Computing Science and Mathematics
University of Stirling

Smart contracts on Hyperledger Fabric

Yordan Gospodinov

Supervised by Dr. Andrea Bracciali

Submitted in partial fulfilment of the requirements for the degree of
BSc with Hons in Computer Science

April 2019

” If you are not making someone else’s life better, then you are wasting your time ”
-Will Smith

Abstract

This project aims to research blockchain technology and its implications in enterprise fields. Blockchain is capable of securing information, while also certifying for its integrity inside the network. Hyperledger Fabric is specifically made to answer the requirements enforced by the business.

An opportunity arised for developing the idea of certifying the identity of a person due to the security and integrity provided by blockchain. The focus of this research is to give a better procedure for identifying an idividual. To secure the private information of a person, when they are providing authorization details. What's more, can an identity be assigned quickly and used worldwide.

The scope of this project was to research the blockchain framework Hyperledger Fabric and the smart contract, called chaincode in this permissioned network. With the knowledge gathered an attempt of self-sovereign identity prototype was made.

The field of blockchain is still in its infancy but it is developing rapidly. At the start of this project, Fabric was at version 1.2 and the project kept its development consistent with it. However, at the time of handing in of the dissertation, Fabric was already at version 1.4.

Smart contracts represent the business logic in a blockchain network. In Hyperledger Fabric they are called chaincode. It runs on top of the blockchain to implement the desired interaction with the ledger. It can be written in two ways, the first is using only Fabric and writing low-level code (golang recommended). The second, which is used in this thesis, is through Hyperledger Composer. Composer is an open development toolset and framework to make developing blockchain applications easier. It creates assets, participants and transactions through high-level code, that is compressed into a single file and installed on top of Fabric blockchain network.

Attestation

I understand the nature of plagiarism, and am aware of the University's policy on this. I certify that this dissertation reports original work by me during my University project except for the following (adjust according to the circumstances):

- The code discussed in section 3.5 was created by Hyperledger Fabric (*https://hyperledger-fabric.readthedocs.io/en/release-1.4/getting_started.html*) and was used in accordance with the licence supplied.
- Figure 1 was taken from *https://blockgeeks.com/guides/what-is-blockchain-technology*.
- Figure 2 was taken from the official website of Docker
- Figures 5, 6, 7 were taken from Fabric documentation.

Signature: **Date:**

Acknowledgements

First and foremost, I would like to thank to Dr. Andrea Bracciali for introducing me to blockchain. What's more I am grateful for his patience, guidance and support, without which I would not have been able to grow into what I am today. Thank you.

I am grateful to my parents for giving me the opportunity to study in University of Stirling.

I am grateful to my classmates for helping me out, when I could not understand something, and just for being around, it was fun.

I would like to thank the library for having the nice quiet, fourth floor, and for always buying the books I requested.

Contents

List of Figures

Chapter 1

Introduction

Blockchain is a newly-emerging, distributed and decentralized database of blocks constructed of transactions. It is of great interest as it has potential to change many aspects of life. It is an opportunity for better authenticity and security of data.

Hyperledger Fabric is a blockchain framework designed, specifically, to be easily adaptable by different business use cases.

The main two objectives of this project are: to research how the smart contracts in Hyperledger Fabric work; to make a prototype of self-sovereign system with the gathered knowledge.

1.1 Background and Context

Information has always been one of the most valuable assets a person could have. Through times information was traded in many ways, from barter to monetization. Recently, personal information has become a great selling point, because it can be used in variety of fields, from science to business. However, the collection of this data is becoming an issu.

As individuals, our identities are, to some extent, not ours anymore. If we cannot certify our identity, we became no one in the eyes of business and government. Had we lost all of the documents that certify our place in the company, city, country, Earth, we would be in a big trouble. [2]

Another approach to critical and private information is how it is being used live. Whenever we want to identify ourselves somewhere, the usual document for identification would be either an ID or a passport. Here is the problem concerning all information on this document. It turns out that whenever a person wants to prove their existence, the party that requires this identification, can take and keep a record of all sensitive data on that document. In some countries this may be illegal as this data could be used for a wicked purpose. [1]

Furthermore whenever a person is signing in to receive a certificate they are leaving sensitive data with this company. In most countries, whenever a person starts living in a city, they must identify himself/herself to the council. In the end, there are a lot of institutions that keep sensitive data for an individual. This is a problem, because institutions and businesses have different levels of security. An attacker only needs to pick the easiest target and a great deal of sensitive data will be exposed, as it has been done before [29].

I believe all of these problems are just subproblems of bigger challenges - what is an identity today and how to be able to give private access to our data. The solution could provide us awareness for a better control of our own data, and also enable us to share only what is exactly needed to provide to those companies and institutions.

1.2 Scope and Objectives

The scope of this thesis will involve Blockchain technology and what is digital identity, using Hyperledger Fabric. Fabric is a permissioned blockchain modular framework developed specifically for businesses.

Being a modular framework, a lot of the scope will involve resolving how customizable Fabric can be. To be personalized is essential for the creation of a good system. The other main features to be examined are the scalability and usability of this blockchain framework.

Research-acquired knowledge will be implemented in a prototype program as the final part of the project. A model of self-sovereign program with a focus on the security and authentication provided by the blockchain framework. This work aims to present advantages of the decentralizing element that can save resources and protect the personal data of the end-user. Finally, the project will focus on how the ledger provides data confidentiality confidence to each of the parties.

The objectives of the project include:

- Understanding how Hyperledger Fabric work;
  - Installing all prerequisites;
  - Installing Hyperledger Fabric;
  - Running a simple network with 2 organizations;
  - Trying out how the chaincode (smart contracts) work;
  - Trying to install and control newly added chaincode on a running system;
  - Installing Hyperledger Composer;
  - Running Composer as a substitute to Fabric chaincode.
- Building a fully functional Fabric blockchain with several different parties;
- What an ID is and identity and how it is defined in the digital world;
- Deeper understanding of self-sovereign identity, what it is and how it should/could be best defined in a blockchain platform in order to be used genuinely and without misappropriation;
  - Trying out different configurations on Composer;
  - Trying out different chaincode functions, to find out the best for the use case.
- Building a prototype of self-sovereign identity system.

1.3 Achievements

The project has been concluded with several milestones reached :

- Running Fabric blockchain with two organizations
- Obtained knowledge how to upgrade a running system
- Obtained knowledge about Self-sovereign identity
- Obtained knowledge about Hyperledger Composer and Fabric
- Building a fully functional Fabric system with Composer business network

A range of smaller achievements were reached. Deeper understanding of what distributed ledger technology is and its applications. Ultimately, a step forward to a career in the Blockchain field.

1.4 Overview of Dissertation

The first six chapters will discuss the reasoning behind selecting Fabric and its applications, stressing on the case about securing the identity of a person and keeping their private data private.

The first few chapters provide insight into what blockchain, Hyperledger Fabric and Composer are, while also giving some examples of working Fabric systems. Thereafter is the chapter about the process of developing a prototype of self sovereign identity. Reasons and graphical representation of the workflow will illuminate the design choices.

Lastly, the dissertation concludes with evaluation of the work done and provide ideas for future development of the prototype.

Chapter 2

State-of-The-Art

This chapter touches on the technical part of distributed ledger technologies, blockchain and cryptography. Furthermore, it highlights a few examples constructed with Fabric blockchain.

2.1 Technical Background

2.1.1 Distributed Ledger Technologies

Ledgers have been in use of the humanity since ancient days. Their medium has been clay, wooden tally sticks, stone, papyrus and paper. They served its purpose as a one-side record-keeping tool. But this also brings concerns around who is going to validate this one-side register. Later in 15th century, the Italian mathematician Luca Pacioli became the first person, recorded, to publish a paper on the double-entry bookkeeping [10].

However, even when all the parties have their own records of deal, someone, or even a group of the participants, may cheat and keep a different record, from the original. Thus, taking advantage over the people who are trying to trade fairly and honorably.

A DLT is a concurrent system, referring to a database that is consistently shared and synchronized across multiple machines/nodes in a network. It allows transactions to be monitored by multiple actors, thereby making a cyberattack more difficult. The participants at each of those machines can access the recordings shared and can keep an identical copy of it. Since it is a distributed ledger, any changes made on it are then reflected and the change is done to all the nodes holding a copy of it [8] . However, in order to know which entry should be spread, and which is/are the correct ledger/s the system must have a consensus among all the peers and reach a final solution.

Consensus

In general, a consensus algorithm is a process in computer science used to achieve agreement on a single data value among distributed processes or systems. Consensus algorithms are designed to achieve reliability in a network involving multiple unreliable nodes. Solving that issue known as the consensus problem is important in distributed computing and multi-agent systems.[22]

Information Sharing

DLT has the ability to maintain tamper-resistant records and the arrangements could be designed to allow participants to have read-only access to certain parts of the common ledger. This decision, even if it limits the options a user can have, gives visibility into the system, because it is easier to follow the changes in the states of the blockchain. Furthermore, it stresses the integrity of the system, since the user is able to see the supply chain of a particular asset or its history through the ledger. At the same time, not all transactions are concerning the customer. That is why, they can be divided into one or more ledgers. However, certain regulatory requirements could be difficult to meet by simply providing access to a ledger.[22]

2.1.2 Blockchain

Blockchain represents several ideas that are now able to work together. In its core, this high tech is decentralized database. Moreover, due to the asymmetric (public - private key) cryptography, every peer has an unique identity. Whenever a peer adds data into the blockchain, everybody in the network can see his or her public address as an initiator of this transaction. Since everyone participates in this database, no duplication of data is made, hence no redundancy.

