another entity.

Such drawbacks result on a pattern that the work group described as *"Search and Try"*, ending on a non-efficient process with a poor user experience.

For current BdP implementation of this program, some additional characteristics were found:

- Third-parties are responsible for holding custody of BdP's securities.
- The lending is performed by those third-parties as long as there is an amount of securities available, interest from clients and the programs limits were not reached.
- BdP does not take part on the negotiation or execution process of a security lending.
- A report is sent by the third-party to BdP by the end of the next day. No internal registration of the securities and the amounts lent is performed.
- The Market Operations team is responsible for updating the securities list on BdP's website weekly. Only public information is published and therefore no amounts are visible to the generic public.
- The current model used by BdP is also used by other NCBs.

Although there might not be a real need to have all information available in real time, as the system could still work as required updating once a day the public information to the list, the sensitive information (like amounts) could still be registered on the system with strict read permissions to each participant NCB, while keeping the public information available to all remaining clients.

BdP's believes that a unified solution could solve the majority of the current drawbacks identified, also allowing each NCB to choose their own model - either using an external entity to take the responsibility for managing the process on behalf of each NCB or having each NCB to assume that responsibility themselves.

7.2 Seclending Chain

Seclending Chain is the blockchain solution proposed by Banco de Portugal to improve the Securities lending process.

The use case has the characteristic of having two distinct life-cycles: one for security and other for lending. The existence of two distinct life-cycles demands a particular attention when analysing the existing business process, as one must take in consideration how both life-cycles interact with each other.

By addressing the security life-cycle, a single unified list is created removing the need for an end user to have to deal with different list structures and standards for each NCB. By addressing the lending life-cycle, it is possible to register on the ledger each transaction regarding the lending of securities, having the list updated in real-time. It's truly important for this use case to succeed that requirements are addressed at the same time for both life-cycles, so that identified drawbacks can be mitigated.

The solution proposed for Seclending Chain results on a distributed ledger that stores all securities owned by each NCB and all the lendings performed. In this way any end user can access each NCB securities portfolio information with just a single query, avoiding the need to execute a query on each centralized system managed currently by each NCB. With this implementation the responsibility for exposing and keep the list updated is shared across the network.

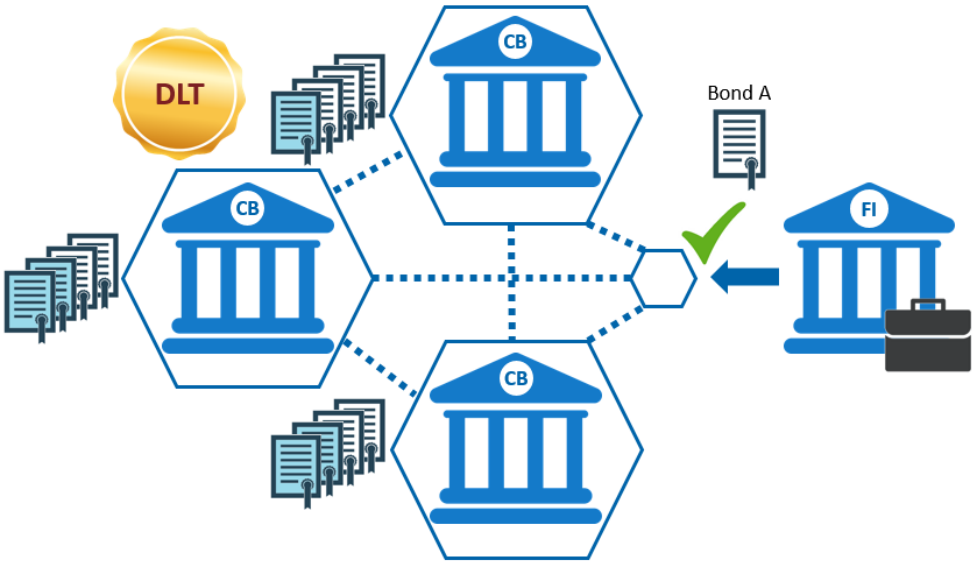

Figure 3: Securities lending To/Be Model (simple view)

7.3 Results

Banco de Portugal successfully identified a use case that could take advantage of the potential capabilities of blockchain technology. The use case is a typical transactional/reference data system comprising the additional challenge of having two distinct life-cycles.

8 Business Model

The first step consisted on exploring the use case within the team, using a collaborative approach for analysing requirements within each life-cycle.

8.1 Security Life-cycle

A total of **four** different states where identified for the security life-cycle:

- **Available**: A security is visible on the list, and it can be lent to end users.
- **Suspended**: A security is kept on the system although not being available for lending. The security is only visible for the owners being removed from the public list.
- **Matured**: When the Maturity Date is reached, the security has finished its lifetime and thus is no longer valid to be lent.
- **Unavailable**: A security is unavailable. This state occurs when it has a 0 amount available for lending.

A total of **six** Security operations were identified:

1. Creation
2. Expiration
3. Suspension
4. Disabling

5. Activation
6. Lending

The Lending operation is the one responsible for linking both security and lending life-cycles. This operation creates a lending transaction and updates the corresponding security status.

8.2 Lending Life-cycle

A total of **three** different states where identified for the lending life-cycle:

- **Lent**: A Security is lent to some counterparty.
- **Matured**: The lending period has ended and the counterparty fully fulfilled the agreed conditions.
- **Default**: The lending period has ended but the counterparty did not fulfil the agreed conditions.

A total of **three** Lending operations were identified:

1. Return Process
2. Default Process
3. Default Regulation

When a Security Lending reaches its lending date the amount of securities lent is returned to the security list. If still valid, the securities become available for a new lending.

More information of the business case is available in appendix B

8.3 Results

The overall business model is represented in the machine-state diagram presented in Figure 4. On this diagram the security's states (yellow), lending's states (green), security's processes (blue) and lending's process (purple) are further identified.