In the blockchain each block's header includes a hash of the blocks transactions, as well as a copy of the hash of the prior block's header, hence blockchain. In this way, all transactions are sequenced and cryptographically linked together. This mechanism keeps the ledger data very secure. Even if one node hosting the ledger has been tampered with, it would not be able to convince all the other nodes that it has the 'correct' data, because the ledger is distributed through a network of independent nodes.[17]

Blockchain is a linked list of blocks and a block is a group of ordered transactions. It is a distributed database on which once a data has been put, that data cannot be changed. Another unique feature is that there are specific rules, which can put data into the block. These rules, protocol, are made so that there could be no conflicts with data that is already in the database. The data is locked on to an owner. Finally, the nodes agree upon the state of the blockchain.[23] It is important that in different blockchains the consensus can be different as well. Thus, two blockchains can have different unique features.

An important note is that a blockchain network can be *permissioned* or *permissionless*.

Permissioned blockchain

This type of network means that only the ones with permission can enter the network. The consensus can be more or less a variation of Proof-of-authority, where selected nodes endorse and agree between each other of the state of the blockchain. In this case, the trade off is that the system is not as decentralized, however the transactions are much faster and cost-effective.

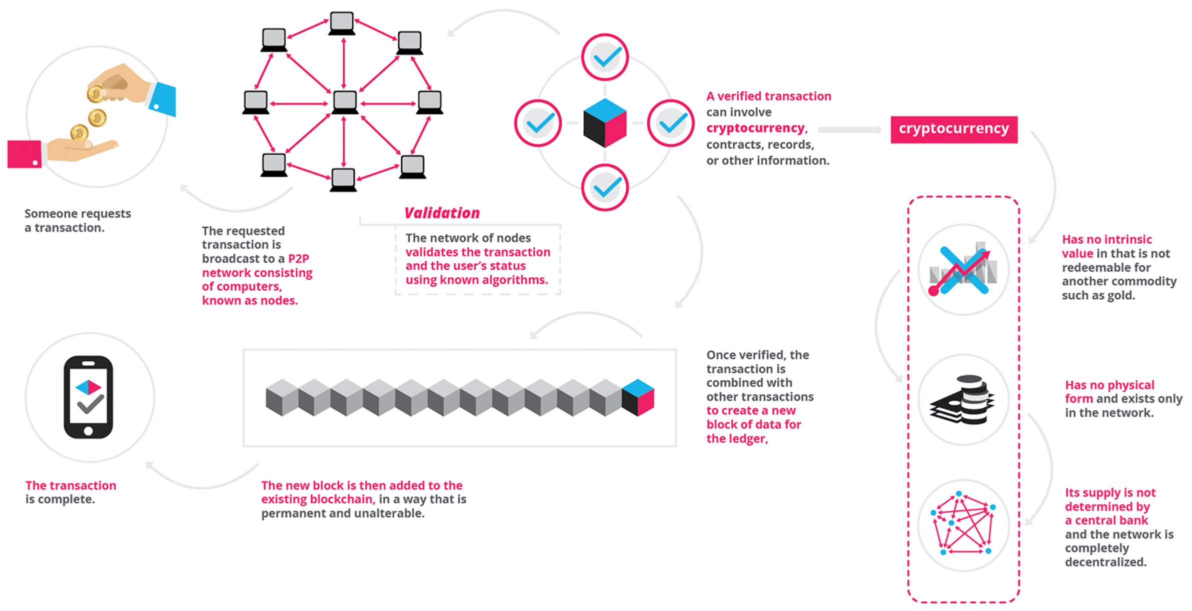
Permissionless blockchain

Everyone can join in the network. Perfect examples of such systems are Bitcoin and Ethereum. Typically the consensus they execute at the moment is called Proof-of-Work. This mechanism allows every node to participate in a fair contest to mine the next block. The winner gets either Bitcoin or Ether respective to the network. This type of consensus and availability to enter the network is giving the blockchain its most famous feature - being decentralized.

Cryptocurrency

Cryptocurrency is a digital asset, medium of exchange in the network. It is created and stored electronically in the blockchain by using encryption techniques to control the creation of monetary units and to verify the transfer of funds. The most important features that cryptocurrency possess are: it has no intrinsic value - you cannot redeem it for a raw material; it has no physical form; its supply is not determined by anyone but the creators of the respective blockchain. [6] An example of a working blockchain system with a cryptocurrency can be seen on **Figure 1**.

A peer makes a transaction. This transaction is then taken upon consideration whether it is valid or not. The decision is made by all nodes or just the ones that have been given permission to validate transactions. Upon reaching the conclusion that a transaction is valid, then it is wrapped up with many more, or in some cases alone, in order to create a block. Two things happen from the last event. First, a transaction is being completed. Second, in permissionless blockchains, the one to win the competition, to mine the newly created block receives a reward.



Someone requests a transaction.
The requested transaction is broadcast to a P2P network consisting of computers, known as nodes.
Validation
The network of nodes validates the transaction and the user's status using known algorithms.
A verified transaction can involve cryptocurrency, contracts, records, or other information.
cryptocurrency
Has no intrinsic value in that is not redeemable for another commodity such as gold.
Has no physical form and exists only in the network.
Its supply is not determined by a central bank and the network is completely decentralized.
Once verified, the transaction is combined with other transactions to create a new block of data for the ledger,
The new block is then added to the existing blockchain, in a way that is permanent and unalterable.
The transaction is complete.

Figure 1: *An abstract of a blockchain system [5]*

2.1.3 Cryptography

Cryptography is the study and practice of protecting information by encoding messages or data with the use of transformation techniques. When the message is received by the correct party, they have the necessary means to decode and consume the message. Encryption method is the algorithm to be used to convert plain message into a meaningless random characters. The method would use some sort of a key to do the encryption. In order for the receiver to make sense of this rambling they would need the correct key to decrypt the message.[12] Cryptography is widely used to protect information and communications. Three of the objectives that this field concerns itself with are:

- Confidentiality - the information is kept secret or presented in a confusing manner from anyone whom it was unintended;
- Integrity - the information cannot be tampered with without the change being detected;
- Authentication - the sender and the receiver can confirm each others identity, thus they can establish the origin of the information. This is easily done with asymmetric cryptography, but not as easy with symmetric. Because symmetric cryptography is easier to break and as such it is prone to unwanted interference

When the same key is used for encryption and decryption, it is called symmetric cryptography, while asymmetric cryptograhpy requires two different keys. One for encryption and the other for decryption. Asymmetric encryption is more secure, but it is relatively slower. [27]

2.1.4 Docker

Docker and Docker Composer were essential for the developing of this project. This technology is being used to run Hyperledger Fabric. Different parts, modules, of the system are mounted on Docker containers. All those containers know about each other and intercommunicate. This system is also known as Fabric.

Docker containers are similar to a virtual machines. Alike resource isolation and allocation benefits, however, containers are more portable and efficient since they virtualize the OS instead of hardware.[11] **Figure 2** shows an abstraction of where Docker containers take place in the software architecture when running.

2.1.5 Node Package Manager

NPM is the world's largest software registry, package manager and installer. Contains over 800,000 code packages. All packages are defined in JSON. [31]. In order to use the package manager, one has to install Node.js.

Hyperledger Composer, discussed in detail in **chapter 4**, is using composer package, part of npm. Composer cli and rest-server are two important packages that are heavily used in this thesis.

2.1.6 Visual Studio Code

Visual Studio Code for short VS Code, is an open source powerful, yet lightweight, text editor. It is available for Windows, Linux and macOS. The text editor comes with built-in support for TypeScript (Angular), JavaScript and Node.js. One of the many features is that it has a built in terminal in the GUI, making simple command calls easier than ever.

Released only 4 years ago, it is doing such a great job that in a Stack Overflow survey, the editor has been chosen as most popular in 2018.[28]

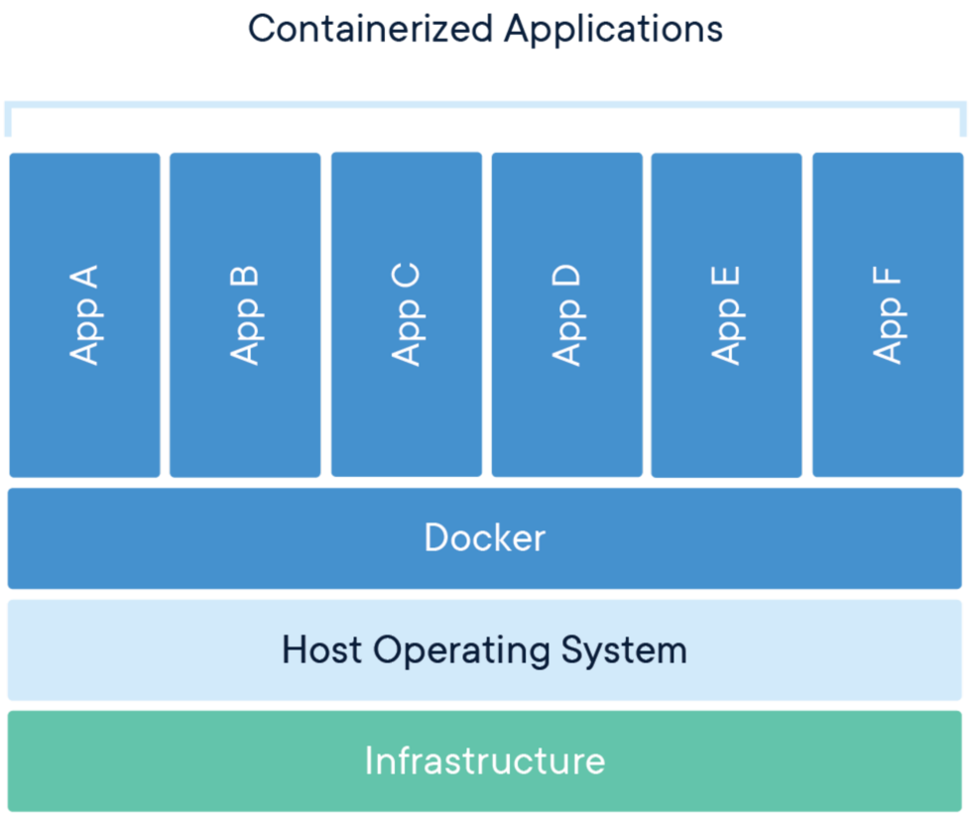
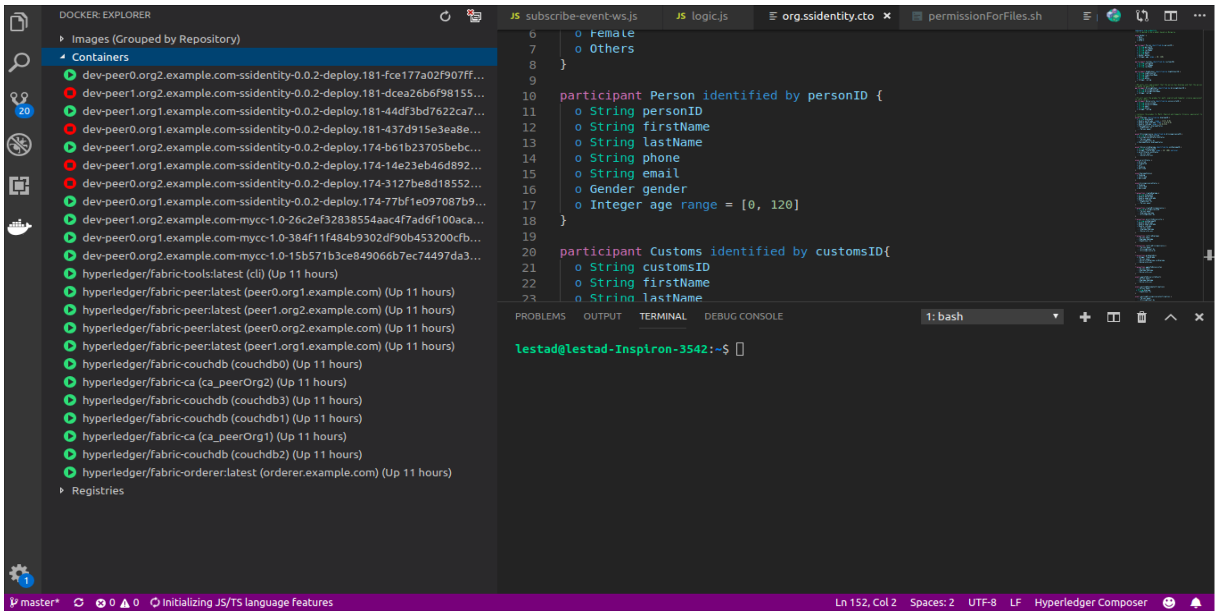


Figure 2: *Containerized applications [11]*

What's more important is that there are a lot of great extensions for this source code editor. The ones immensely important for this project are Docker, Hyperledger Composer and Beautify.

The docker extension is providing enormous help. Not only one can see the all images, and containers that are created or running on the machine, but also one can manage, control and inspect the images or containers. Usually, to monitor the state of all running Fabric nodes one need to be proficient with Linux terminal and docker commands. This plugin is easing the developer, by showing all running nodes in VS Code as shown on **Figure 3** . That way, the developer can navigate and inspect each container with a few clicks of the mouse, which saves a lot of time. The build in terminal is used when invoking or inspecting what happens in a specific container. The contrary to this method, is having to manually search and type commands through the terminal in order to find and inspect the specific container.

Hyperledger Composer extension is validating the structure of the business model files, discussed in **section 4.4** . The validating goes through the .cto, model, file. The model file contains the structure



```
DOCKER: EXPLORER
Images (Grouped by Repository)
Containers
  dev-peer0.org2.example.com-ssidentity-0.0.2-deploy.181-fce177a02f907ff...
  dev-peer1.org2.example.com-ssidentity-0.0.2-deploy.181-dcea26b6f98155...
  dev-peer1.org1.example.com-ssidentity-0.0.2-deploy.181-44df3bd7622ca7...
  dev-peer0.org1.example.com-ssidentity-0.0.2-deploy.181-437d915e3ea8e...
  dev-peer1.org2.example.com-ssidentity-0.0.2-deploy.174-b61b23705bebc...
  dev-peer1.org1.example.com-ssidentity-0.0.2-deploy.174-14e23eb46d892...
  dev-peer0.org2.example.com-ssidentity-0.0.2-deploy.174-3127be8d18552...
  dev-peer0.org1.example.com-ssidentity-0.0.2-deploy.174-77bf1e097087b9...
  dev-peer1.org2.example.com-mycc-1.0-26c2ef32838554aac4f7ad6f100aca...
  dev-peer0.org1.example.com-mycc-1.0-384f11f484b9302df90b453200cfb...
  dev-peer0.org2.example.com-mycc-1.0-15b571b3ce849066b7ec74497da3...
  hyperledger/fabric-tools:latest (cli) (Up 11 hours)
  hyperledger/fabric-peer:latest (peer0.org1.example.com) (Up 11 hours)
  hyperledger/fabric-peer:latest (peer1.org2.example.com) (Up 11 hours)
  hyperledger/fabric-peer:latest (peer0.org2.example.com) (Up 11 hours)
  hyperledger/fabric-peer:latest (peer1.org1.example.com) (Up 11 hours)
  hyperledger/fabric-couchdb (couchdb0) (Up 11 hours)
  hyperledger/fabric-ca (ca_peerOrg2) (Up 11 hours)
  hyperledger/fabric-couchdb (couchdb3) (Up 11 hours)
  hyperledger/fabric-couchdb (couchdb1) (Up 11 hours)
  hyperledger/fabric-ca (ca_peerOrg1) (Up 11 hours)
  hyperledger/fabric-couchdb (couchdb2) (Up 11 hours)
  hyperledger/fabric-orderer:latest (orderer.example.com) (Up 11 hours)
Registries
```

```
  o Female
  o Others
}

participant Person identified by personID {
  o String personID
  o String firstName
  o String lastName
  o String phone
  o String email
  o Gender gender
  o Integer age range = [0, 120]
}

participant Customs identified by customsID{
  o String customsID
  o String firstName
  o String lastName
```

```
lestad@lestad-Inspiron-3542:~$
```

Figure 3: *Docker extension in VS Code*

of the business model. So, from time to time, there could be errors, just because you are modifying one of the other files, without having the model file open. As a rule of thumb, while working on your network, its best if you have opened all files of that system at once.

Hyperledger Composer extension enables the freedom to write code in a safer environment, where one can see what's happening, in comparison to playground, online tool to run business networks discussed later in **section 4.1**. Once the code is written, testing it in the composer playground is very good idea.

2.2 Successful projects made with Hyperledger Fabric

2.2.1 Altoros

Altoros is a software company that delivers different solutions. One of the problems their customers have is issuing bonds. The customer, Russia's National Settlement Depository (NSD), wanted a system that allows automate bond placement and accounting with blockchain, while minimizing risks of reconciliation and ensuring transparency. The reason they chose Fabric is for its support of confidential transactions and resilience in the production environment. [3]

Altoros customized Fabric as needed for the different roles and actions. They set up four different channels so the communication and data transferring, between the peers and the NSD could be safe and secure. Every channel has its own chaincode (smart contract) that is basically the logistics behind the given channel.

One of the challenges they had was that the REST API was still in development. Fortunately, this is not the case anymore. Another challenge is that Fabric does not support cross-channel transactions. [4]

The benefits of choosing Fabric are:

- Faster transactions compared to the traditional solution, where a lot of data exchanging has to be done through a middleman. Thus, not only making it faster but also cheaper.
- Minimizing fraud in a secure trusted network. The permissioned feature does not allow for anyone that does not meet the requirements to monitor whats happening into the world ledger. Whats more because of the non cross-channel transactions, a peer could observe only the channels he is using. And even when he or she is inspecting another peers transaction, because of the encryption, he or she would not get any valuable information.
- Reduces expenses of the bond issuer by making the process faster and simplified [4]

2.2.2 Verified.Me

SecureKey is a company providing identity and authentication provider for simplified access to online services and applications. They use trusted providers such as banks, telcos and governments to make their clients assert identity information and connect to critical online services with digital credentials.

After the government of Canada recognized their problem sending private data to a citizen, they asked for a solution. SecureKey responded to this call in collaboration with IBM with a blockchain based solution in the form of a mobile app. The app, called Verified.Me, allows the user to connect different types of services providing only specific data. So what happens is the user connects to the blockchain through the phone. Then, it connects with the service actors. It is important to note that in the phone there are only pointers to the data and not the data itself. Whenever a person is sharing his or her identity with the new service they can see exactly what information is requested. [30]

The SIM card is used as an anchor of trust. Since the system is private and permissioned blockchain, only trusted actors like banks and government can write on it. Upon losing or breaking the SIM, the creators ensure easy recovery. Again, here one of the main reasons to choose Fabric for the development of this service is mainly - the adaptability of the platform and the zero-knowledge proof supported concept. [25]

The benefits of using Fabric are :

- Data integrity
- Security and resiliency
- No central database or honeypots
- No central point of failure
- Cannot track user across relying parties; privacy of the data
- Cost efficient due to simplifying the process

Cons:

- New - open standards needed

2.2.3 TradeLens

TradeLens is a company founded by collaborative work of Maersk and IBM. Maersk is an integrated container logistics company working on improving the supply chain area. The idea is to make the shipping process cost-efficient, faster and in respect to accessing the needed documents - simpler.