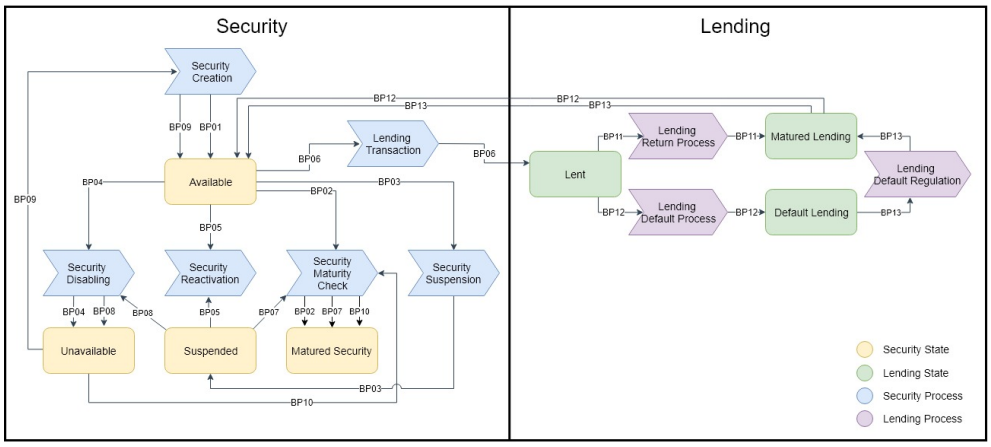

Figure 4: Securities lending Business Model Machine State Diagram

Each connection between each concept represent all possible transactions between the identified states. Each process represents an operation required to be executed in order to guarantee the transition between states, and thus needed to be developed and deployed on the network.

The experimentation resulted on a better understanding on both Security and Lending life-cycle, and how they are inter-connected. It also highlighted the need to have a broader vision on how the assets state changes, as a result of the different business operations identified.

9 Chaincode development

The development of smart contracts, also referred as chaincode on Hyperledger Fabric, started by leveraging the results gathered from the business requirements. Smart contracts were developed using Go[18] language as it was the most commonly used language on this platform and therefore the one with more information available on the internet. It is also the one with the best-known performance.

The chaincode development consisted on implementing for each identified process, a ledger operation that could be executed by any node on the network. Each operation has taken in consideration the current and future status of the asset - security or lending. For the development, the interface *mock shim*[19] from Hyperledger Fabric was used. This interface simulates a network, allowing one to develop and test code without the need to have a network available.

9.1 Main Challenges

Elapsed Time Events

Some of the developed processes are elapsed time events regarding operations on securities and lendings (e.g.: when security's maturity date is reached). The initial intention was for these operations to be executed autonomously by the network.

From this experimentation it was verified that it is not possible to execute natively time-based events on this blockchain distribution. Hyperledger Fabric network architecture is designed so that any chaincode invocation needs to start from a participant of the network, and never from the network itself.

To achieve the needed automation, it is required to have an external service implementing a timer function that automatically invokes the required operation over the network. Being this a distributed network, concerns regarding clock drifts across all nodes must be taken in consideration.

Atomic Transactions

When lending a security, two transactions must be registered into the ledger: one that states the updated security state and another one to create a new lend-

[18] Go is an open source programming language. More in https://golang.org/

[19] Mock class that simulates most of the available network operations. More in `https://godoc.org/github.com/hyperledger/fabric-chaincode-go/shimtest/mock`

ing state confirming that the lending was indeed validated. These transactions must be atomic and linked together, as one cannot occur without the other.

This challenge was not fully addressed during the experimentation, as a significant effort was perceived to analyse and decide on the best solution.

Authentication and Authorization

Some business operations must only be available to some participants of the network. To achieve such segregation, chaincode invocations must take in consideration the user's own identity and associated properties.

On Hyperledger Fabric, the identity of the user is based on digital certificates. The digital certificate comprises a set of customized properties which can be used for further authorization purposes.

For this experimentation a set of groups were created regarding the typology of the identity (e.g.: owner) and the identity's identification (e.g.: BdP). Using this combination, it is possible to enforce rules to ensure that only owners were allowed to create securities and that each owner is only able to manage its own securities. The end user authentication is always ensured by using certificate validation.

9.2 Results

At this stage each business transaction was transformed into a chaincode operation. Both life-cycles could be executed by now, but still requiring an integration layer to further expose each functionality to end users.

As for the capabilities delivered by the platform, a better understanding on how authorization can be applied into the chaincode implementation was achieved, using certificates and metadata. It was also possible to verify that the use of timers is not a native capability designed on Hyperledger fabric.

Future work must also address questions regarding the atomicity of transactions as it is a far more complex challenge required by systems that involve the interoperability of assets with different life-cycles.

10 Setting up the BdP DLT Infrastructure

To further explore blockchain technology, BdP decided to implement its own DLT environment. This decision resulted from the desire to acquire knowledge on how to create, configure and manage such kind of distributed environment.

In this chapter a description of the technology and network topology implemented is given. Additional details such us software and additional configurations are available in appendix A.

10.1 Technology and Infrastructure

BdP leveraged on its Azure Cloud environment to create such a distributed environment. The decision behind choosing the cloud as the operating model

to host this experimentation was the need to have an experimental environment with easier access from and to the internet[20], and thus be able to accept a certain level of exposure.

The configured DLT environment consists of four individual IaaS[21] Ubuntu 18.04 LTS servers: one is configured as a sandbox used to develop and deploy services, and the remaining ones are used to configure all nodes that take part on the network. Even though it is possible to deploy all network nodes on a single server, independent servers were instantiated so that the final deployment could be as closest as possible to a real-life scenario, reducing the gap for future collaborations and needs.