For this task, the collaborators combined their technical and specialized resource to build a system on top of Hyperledger Fabric. They created is a network, that tracks the supply chain - the documents needed for starting a shipping process, the deal that is made, the location of the containers.

To participate, a user must pay a price to enter the network. Still it is not confirmed what the requirements are. However, once a user decides to enter, he will experience something way different from the usual way of things. Due to the blockchain technology, a user can check a block on the blockchain to track the location of the container or any other process involved. The usual way for this simple task would be to request this information from a middleman. TradeLens suggests that this can reduce the costs and labour by a considerable amount. [18]

It is important to be mentioned that TradeLens is not fighting the frauds. If a user input false data at start,that seems to be correct to the endorsement parties, the system will be unable to catch it. So the network reduce fraud, but it is more of a side effect rather than main function.

Another great use of this system is that, according to the World Trade Organization, simplifying the supply chain will not only reduce costs, but also help developing countries to increase their export by more than 30% . [16]

Chapter 3

Hyperledger Fabric

This chapter is going to present the technical side of Fabric.

Permissioned blockchain

Permissioned blockchain requires peers and users to meet certain requirements to enter the network where they can perform certain actions is called permissioned. These systems are more attractive for the business and enterprise because they are faster and more cost-effective. Another feature that is appealing for those clients is the role system. That way actors, that all companies trust, can be the endorsers, the ones to validate the transactions. The feature could also be used to classify different players into respective roles, which can give a particular system a better clarification and simplicity around executing different tasks.

It is not as decentralized system like the permissionless blockchain, however the tradeoff is acceptable enough for businesses to prefer it. The processes of Anti-Money Laundering and Know Your Customer require that service providers can confirm a peers legal identity and give clearance to make a transaction. The adoption of these processes in permissionless blockchain would be wrong, since they can illuminate who this peer is, thus breaking the promised anonymity. On another note, a permissioned blockchain can have larger volume of transactions per given time compared to a public one.

What's more, many prefer permissioned blockchains for supply chain. Since only the peers inside the blockchain can see what is happening on the path of a material to its final destination. The tracking data can travel much faster, due to the simplified verification and less peers.

Hyperledger

Hyperledger is a group of open source projects focused around cross-industry distributed ledger technologies. Hosted by The Linux Foundation, collaborators include industry leaders in technology, finance, banking, supply chain management, manufacturing, and IoT.

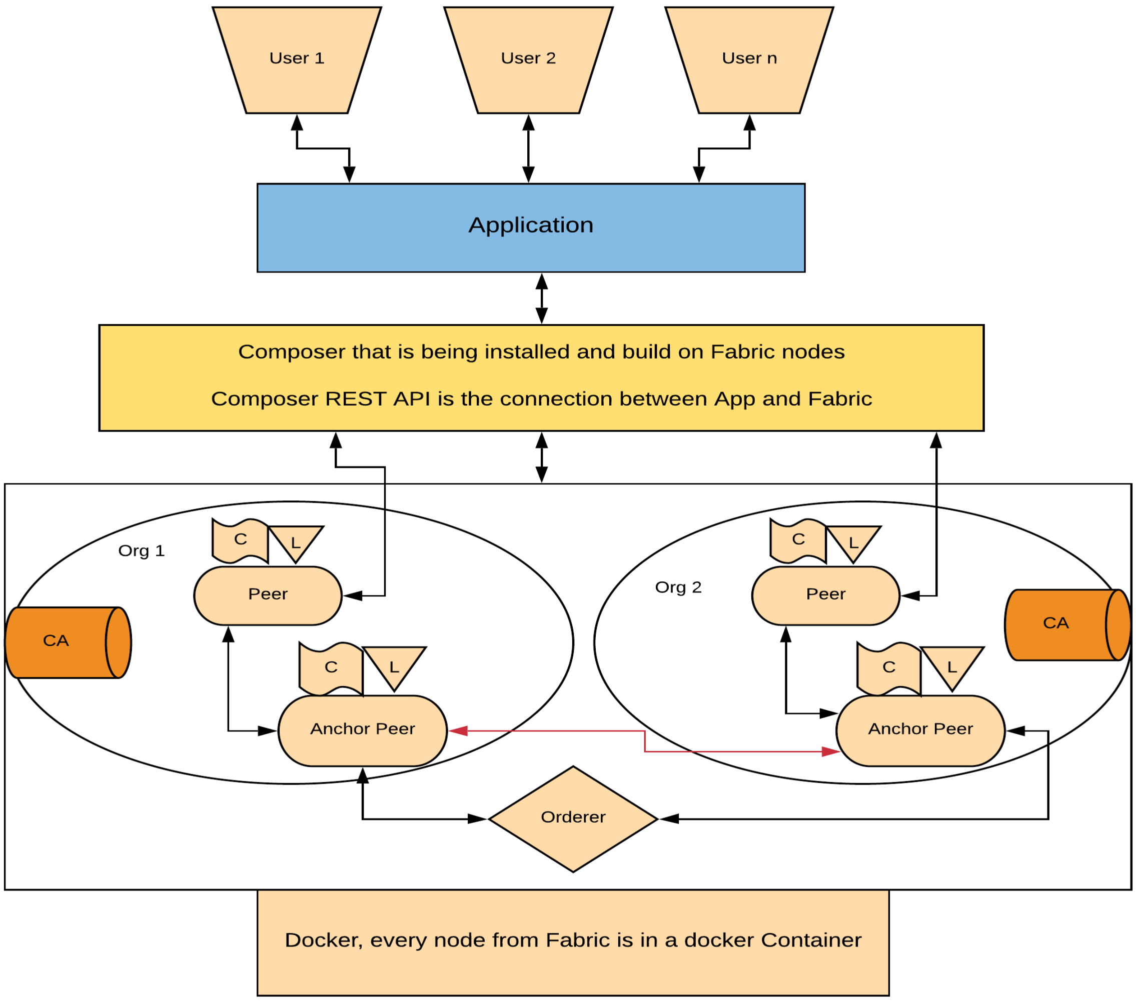

User 1
User 2
User n
Application
Composer that is being installed and build on Fabric nodes
Composer REST API is the connection between App and Fabric
Org 1
C
L
Peer
CA
C
L
Anchor Peer
Org 2
C
L
Peer
CA
C
L
Anchor Peer
Orderer
Docker, every node from Fabric is in a docker Container

Figure 4: *High-level overview of Fabric system*

3.1 Fabric overview

Fabric is one of those open source projects under the Hyperledger umbrella. It is a modular distributed ledger, which makes it highly customizable and adaptable to a variety of ideas and restrictions. A graphical overview of Fabric's structure with Composer framework is shown on **Figure 4**.

The feature which makes Fabric the perfect choice is that it can create different communication channels between different peers. Some of those channels could be for contract making between a supplier and a buyer. If a supplier has a favorite customer, he or she may give an exclusive deal. However, if everyone sees this exclusive deal, then the business of the supplier would break down. That's why this exclusive deal could exist in a confidential channel, one that only the two of them can see.

This Hyperledger project is a preferred platform mainly because of its adaptability to different use cases. One interesting feature, and main reason for the self-sovereign use case, is that Fabric supports zero-knowledge proof (ZKP). What this means is that it allows a peer to identify itself without having to show any private data. This gives authority to ZKP to offer anonymous authentication for clients in their transactions. [20]

The act of communication between different peers from different organizations (or groups) is through channels. These channels can be public or confidential. The communication inside works based on the chaincode, the smart contract. All logistics and functionality of a new blockchain application is based on its smart contracts. That is why they are extremely important and main object of interest in this undergraduate project.

3.2 Essential Fabric elements

Hyperledger Fabric is used by organizations that want to setup a consortium. Every organization is constructed by several type of peer nodes (or just peers). All that these peers require is appropriate configuration and cryptographic materials like certificate authority (CA) and endorsement policy.

Committing peer node

The normal peer node or just peer, is a vital part of the Fabrics network. This node is holding instances of the ledger, thus they commit the new blocks when received. Usually in production, multiple peer nodes are created. This imposes redundancy, but it also creates a no-single point of failure for the system.

Endorsing peer

An endorsing peer is a special kind of commiting peers, that are important in the transaction flow and the consensus. When transaction is processed for verification, the endorsing peers are taking it and simulate what can happen if the transaction is added to the ledger. For that purpose, endorsing peers also have an instance of the chaincode in order to run the transaction proposal. If the simulation went well, and there were no problems with the current state of the ledger, the endorsing peer signs the transaction as validated. The endorsement is done by the committer nodes against the endorsement policy, which is specified when the chaincode is deployed. That means that while some channels will require majority of endorsing peers to run the transaction proposal, other could be happy with just a single endorsing peer.

Orderer node

Orderer node is pivotal for the consensus mechanism and transaction flow. It is responsible for consistent Ledger state across the network. Once the endorsing peers are done with the validation of a transaction, they send it to the orderer node. The orderer node is taking all transactions and then puts them into order, then batches them into blocks. Thereafter, the block is sent to the commiting peers via the anchor nodes.

The orderer node neither executes the chaincode nor holds a copy of the blockchain. However, the ordering service (multiple nodes) are implementing specific ordering algorithms to decide what to do with the responses given from the endorsing peers. More about them in the consensus section.