10.2 Network topology

The network topology consists on a single Orderer with two single peer Organisations (PT and EU).

Each peer also hosts a Certificate Authority (CA) and a World State View. With a dedicated CA, each peer can generate and validate their own identities. For the World State View, a CouchDB Database implementation was chosen, allowing each peer to execute more elaborated and advanced queries, adding an extra level of agility to the process.

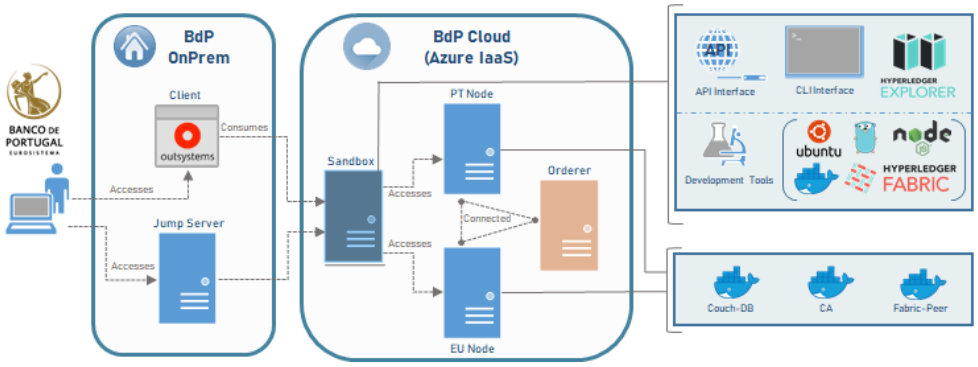

Figure 5: Banco de Portugal DLT Network (simple view)

10.3 Starting a Network

Starting an Hyperledger Fabric network can be briefly described in three steps:

1. Generate and distribute relevant artefacts.
2. Ensure connectivity and start the network.
3. Create a channel and install the chaincode.

Generate and Distribute Relevant Artefacts

In this step, the required artefacts for deploying a distributed network on Hyperledger Fabric platform are generated using the respective binaries.

[20] Internet connectivity was identified as a requirement for future integrations with external partners

[21] Infrastructure as a Service

During this process, it is required to already have a pre-defined network topology which will be the base for the artefact generation. The Artefacts include files like the *docker-compose.yaml* configuration files, root certificates for each Organisations' peers and the genesis block.

The artefacts generation is usually performed on a single server and then the resulted files are distributed among all participants. The distribution process is not restricted which means any channel can be used. The use of email, shared drives or a dynamic FTP server are some possibilities.

Ensure Connectivity and Start the Network

With the assets generated and distributed, each participant will be able to start its service. A core component on any Hyperledger Fabric Network is the Orderer service which is usually the first one to be deploy. The service runs on docker and it can be started by leveraging on the generated docker configuration file. The same is true for each Organisation peer.

During this process it is required to have connectivity between all network services. For that it is required not only to have access to the configured and exposed network addresses (IPs and Ports), but also to have the certificates from all services.

It is during this process that additional components, such as Certificate Authorities and CouchDB databases, are also deployed.

Creating a Channel and Install Chaincode

Assuming all connectivity issues are resolved and the services are up and running, the final step consists on creating a channel between peers and deploy a chaincode on it.

For that, again, the binaries from Hyperledger Fabric are required. The process of creating a channel must be executed by a single peer. After its creation, the same peer can join the channel, install, and instantiate a chaincode[22]. By the end of this process, the network has a channel available with a chaincode running. A ledger is associated to this channel with already two transactions created: one for the network configuration with all Organisations' peers and its metadata, and another stating the information of the peer that that successfully joined the channel. From this moment, any Organisation can join this channel and install the respective chaincode locally. After this process, any chaincode invocation will trigger the ledger update, replicating it to this new peer.

For new Organisations to be able to have access to the channel, the already running network must first update the configuration block by adding this new Organisation information to it[23].

[22]Install a chaincode is a process done locally while instantiate consists on deploying the chaincode on the channel

[23]More information is available in section '12.2 Extending the Network'

10.4 Results

A complete environment for developing and deploying blockchain solutions was successfully configured. The environment was ready for deploying a two single peer Organisations network with a single Orderer.

Further work should be developed in order to achieve a more comprehensive solution for logging events of the environment as a whole, instead of using the native decentralized logging model currently implemented. Also, additional work is required to define the best model for distributing the generated artefacts among the participants of the network.

11 Integration

In this stage the goal was to explore better interoperability scenarios between business users and the blockchain solution developed. For that it was decided to implement three distinct types of interfaces:

- A Command Line Interface (CLI) to interact with the network through the available SDK.
- A REST API interface to abstract the platform operations allowing the integration with other applications components.
- A Web Interface to interact with the platform through the previously implemented REST API interface.

11.1 CLI and REST Interface

Both CLI and REST API interfaces were developed leveraging on the SDK for NodeJs[24] available in Hyperledger Fabric. This specific SDK was chosen for being the most commonly used by Hyperledger community.

For the development of both interfaces a two-layer architecture was developed comprising an abstraction layer that exposes some SDK endpoints. As a result the development of each interface was focused on the interface layer exclusively. For the REST API the SWAGGER specification was used for better integration capabilities.

The process of authentication was also abstracted trough the developed abstraction layer. In this layer the user identity is validated through a specific Certificate Authority of a given peer and, if valid, the user is allowed to execute the interfaces.

For the REST API, as it is rarely used as a user interface, a local repository was setup to configure local identities where the user also needed to be registered. In order to ensure an end to end authentication flow, the local identity username needed to match the username's identity registered on the Certificate Authority. This model gives a certain level of confidence that the identity authenticated to the REST API is the same executing each operation on the network. Otherwise, there is a risk of having identity impersonation when executing chaincode on the network. Nevertheless, additional work is

[24] `https://hyperledger.github.io/fabric-sdk-node/release-1.4/index.html`

required to study the possibility of using the end user identity through the whole pipeline.