Anchor peer

The anchor peer is the connection of the organization with the network. If there is no anchor peer in the organization, then this organization cannot connect to any other organization. More importantly, the anchored peer is the link between the orderer node and the ledger inside the organizations. If there is no mechanism to send or receive transactions then the whole organization will become obsolete. Thus, having setup several anchor peers, just in case of some unfaithful crash, is the safe bet.

CA

Uses the BCCSP, Blockchain Cryptographic Service Provider, to generate cryptographic material for Peers, Orderers and Users, that certifies them and allowing access to the respective network. CA provide dynamic certificate lifecycle management capabilities such as registering, revoking and erolling users via the REST (representational state transfer) API.[21]

Membership Service Provider

MSP is responsible for all the cryptographic operations - issuing, verification, signing. Every organization has local specification of MSP with local specification of CA.

Cryptography in Fabric

Fabric uses BCCSP, the Blockchain Cryptographic Service Provider. It is a creation of IBM and it offers implementation of cryptographic standards and algorithms. Written in golang, mainly to be used on Hyperledger Fabric.

BCCSP is designed to be pluggable component into the fabric network. It can mount different public key cryptographic standards, depending on the client's requirements. Supports both symmetric and asymmetric encryption. Fabric needs both simple encryption/decryption and signed. Once the key for the preferred algorithm is retrieved BCCSP exposes encryption and decryption methods to be called. The sign is obtained by the key as well, thereafter sign and verify methods are exposed to be used. [13]

Certificates and PKI

Every organization in Fabric has a membership service provider (MSP) which is pluggable interface to support variety of credentials architectures. In other words, it takes what the BCCSP provides. The MSP is responsible for the identity and the authenticity of respective organizations' peers and users. By default, the security implementation is Fabric-CA. Public key infrastructure that writes certificates defined by x.509 standard.

3.3 Fabric's Ledger

Consists of two distinct, though related, parts - **world state** and a **blockchain**.

3.3.1 World state

The world state is a database that holds the current values of the assets in the ledger as ledger states. The ledger states are usually expressed as key-value pairs, though there is some flexibility in this regard. The world state changes frequently because of the CRUD operations applied on the network. Only validated transactions can change the ledger.

The world state is created with the premise of faster transactions. Instead of traversing the entire blockchain to calculate the current value of the asset, a program can just take it from the world state.

The world state is a NoSQL database. It provides rich set of operations for the efficient storage and retrieval of states. Fabric can be configured to use different types of db that will answer to the requirements of the network. Usually Fabric will be either with LevelDB for simple networks and CouchDB for more complex networks.

In order to keep track of the changes in the WS, a counter called version number is incremented every time there is a change. This counter is checked whenever the state is updated to make sure that the current state of an asset matches the version at the time of validating the transaction. This ensures that the world state is changing as expected, meaning there has not been a concurrent update. [17]

3.3.2 Blockchain

The blockchain is with its defining qualities - immutable sequence of blocks, each of which contains a set of ordered transactions. Every new transaction is being validated or rejected. The successful transactions are batched into blocks and appended to the blockchain - enabling you to understand the history of changes, which result into the creation of the WS. On **Figure 5** is a representation of the ledger

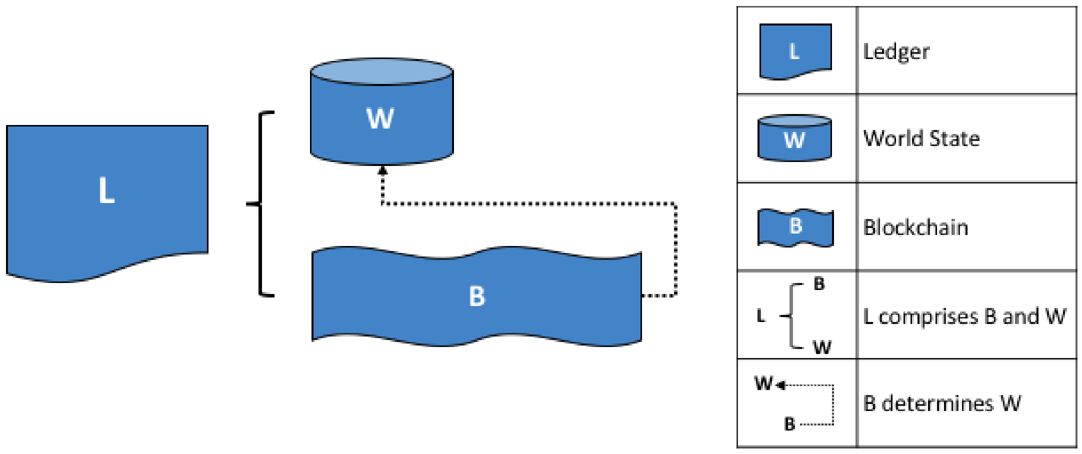


Figure 5: *An abstract of the Fabric Ledger [17]*

Physically, the blockchain is always implemented as a file, in contrast to the world state. This is a reasonable design choice as the set of operations on the blockchain data structure is heavily biased towards small limited number. Appending to the end of the blockchain is the primary operation, querying is currently infrequent operation because of the WS.[17]

Blocks and their structure

Figure 6 shows how the blockchain is structured and their graphical representation.

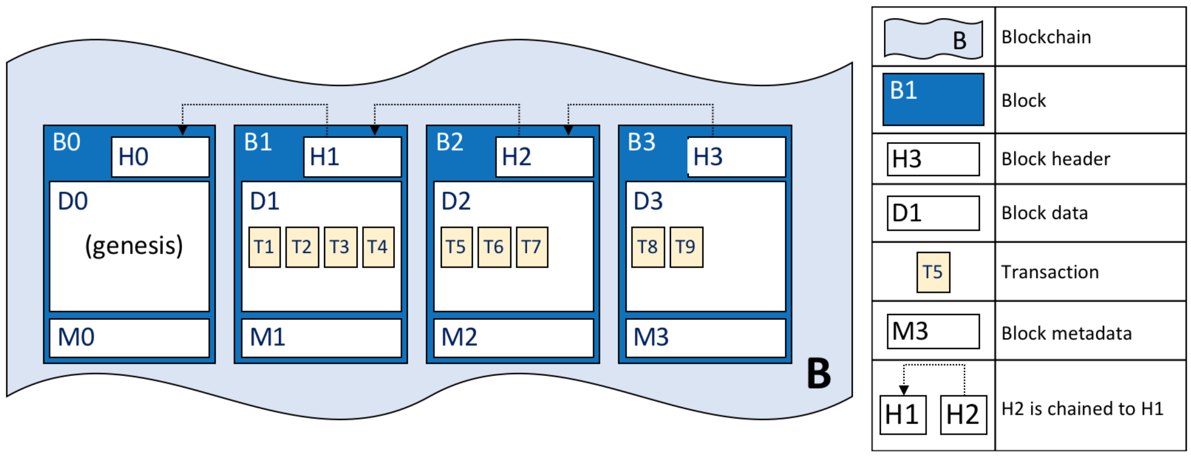


Figure 6: *A graphical representation of the block [17]*

- Block header - consists of three fields:
  - Block number - integer, at 0 is the genesis block, increased by one for every new block that is appended to the blockchain
  - Current block hash - hash of all the transactions contained in the current block
  - Previous block hash - copy of the hash from the previous block
- Block data - contains a list of transactions arranged in order of appending.
- Block metadata - contains the time when the block was written as well as, the public key, certificate and signature of the block writer.
- Transactions
  - Transaction header - captures the essential data about the transaction like name of the chaincode and its version.
  - Signature - is being generated only by the user's private key. The field is used to check that the transaction details have not been changed.
  - Proposal - encodes input parameters supplied by the user via an application to provide them to the chaincode which creates the proposed ledger update. But before the proposed

transaction is being added to the blockchain it first has to be verified and validated by the *endorsing peers* which are discussed in the following subsection.

- Response - captures the before and after values of the world state, as a read-write set. It's the output of a chaincode (smart contract), if the transaction is successfully validated, the response will be applied to the ledger to update the world state.
- Endorsements - Although there is only one transaction response, there are multiple endorsements from different *organizations*.If there are not enough endorsements, specified in the transaction verification process, the response would not match the needed number and the transaction will be rejected as invalid and will not update the world state.

3.3.3 Channel

In Ethereum or Bitcoin, when someone joins the network, they are connecting to the blockchain. That being just one ledger starting from genesis block. Every participant has all blocks or just the headers, however, the important note is that everybody keeps information from the same blockchain.

Hyperledger Fabric, contrary to the blockchains mentioned, can have several different ledgers in one network. Since it is built with the premise of business, having several ledgers in a network is beneficial. It is possible, because the ledger is owned by a channel. A channel is like a communication channel, it is the mechanism by which different organizations are interacting between each other. Those interactions are appending on the immutable ledger.

What's more, every channel is having its own business logic, chaincode installed. So, a network, depending on the use case, can have one channel with all business partners, one channel with all end users, and several other channels with specific business partners and end users. And all of them can have different chaincodes or the same, depending on the case and requirements of the system.

All peers from a channel have the same ledger. What's more a peer can have multiple ledgers, hence multiple instances of different chaincode. However, those ledgers are completely different and cannot interact between each other.

3.4 Fabric Consensus

The consensus in fabric is broken out into three phases: Endorsement, Ordering and Validation. At the end of the section, **Figure 7** is representing the whole process.

3.4.1 Endorsement

Endorsement is driven by policy (m out of n signatures) upon which participants endorse a transaction.