On this setup each interface instance is coupled with a single peer. Further work needs to be done in order to understand how to create a more resilient connection between interfaces and the network.

11.2 Web Interface

The web interface was implemented, leveraging on the capabilities of a low code platform for faster development and iterative deployments – OutSystems – and also the developed REST API. This approach followed the initial goal to integrate the blockchain platform with the current BdP's enterprise ecosystem and technology stack.

11.3 Hyperledger Explorer

Hyperledger has an explorer solution available called Hyperledger Explorer. Based on APIs natively available in an Hyperledger Fabric network, this solution exposes a dashboard with information from network components, ledger status, blocks' statistics and its content, among other functionalities. The whole solution is built on NodeJs and requires a direct connection to a network peer, being the information exposed the one the peer has access to.

Monitoring the network is also a very relevant requirement for knowledge improving as it allows to learn the networks own behaviour by visualizing each update in real time.

11.4 Results

Three integration components were developed: a CLI, a REST API and a Web interface, allowing the evaluation of interoperability scenarios within BdP's enterprise ecosystem. A native network explorer service was also configured, delivering valuable insights on the network state.

During the implementation of those interfaces, concerns regarding user authentication and authorization were analysed. For that, Bank of Portugal came with a solution where the user identity consuming the interface and the identity invoking the chaincode are mapped through the same username.

Future work might be required to analyse the use of other authentication and authorization models, aligned with the more advanced patterns for Identity and Access Management - like OAuth or SAML. It is also relevant to consider the need and the feasibility of using a single identity through the whole pipeline. The goal is to better understand on how to avoid impersonation on chaincode invocations through malicious interfaces.

Currently, each developed interface is coupled to a single node, and the respective CA. Further work is also required to create a more resilient scenario. For instance, an interface could have a connection to several peers from different Organisations.

12 Extending the Network

Following the internal experimentation conducted by BdP in creating and deploying a DLT network with a relevant use case, other NCBs from the ESCB community were challenged to further extend the developed use case.

The De Netherlands Bank (DNB) and the Oesterreichische Nationalbank (OeNB) responded positively to this call to action which resulted in the beginning of a joint experimentation. This collaborative work was designed for two iterations: the first focused on starting a network with predefined Organisations and the second on extending the network with new ones.

12.1 Start a DLT Network

Goal

This iteration's goal was to deploy the Seclending Chain chaincode on a DLT network composed by single peer Organisations from both DNB and BdP.

Architecture

Both Banco de Portugal and De Nederlandsche Bank leveraged on using Azure Cloud IaaS services to set up their network nodes, supported by the Ubuntu operating system. All nodes had the same DLT capabilities, delivered through the use of Docker images.

The network is composed by three single peer Organisations (PT, EU and NL) with a single Orderer. The Orderer, PT and EU peers are hosted by BdP, being the NL peer hosted by DNB. The network's topology is identical to the one used by BdP on its internal experimentation but now with one additional Organisation hosted by DNB.

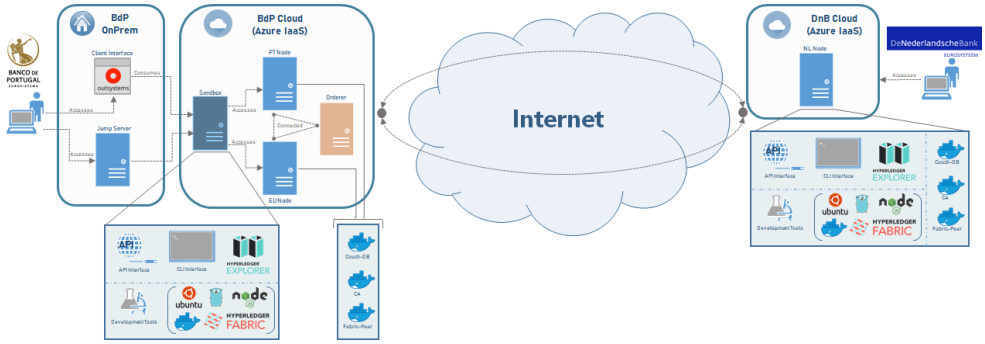

Figure 6: DLT Joint Experimentation Architecture

Improvements from previous Work

Previous scripts developed initially by Central Bank of Portugal were improved during this joint experimentation. The process of starting the network was performed successfully within a very reasonable effort and time frame[25]. The following tasks were addressed:

[25] A total effort of approximately 12 hours of joint work

- Ensure Connectivity between nodes.
  - Setup Orderer and nodes from BdP and DNB.
  - Create a Channel and join it.
- Setup the Securities lending use case:
  - Install and Instantiate the Seclending Chain Chaincode.
  - Deploy and configure the interfaces (CLI, REST API and Hyperledger Explorer).
- Test the network by creating both BdP and DNB Securities.

The most significant effort in this step was related to network and interface configuration. As it was decided to perform the configuration manually, it was common to end up with some configuration mismatched. Automating the configuration process reduced significantly the effort. By the end, a successful setup of a running network with three single peer Organisations (two from BdP and one from DNB) was achieved. Interaction with the network was also possible using the available interfaces.

12.2 Extend a running Network with new Organisations

Goal

The Goal of the second iteration was to extend the running network from the previous iteration with new Organisations, avoiding the need to start the network from scratch.

Architecture

The second iteration of this joint experimentation was conducted in closed collaboration between BdP, DNB and OeNB. The network architecture was an extension of the one used on the first iteration, but now onboarding a new NCB Organisation - Central Bank of Austria.