This phase starts with client application sending transaction proposal. A transaction proposal is an action that will change the state of one or more assets in the ledger. The endorsing peers are taking this proposal and simulate it into the network. The transaction has been signed with the result of the simulation. Depending on the policy, which is defined upon installation of the chaincode, there will be a number of signatures needed before the transaction to proceed into the block. If the policy is not fulfilled, then the transaction is going to be rejected.

At this point, nothing in the ledger is changed. Many different transaction proposals can be taken and put against the current ledger to check out whether the transaction is going to be valid or rejected. While it is done with the premise of scalability, this parallel validation is introducing one problem.

If there are two transactions being validated that update the state of the same asset, then they will be voted both valid from the endorsing peers. However, once the transactions are being sent to the next phase, the ordering, one of them will fail. Upon batching the transactions, they are put into order. Depending on that order, what happens to be second transaction is going to fail.

3.4.2 Ordering

Ordering phase will get the endorsed transaction and agrees to the order to be committed to the ledger.

The ordering phase or service consists of cluster of orderer nodes. They are receiving the transactions. Batch the transactions into a block. An important event occurs here, that once a transaction is put into the block, it it said to be final. This means that the position of the transaction in the ledger is immutable. The order is consistent and strict. Finally, the blocks are distributed to the committing peers.

Note, the order of the transactions is not necessarily as in the order of the transactions send to the node. This is important, because once the batch is done, some of the transactions may be rejected. It can happen due to the fact that two transactions may have tried to change the same resource, and only one change can happen per block. When transaction proposals enter the network, the endorsers are simulating them in a parallel manner, against the current ledger. So, both transactions will be looking at the same state of the asset they want to change. Both can be valid, however, the first that gets to be put in the order batch is the one that is going to make the change in the blockchain and world state. The second is going to be rejected.

Batches are defined mainly by two factors. The first is the time to wait before a block is being generated. The countdown starts after the first transaction is received and can finish on the time specified or before that. For a block to be generated before the end of the time set, the blocks size had to reach its limit. If the configuration is done so that 10 transactions can be put into a block, and 2 seconds to be time set, then there are two cases. First - have a block in 1 second with 10 transactions. Second - have a block with less than 10 transactions after 2 seconds. The configuration can be found in confingtx.yaml. Will be discussed in more detail in the configuration section. (12 min [19])

So, in order to effectively avoid invalid transactions due to two transactions updating the same resource, the configuration of generating the blocks have to be well-thought for the specific system.

There are several types of ordering mechanisms implemented in Fabric. They are all pluggable, so the engineers of the network could test them independently, simply to see which one will be most efficient for the case:

- SOLO - involves a single ordering node, single point of failure. It is very fast, but unreliable for real data, which makes it the perfect mechanism in developing stage.
- Kafka - based on Apache Kafka, high-throughput, low-latency platform. Crash fault-tolerant solution.

- PBFT - practical byzantine fault tolerant mechanism. It is both crash fault and byzantine fault tolerant, meaning it can reach an agreement even in the presence of malicious or faulty nodes. Slow, but secure.

According to an IBM paper [7], PBFT is the most used one in production, however, due to the pluggable design, depending on the network, it can be changed with Kafka or a future solution.

Usually, since the first and the third steps are always the same, people would refer to Fabric consensus just as the name of the ordering mechanism.

3.4.3 Validation

Validation takes a block of ordered transactions and validates the correctness of the result. When a committer node receives the new block, it starts checking every transaction and the transaction result from the endorsing peers. This check is against the endorsement policy. Here is the moment where if two transactions are trying to change the one asset, the second fails. However, instead of returning the whole block, the faulty transaction is just labeled as *invalid*. The committer updates the ledger. Lastly, asynchronously returns to the user/app that the transaction is successful or not.

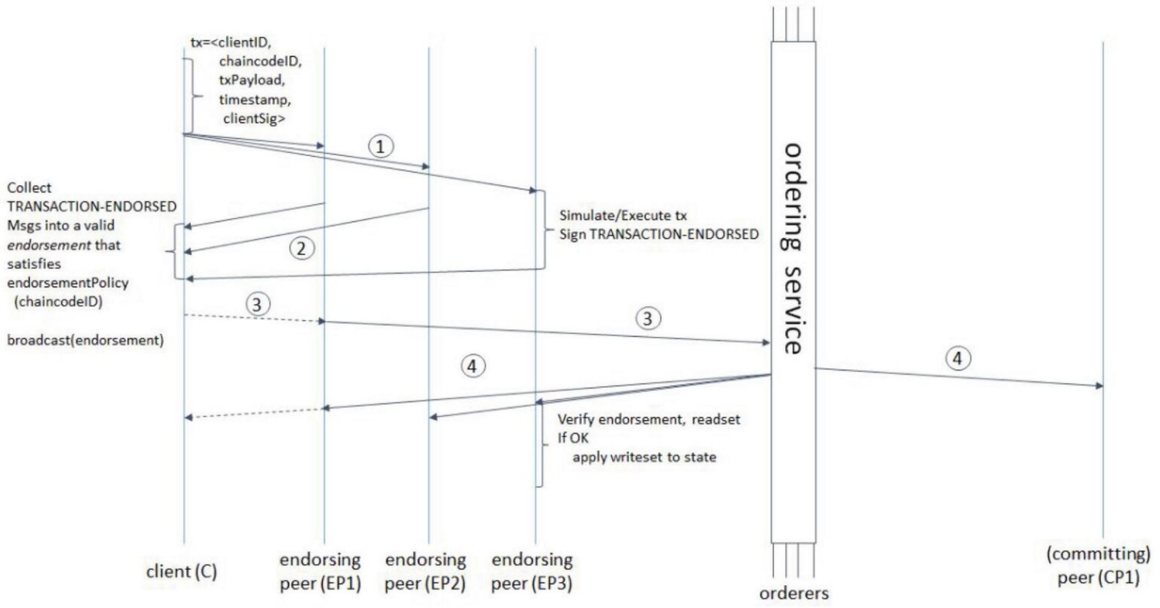
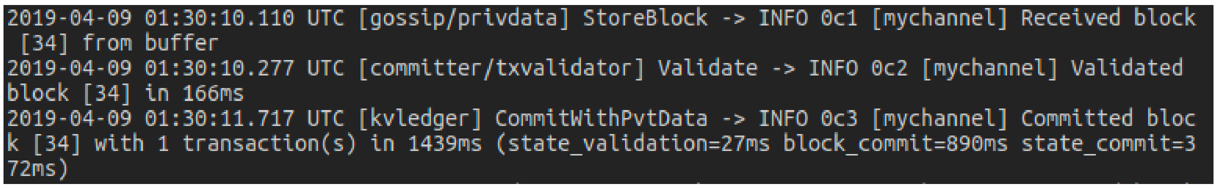


Figure 7: *A graphical representation of the transaction flow [15]*

By inspecting a peer node of running network, one can see a report from the ongoing process like the one shown on Figure 8.

3.5 Fabric's configuration

When setting up Fabric, there are two very important files. It is out of the scope of this project to show how one can generate custom network. However, it is critical that the reader is introduced to



```
2019-04-09 01:30:10.110 UTC [gossip/privdata] StoreBlock -> INFO 0c1 [mychannel] Received block
 [34] from buffer
2019-04-09 01:30:10.277 UTC [committer/txvalidator] Validate -> INFO 0c2 [mychannel] Validated
block [34] in 166ms
2019-04-09 01:30:11.717 UTC [kvledger] CommitWithPvtData -> INFO 0c3 [mychannel] Committed bloc
k [34] with 1 transaction(s) in 1439ms (state_validation=27ms block_commit=890ms state_commit=3
72ms)
```

Figure 8: *Report from committed block to the ledger*

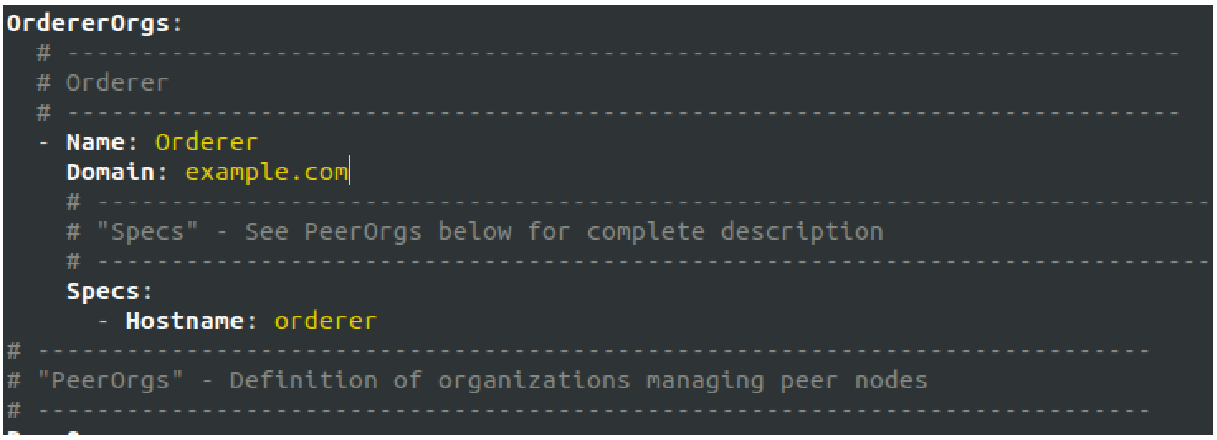
those two configuration files and see the important bits and pieces.

The two files are crypto-config.yaml and configtx.yaml (If you follow the user manual and download the git repository, then the files should be located at /fabric-samples/first-network/).

Yaml is a human-friendly data serialization standard for all programming languages. In Hyperledger Fabric it is being used for all configuration files.