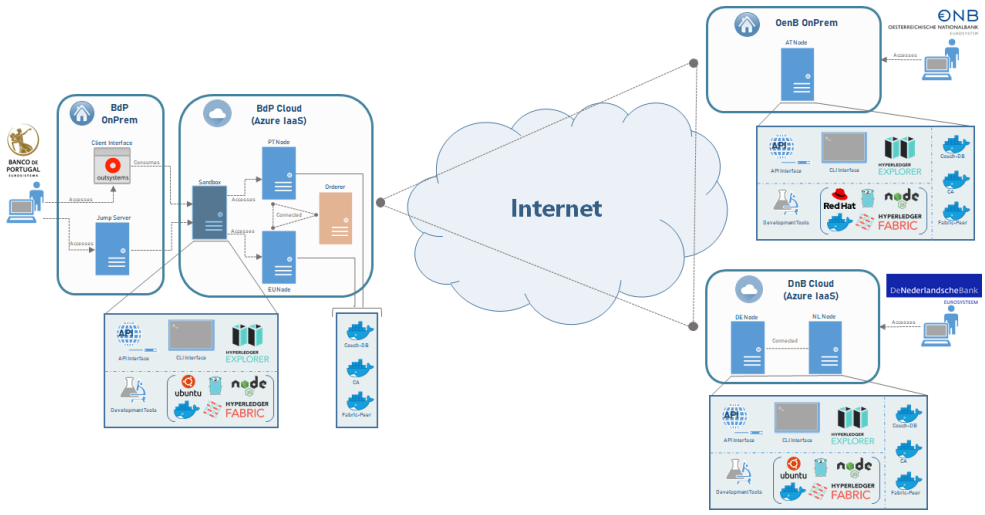

Figure 7: DLT Joint Experimentation Architecture (BdP, DNB and OeNB)

Central Bank of Austria decided to host the node representing this new Organisation on its own internal infrastructure and chose the Red Hat operating system to run the node.

The network topology chosen for this second iteration was composed by five single peer Organisations (PT, EU, NL, DE and AT): two hosted by BdP, two hosted by DNB and one hosted by OeNB. The network counted also with a solo Orderer service, hosted by BdP. It was decided to start the network with each NCB hosting a single Organisation - three nodes - and later extend it with additional Organisations from BdP and DNB. The diagram of the architecture is available in Figure 7.

Improvements on previous Work

On Hyperledger Fabric, the network composition is defined on the ledger itself, as a transaction that states which Organisations belong to the network, their peers and the corresponding metadata.

The process of adding a new Organisation to the network requires the retrieval, extraction and update of this configuration block by adding this new Organisation's information to the ledger. After updating the block, it is required to create and submit a request for a new transaction to the network. For the submission to be successful, the submitting node must gather the required amount of signatures[26] from the remaining network's nodes in order to fulfil the network's consensus policy. If consensus is achieved, the block is accepted and added to the ledger as a new configuration of the network. From this point, the new Organisation is now considered part of the network.

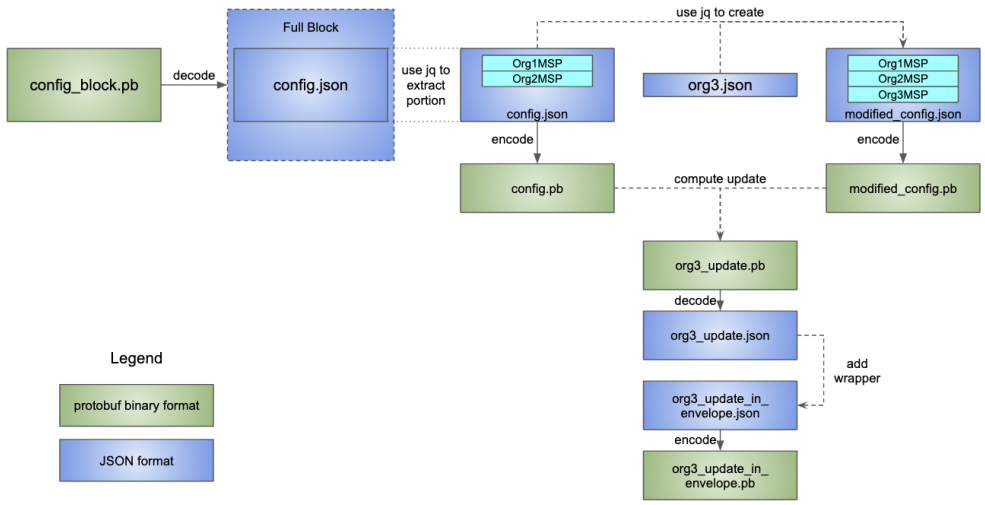

Figure 8: Process of updating the configuration block with a new Organisation. Image from Medium [25]

After being configured on the ledger, with the configuration block updated, the new Organisation needs to join the already existing channel and install the corresponding chaincode. In order to be able to endorse transactions

[26]The default policy is majority

on that channel, a chaincode upgrade needs to be performed. The upgrade allows to update the endorsement policy including the new Organisation as an endorsement peer. The chaincode upgrade must be performed by one of the existing Organisations as the new one does not have permission to execute such an operation.

The process of extracting and updating the block is an error prone procedure, requiring several sequential transformations manually applied (check figure 8). For that, BdP decided to improve and extend the automated scripts in order to reduce the effort required to extend the network. The updated version of these automated scripts, resulted in a faster deployment of the network by significantly decreasing the required amount of manual interventions during the network and interface configuration.

Initially the network was composed by three distinct Organisations (PT, NL and AT). For the first extended Organisation (EU, hosted by BdP) it was required to have a signature from two of the three Organisations. After adding this new Organisation to the network, and for the next Organisation (DE, hosted by DNB) to be accepted, three signatures were required, as now the network is composed by four distinct Organisations (PT, NL, AT and EU). Again, it is important to state that the policy for updating the network with a new Organisation is completely different from the endorsement policy, which is associated to a chaincode version on a given channel.