3.5.1 Crypto-config

The crypto-config.yaml holds the information about the configuration of the orderer nodes and the organizations. The location of the machines is not important, as long as they are correctly assigned to their corresponding organizations in the network and have the respective certificate to operate with those organizations.



```
OrdererOrgs:
  # ---------------------------------------------------------------------------
  # Orderer
  # ---------------------------------------------------------------------------
  - Name: Orderer
    Domain: example.com
    # ---------------------------------------------------------------------------
    # "Specs" - See PeerOrgs below for complete description
    # ---------------------------------------------------------------------------
    Specs:
      - Hostname: orderer
# ---------------------------------------------------------------------------
# "PeerOrgs" - Definition of organizations managing peer nodes
# ---------------------------------------------------------------------------
```

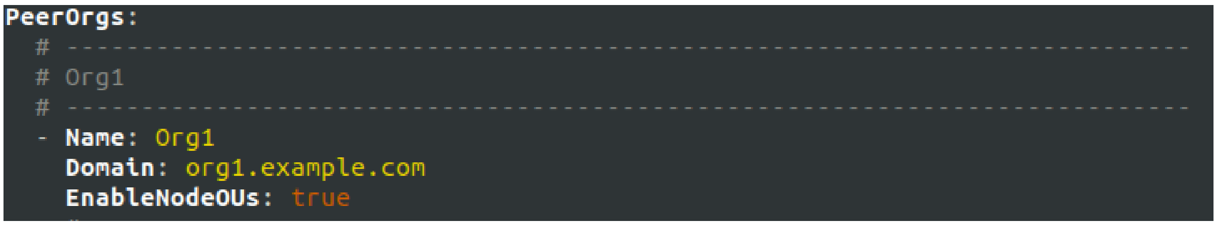
Figure 9: *Orderer configuration*

The first configuration is about the orderer nodes shown in **Figure 9**. The name specifies the name of cluster of orderer nodes. Different clusters can be of use for different channels. Depends on the use case. In this file, there is one cluster.

The domain is where this entity will run. This is what other nodes will try to find in order to connect to the orderer. It is not necessary to be a top-level domain, like a web page. Depending on the use case, the network could have a local private domain space that could be used in the context of the network.

Certificates are bind to specific domains. So, if an anchor node does not have the correct certificate, it would not be able to connect to this orderer node. Security measure that prevents outsiders to invoke information from this orderer.

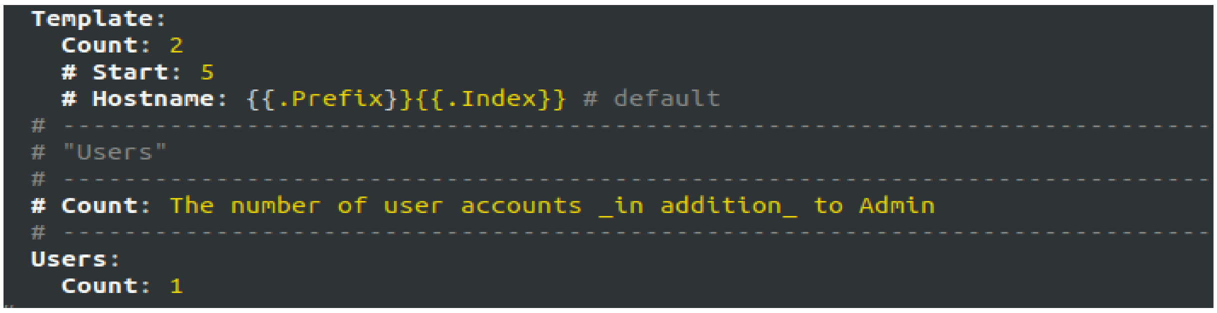
The hostname specifies the name and creates a node. In **Figure 9**, there is only one node, called orderer. Left with one node for the orderer, is dangerous, as it is a single point of failure. For testing purposes, it is okay, however in real-life system, there should be a lot more. The name by which other nodes are going to find this one is by the schema, first the hostname then the domain - orderer.example.com



```
PeerOrgs:
  # ---------------------------------------------------------------------------
  # Org1
  # ---------------------------------------------------------------------------
  - Name: Org1
    Domain: org1.example.com
    EnableNodeOUs: true
```

Figure 10: *Peer organization configuration*

The next part is about the peer organization shown on **Figure 10**. The name and domain serve the same function.



```
  Template:
    Count: 2
    # Start: 5
    # Hostname: {{.Prefix}}{{.Index}} # default
  # -------------------------------------------------------------------------
  # "Users"
  # -------------------------------------------------------------------------
  # Count: The number of user accounts _in addition_ to Admin
  # -------------------------------------------------------------------------
  Users:
    Count: 1
```

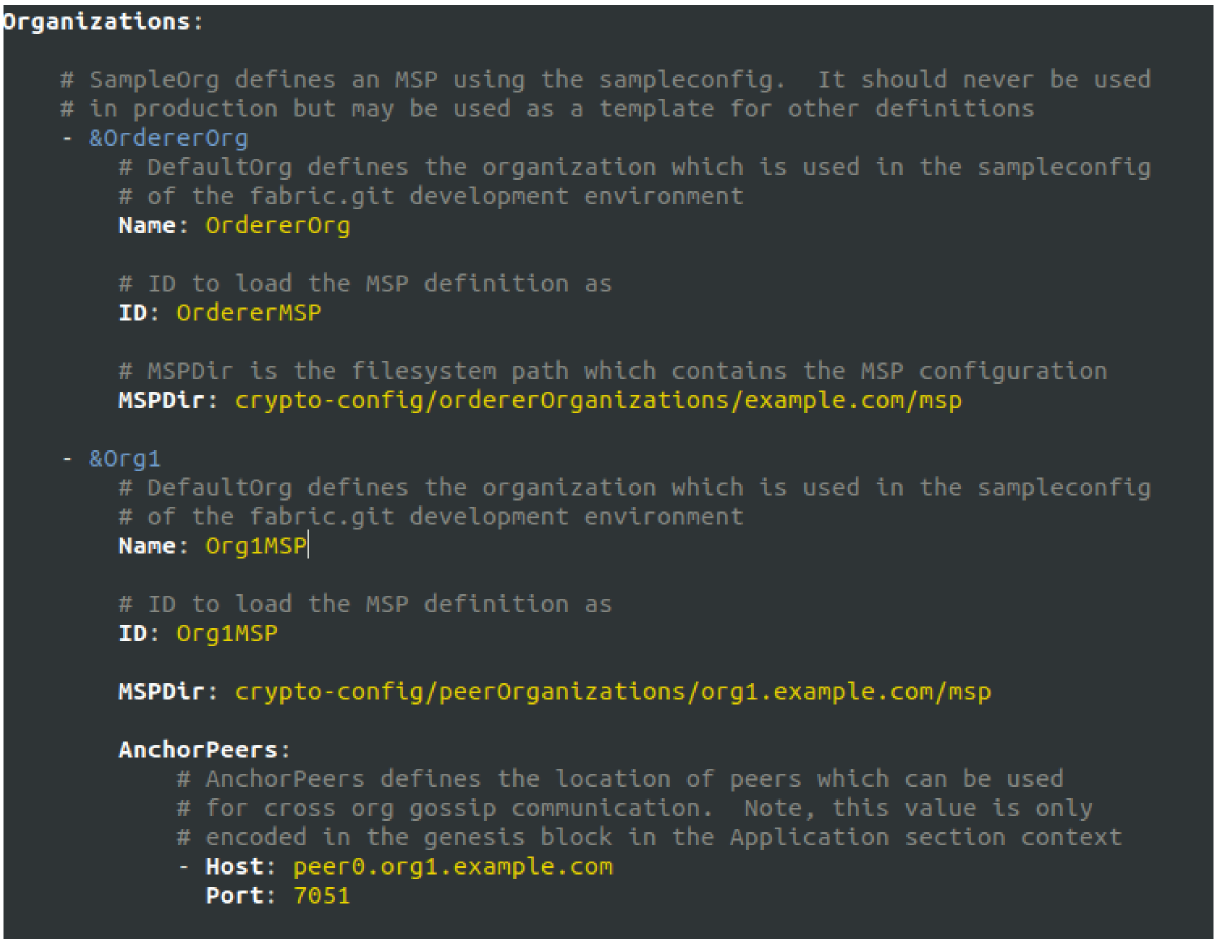
Figure 11: *Template for organizations configuration*

Figure 11 shows the configuration about the template. Template is about the different nodes in the particular organization. The count variable sets with how many nodes will the organization start. In this case there will be two peers from this organization. Later on, peer nodes can be introduced or deleted, depending on the requirements and what would be the most beneficial occasion. Important note is that every peer is having its own certificate. The name of the node will be peer0.org1.example.com.

The user's section is about how many users, other than the admin, should the network start with. If the system is about a lot of people, then this variable would not matter. The Certificate Authority can dynamically introduce new users and nodes to the system.