12.3 Results

At the end of the two iterations a successful deploy of the Securities lending use case was achieved in a distributed network composed by three different NCBs. Additionally, it was possible to validate and further automate the process of extending the network to new Organisations, confirming the possibility of such requirement.

Knowledge on how consensus works for network updates was also acquired. Although not verified, the same procedure may be applied to remove an Organisation from the network by removing its associated configuration from the network's configuration block.

The time and effort spent in both iterations revealed that automating the process, in an incremental way, works also as a learning process[27]. The progressive evolution on automating the scripts clearly decreased the time required to start and further extend a running network, when comparing with values gathered from the initial setup.

13 Takeaways

The deployed DLT network evolved from an internal experimentation by Banco de Portugal to a more collaborative joint experimentation with both DNB and OeNB. From this joint experimentation it was possible to learn on

[27] The whole iteration took around 8 hours of joint effort for both starting a network, and extending it with two new Organisations. These values do not take in consideration the effort required for the initially configuration of the nodes

how to develop, deploy, manage and extend a DLT network developed on Hyperledger Fabric platform.

The design of a framework to be used for this experimentation played an important role in achieving the purposed goals. Techniques like the creation of a state-machine diagram highlighted the importance to address the requirements of both life-cycles at the same time. This was a crucial step forward because it allowed to translate business requirements into a more transparent, easy and understandable approach.

Producing the documentation for the executed procedures was also seen as a really important deliverable. Such documentation was the main tool used to share the work done in collaboration with both DNB and OeNB.

The process of creating and starting an Hyperledger Fabric distributed network may seem overwhelming due to the great amount of different configurations and distinct steps required between the Organisations involved.

The best place to start is with Hyperledger tutorial "Build First Network"[28]. Using this tutorial, it's feasible to start a simple network, with multiple nodes on a local machine, from scratch. It is then easier to leverage on the acquired knowledge to use the same approach and reproduce the same results on a more complex network deployed across distinct servers.

When moving from a single local-machine environment to a multi-server configuration environment, the complexity of managing and deploying the network increases. Such complexity involves challenges with connectivity, as there is the need to ensure that all nodes are able to communicate and share the generated assets.

From the second iteration of the joint experiment, extending the network to new Organisations become no longer an unknown field. The process tested, although not as straight as starting the network, is simple enough to understand and execute.

For the joint experiment, the time and effort required to start and extending the network, ensuring connectivity and finishing script automation, is a good indicator that the created process is easy to replicate for further Organisations and may be extendable to other use cases. This was only possible because the decision to automate the whole process, in an incremental way, also worked as a learning process, thus reducing the need for manual intervention and resulting on less configuration errors.

The development of smart-contracts using Hyperledger Fabric platform revealed to be quite straightforward. Despite the lack of experience using the Go programming language, there is plenty of information available on the internet that can be used as a guide trough the development process, including Hyperledger own code samples. The existence of a mock-up interface of the network - *mock shim* - was also helpful for a faster development.

During the chaincode development some challenges remained unaddressed:

1. How to achieve an atomic transaction when two ledger updates are required.

[28] `https://hyperledger-fabric.readthedocs.io/en/release-1.4/build_network.html`

2. How to ensure end-to-end authentication and authorization of an entity when using that identity to invoke smart contract operations.

In this experimentation the authentication and authorization process were somehow approached by leveraging identities with digital certificates and metadata. The solution has still space for improvements and could be further aligned with modern authentication frameworks and protocols [29].

One important limitation identified was the lack of native support for time-based events, as all transactions must start from a node to be accepted by the network. For the Seclending use case this was an important requirement, as the use case was initially designed to take advantage of concepts around event-driven architectures, aligned with the need to analyse, at the same time, both Securities and Lending life-cycles.

Nevertheless, operations involving time on distributed systems are always an implementation challenge as small time deviations between different nodes on the network may occur, ending up in discarding valid transactions.

The integration of the blockchain solution with the external environment turned out to be fairly easy. This was due to the availability of an SDK with the required API endpoints that allowed a successful interaction with the network. The decision to use a two-layered architecture model, and reusing the implemented abstraction, resulted in a faster development, as both the CLI and REST API interfaces integrate seamlessly with the network.

The integration of the Web interface with the network didn't present a challenge as it was a typical web integration process by leveraging on the developed REST API. The use of the SWAGGER framework also enabled the automatic import of all available endpoints, allowing the web application to reuse all contract's definition.

One drawback identified was the exclusive connection between the interfaces to a single peer on the network. For a pure distributed system such implementation seemed very limited.

The deployment of a network visualizer - Hyperledger Explorer - had also some initial challenges to get the correct configuration working. After setting up the desired configuration, the solution delivers what it promises, although a more real time approach would be an important improvement.

One of the most relevant characteristics of blockchain is immutability. In Hyperledger, such property is only intrinsic to the ledger, as the World State View used can be easily tampered by the owner of the hosting server.

Nevertheless, tampering the World State does not have any direct impact on the network itself, as this system of records is used to guaranty a faster access to the ledger state (both from user interfaces and smart contract execution). Such tampering does not affect the network's results as the operations in the end are always validated trough the information available on the ledger. Nevertheless, further work must be done in order to perform a broader analysis on the real impacts of tampering the World State View and also on the possibility of a single node to tamper its own ledger, and the implications on the network.

[29] E.g.: OAuth 2.0, SAML 2.0 and OpenID Connect

14 Next Steps

Following the learnings from the experiment, further exploration must be addressed on the following topics:

- Consensus:
  - System behaviour with different endorsement policies.
  - Orderer Role in consensus implementation.
- Privacy and Confidentiality
  - Usage of sensitive information.
  - Information encryption and signature with the owner's own key.
- Authentication and Authorization:
  - Modern Identity and Access Management paradigms.
  - Use of the user's identity through the whole pipeline of a request, including chaincode's execution.
- Scalability, Performance and Resiliency of the Network:
  - Network behaviour under heavy load.
  - Tampering the World State View and the Ledger on a single node.
- Automation and scripts improvement.

Some of these topics were briefly addressed on this experimentation but additional work is required to better understand these novel technology capabilities and its limitations.

15 Blockchain: The Good the Bad and the Ugly

Blockchain, although not being a completely new technology, has recently raised a clear interest on different activity sectors and business models. It begun in the payments domain with bitcoin but it is scaling to other domains such us health, retail tracking or even games.

The fact that blockchain started so famously created a huge hype around it, creating the idea that this technology has the power to solve every single problem on every enterprise. Such interest resulted on high waves of investment to explore this novel technology in searching for the next big thing.

This rapid growth created high expectations that end up not being totally fulfilled. Blockchain, as any technology, it is not suitable for any use case.

There is still space for blockchain to be relevant and bring innovation to use cases. More than just a hype, it has real benefits and the potential to improve the way information is managed, removing the control from centralized entities to purely distributed networks. But again, the use case must be carefully chosen as blockchain is not a one size fits all technology.

For the years to come, new applications involving new use cases will emerge, potentially disrupting some business models. The need to have advertise the use of blockchain technology will fade away and users will start using applications supported by this technology without knowing it.

It is also believed that blockchain has the capability of changing the internet as we know it. We may see a move of having our information on secure communications (HTTPS), to being secure and distributed (e.g.: HTTPB), indicating that the information being accessed, at the end, is totally controlled by the end user, leveraging on Blockchain as the single source of truth and trust.

16 Acknowledgments

Special thanks are due to the Central Bank of Netherlands and the Central Bank of Austria for accepting the challenge and the call to action. The constant flow of motivation, the sharing of experiences and their knowledge demonstrated by practicing on the field have made the final joint experiment a real success, contributing significantly to our initial goals and expected outcomes.

Appendices

Appendix A BdP DLT Infrastructure

A.1 Software and Configuration

Sandbox server

- Ubuntu 18.04 Lts
- Hyperledger fabric1.3 binaries
- Golang 1.3
- Nodejs 8.5
- JQ

Additionally, and in order to be able to have access to a browser interface, a GUI[30] package was installed on the sandbox server. this allowed to have access to application with graphical interfaces such as the developed Rest API and the Hyperledger Explorer.

Network nodes

- Ubuntu 18.04 Lts
- Hyperledger fabric 1.3 binaries
- Docker and the following docker images
  - Orderer
  - chaincode
  - couchdb
  - peer
  - Certificate Authority

A.2 Logging and Debugging

As the whole network components run on top of Docker containers, the logging systems is also based on this technology. Each container has a log file which can easily be accessible through the command *'docker logs {container-id}'* and contains the whole logging messages.

Appendix B Business Exploration

B.1 Entities

The following entities take part on Securities lending Business process:

- **Owner**: Entity who owns the security and has the responsibility to list it and lend it to their end-users.
- **Lender Player**: The entity responsible for process the lending of a given security. In some cases, the lending process is done by the owner which has in such situations the role of lender player.

[30] Graphical User Interface

- **Counterparty**: The entity that borrows the security.
- **Regulator**: The entity responsible for the regulation of the system.
- **Negotiator**: The entity responsible for negotiating the terms of a transaction. The role is typically the same as the Lender Player.

B.2 Security Properties

- ISIN
- Status (Available, Suspended, Matured, unavailable)
- Amount
- Owner
- Lender Player
- Negotiator
- Interest Rate
- Maturity Date

B.3 Lending Properties

- Transaction ID
- ISIN
- Owner
- State (Lent, Matured, Default)
- Counterparty
- Lender Player
- Negotiator
- Collateral
- Amount
- Fee
- Interest Rate
- Settlement Date
- Maturity Date

Appendix C Scripting and Automation

C.1 Scripting and Automation

The process of configuring the network can be costly and very error prone as it evolves considerable configuration files and requires the execution of several sequential steps.

In order to reduce the error rate and thus improve the velocity from where one can start and extend a network from scratch, Banco de Portugal developed automation scripts which abstracts the whole process of setting up the network. By creating the automatic process it is also guaranteed the correct reproduction of results.

The automation process includes:

- Single Configuration file with the whole network configuration for both peer and Organisations
- Generating network artefacts such as:

  - Cryptographic assets for all nodes
  - Docker compose files for containers based on pre-defined templates
  - Configuration files for all interfaces
  - Configuration files for new Organisations (network extension)
- Start up docker components
- Create and join channels
- Install and instantiate Chaincode
- Create and manage identities within a CA
- Extend the network to new Organisations

At the moment, without taking in consideration the time required to ensure connectivity among all nodes, it was possible to start a network from scratch in less than an hour including starting up the developed interfaces.

At the moment the bottleneck of the whole process consists on the distribution of the generated assets. This process still requires a considerable effort as it remains manual, performed in a not very agile way.

Acronyms

API Application Programming Interface.

BdP Banco de Portugal.
BFT Byzantine Fault Tolerance.

CA Certificate Authority.
CLI Command Line Interface.
CRUD Create, Read, Update and Delete.

DLT Distributed Ledger Technology.
DNB De Nederlandsche Bank.

ECB European Central Bank.

FTP File Transfer Protocol.

HLF Hyperledger Fabric.

ICO Initial Coin Offer.

NCB National Central Bank.

OeNB Oesterreichische Nationalbank.

SAML Security Assertion Markup Language.
SDK Software Development Kit.

Glossary

51% Attack When an entity or group controls more than half of the computing power of a crypto-asset, and thus, have higher probability to control the next block to be registered on the network.

API An application programming interface that exposes a set of services for others to consume.

Bitcoin Crypto-asset proposed by Satoshi Nakamoto on 2008.
Blockchain DLT implementation where the ledger is composed of chains of block of data, cryptographically connected and distributed across peers.

Certificate Authority Entity responsible for managing and issuing digital certificates.
Consensus Algorithm by which the distributed network agrees which block should be added next.
Corda Open source blockchain platform focused on financial enterprise solutions.
crypto-asset Digital asset based on cryptography, design to be used as a medium of exchange.

Digital Signature A cryptographic operation where a digital code is generated using a digital certificate, ensuring the identity of the user and the integrity of the message.

Distributed Ledger Technology Technology where the database, also known as ledger, is shared across several entities, each one with its own copy.

Double Spending When an entity uses the same digital asset on two distinct transactions.

Ether Crypto-asset associated to the Ethereum platform.

Ethereum Open source blockchain platform where it is possible to implement smart contracts for Distributed Applications (DApps). It is also a known crypto-asset along with bitcoin.

Fintech Computer programs and technologies used to support banking and financial services.

Genesis Block Initial block on a Blockchain network.

Hash Cryptographic function that transforms any given type of information, no matter its size, into a fixed length unique identifier.

Hyperledger Multi-project open source collaborative effort hosted by the Linux Foundation.

Hyperledger Fabrics Specific project from Hyperledger which offers a modular platform for the development of blockchain applications.

Ledger Digital system of records of a DLT network.

Mining The process of adding a new transaction to the ledger on Bitcoin's by solving a computational demanding challenges (PoW).

Node Single point on the network associated to an entity or Organisation.

Organisation Abstraction used to identify the entities that are part of the network.

OutSystems A Portuguese platform for low code applications development.

Peer Instance of nodes for the same organisation.

Permissioned Network Networks where only authenticated identities can take part of the network.

Permissionless Network Networks where any identity can be part of the network and thus submit and validate transactions.

Proof of Stake Mechanism alternative to the Proof of Work where instead of computer power a stake is used.

Proof of Work Mechanism implemented on certain blockchain networks as a security measure where each transaction must require a certain amount of work.

Smart Contracts Digital piece of code that implements the business rules by which the DLT network should run.

Token A digital representation of an asset which can be transact on a blockchain Network.

References

1. Andrea Pinna and Wiebe Ruttenberg. Distributed ledger technologies in securities post-trading. *ECB Occasional Paper*, (172), 2016.

2. Svein Ølnes, Jolien Ubacht, and Marijn Janssen. Blockchain in government: Benefits and implications of distributed ledger technology for information sharing, 2017.

3. Peter Thiel. Blockchain 3.0 the future of dlt? `https://cryptoresearch.report/crypto-research/blockchain-3-0-future-dlt/`.

4. M Frans Kaashoek and David R Karger. Koorde: A simple degree-optimal distributed hash table. In *International Workshop on Peer-to-Peer Systems*, pages 98–107. Springer, 2003.

5. M Divya and Nagaveni B Biradar. Iota-next generation block chain. *International Journal Of Engineering And Computer Science*, 7(04):23823–23826, 2018.

6. Zhe Yang, Kan Yang, Lei Lei, Kan Zheng, and Victor CM Leung. Blockchain-based decentralized trust management in vehicular networks. *IEEE Internet of Things Journal*, 6(2):1495–1505, 2018.

7. Nitish Singh. Blockchain vs database: Understanding the difference between the two. `https://101blockchains.com/blockchain-vs-database-the-difference/`.

8. Arati Baliga. Understanding blockchain consensus models. In *Persistent*. 2017.

9. Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

10. Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital supply chain transformation toward blockchain integration. In *proceedings of the 50th Hawaii international conference on system sciences*, 2017.

11. JP Morgan. Jp morgan creates digital coin for payments, 2019.

12. Eurochain. Exploring anonymity in central bank digital currencies. *European Central Bank - IN FOCUS*, 4, 2019.

13. Ori Jacobovitz. Blockchain for identity management. *The Lynne and William Frankel Center for Computer Science Department of Computer Science. Ben-Gurion University, Beer Sheva Google Scholar*, 1:9, 2016.

14. Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. *asdas*, 2008.

15. Fahad Saleh. Blockchain without waste: Proof-of-stake. *Available at SSRN 3183935*, 2019.

16. Joao Sousa, Alysson Bessani, and Marko Vukolic. A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform. In *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pages 51–58. IEEE, 2018.

17. Wouter Penard and Tim van Werkhoven. On the secure hash algorithm family. *Cryptography in Context*, pages 1–18, 2008.

18. Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310, page 4, 2016.

19. Hyperledger Fabric. A blockchain platform for the enterprise - hyperledger fabric - key concepts. `https://hyperledger-fabric.readthedocs.io/en/release-1.4/index.html`.

20. Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

21. Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm (extended version), 2013.

22. Richard Gendal Brown, James Carlyle, Ian Grigg, and Mike Hearn. Corda: an introduction. *R3 CEV, August*, 1:15, 2016.

23. Martin Valenta and Philipp Sandner. Comparison of ethereum, hyperledger fabric and corda. *[ebook] Frankfurt School, Blockchain Center*, 2017.

24. European Central Bank. Securities lending of holdings under the asset purchase programme (app). `https://www.ecb.europa.eu/mopo/implement/omt/lending/html/index.en.html`.

25. KC Tam. Add a new organization on existing hyperledger fabric network. `https://medium.com/@kctheservant/add-a-new-organization-on-existing-hyperledger-fabric-network-2c9e303955b2`.