3.5.2 Configtx

Configtx.yaml holds information about the organizations and the orderer in a bit more detail in regard to input and output. It is the configuration about the genesis block on a channel.



```
Organizations:

    # SampleOrg defines an MSP using the sampleconfig.  It should never be used
    # in production but may be used as a template for other definitions
    - &OrdererOrg
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: OrdererOrg

        # ID to load the MSP definition as
        ID: OrdererMSP

        # MSPDir is the filesystem path which contains the MSP configuration
        MSPDir: crypto-config/ordererOrganizations/example.com/msp

    - &Org1
        # DefaultOrg defines the organization which is used in the sampleconfig
        # of the fabric.git development environment
        Name: Org1MSP

        # ID to load the MSP definition as
        ID: Org1MSP

        MSPDir: crypto-config/peerOrganizations/org1.example.com/msp

        AnchorPeers:
            # AnchorPeers defines the location of peers which can be used
            # for cross org gossip communication.  Note, this value is only
            # encoded in the genesis block in the Application section context
            - Host: peer0.org1.example.com
              Port: 7051
```

Figure 12: *Organizations MSP and anchor peer configuration*

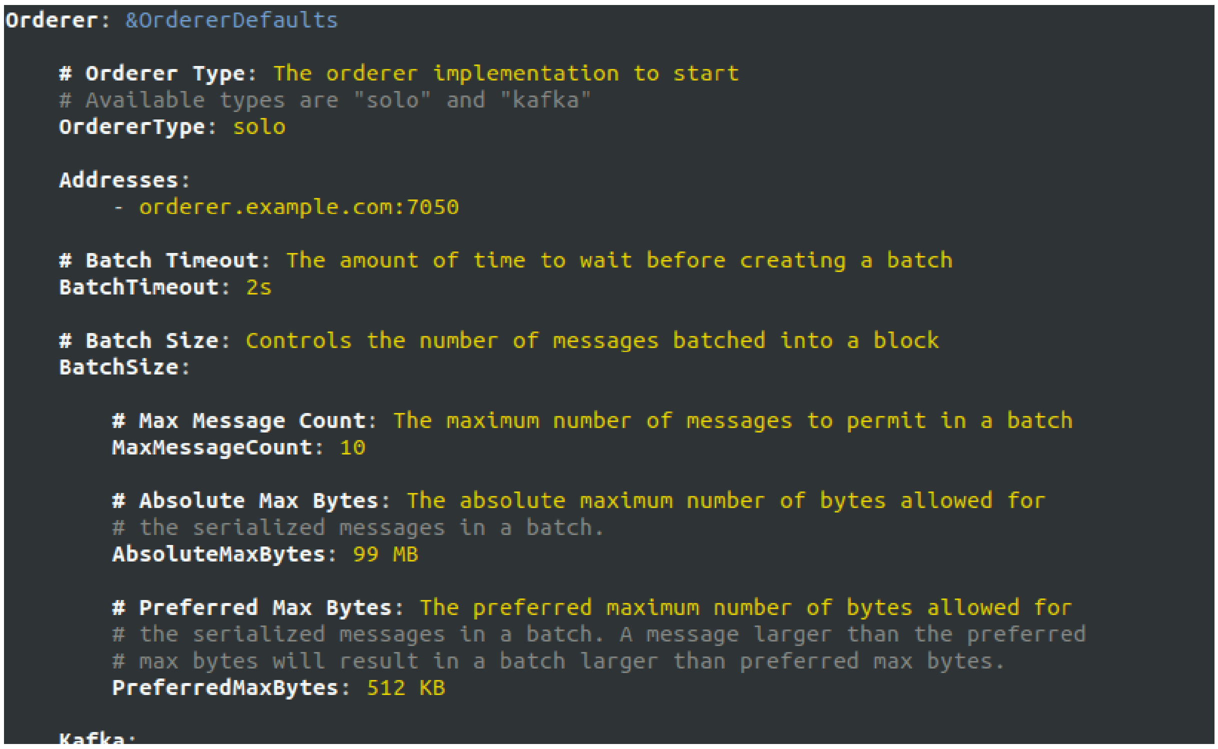
In **Figure 12** you will see Name, ID and MSP. The three variables are regarding the cryptography and accessibility. The ID is defined by the membership service provider. It is associated with all the crypto-materials - certificates by admin, ca and tls. The MSP ID is verification tool in the backend to verify the user's certificates.

Specific section about the Anchor Peer - selecting which node is going to be the anchor peer in an organization. These nodes will be able to see each other by host name and communicate via port. This is how a connection between two or more organizations is configured.

Specific setting for the orderer. The orderer service is specified under the variable OrdererType, which is SOLO, not good for production, but good for testing. Under the addresses are specified all nodes from the cluster, which are going to batch the transactions into block. In this default configuration that is only one node.

Next in line are the configurations about the block size and finalization shown on **Figure 13**. BatchTimeout is the variable holding how many seconds should the ordering service wait before it batches all available transactions into a block. The only case where transactions are going to be batched earlier is if the max number of transactions for a block has been reached.

BatchSize has three variables MaxMessageCount, which is defines the top limit of transactions



```
Orderer: &OrdererDefaults

    # Orderer Type: The orderer implementation to start
    # Available types are "solo" and "kafka"
    OrdererType: solo

    Addresses:
        - orderer.example.com:7050

    # Batch Timeout: The amount of time to wait before creating a batch
    BatchTimeout: 2s

    # Batch Size: Controls the number of messages batched into a block
    BatchSize:

        # Max Message Count: The maximum number of messages to permit in a batch
        MaxMessageCount: 10

        # Absolute Max Bytes: The absolute maximum number of bytes allowed for
        # the serialized messages in a batch.
        AbsoluteMaxBytes: 99 MB

        # Preferred Max Bytes: The preferred maximum number of bytes allowed for
        # the serialized messages in a batch. A message larger than the preferred
        # max bytes will result in a batch larger than preferred max bytes.
        PreferredMaxBytes: 512 KB

    Kafka:
```

Figure 13: *Block configuration*

can a block have. AbsoluteMaxBytes and PrefferedMaxBytes are defining the size of the block.

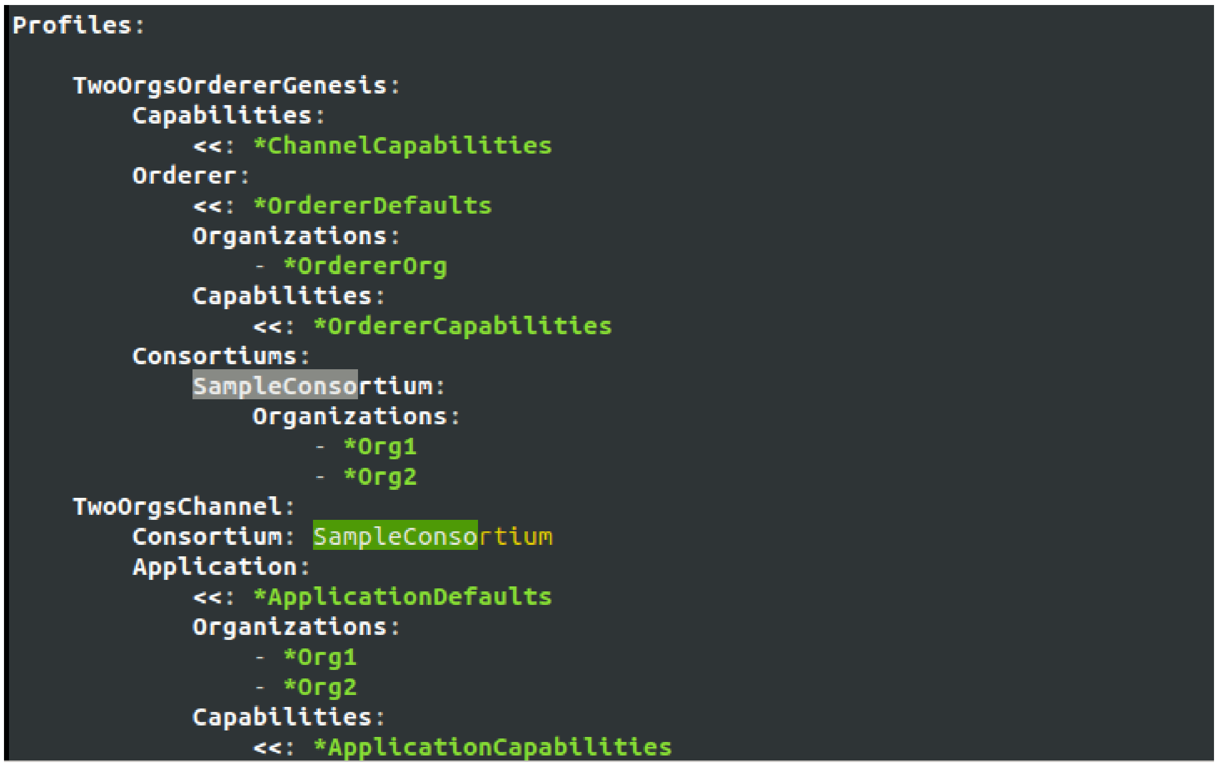
Since endorsing peers are simulating the transaction proposals on the current state of the ledger, the finalizing of transactions is of utmost importance. As previously mentioned, valid transactions may be rejected because the ledger is not updated frequently enough. In the perfect case, the system would be able to take enough transactions for small amount of time, so that the blockchain would always be updated with the latest actions and state changes.

BatchTimeout and BatchSize are extremely important. If not set correctly they can break the system. On the contrary, if set with care the configuration could make the system faster and resistant to invalid transactions. It is case specific what would be the best strategy for their configuration. The architect should always have in mind the blocks to be generated fast enough and to be the correct size. If the blocks are not going to the transactions number limit, then they would be just wasting the resources allocated.

In the profiles section shown on **Figure 14** are the final configurations for the genesis block of the channel and the participants of this network.

Configurations about the genesis block are set before the channel. The settings are defining the ordering system, the participants in the blockchain. Setting which organization are going to enter the consortium, later to be given in the channel configuration. Consortium in this context is which organization will be in a channel that is being served by the orderer.

Lastly is the channel definition, where the architect would specify which consortium(s) is going to be used, and which organizations are going to be initially in the particular network.



```
Profiles:

    TwoOrgsOrdererGenesis:
        Capabilities:
            <<: *ChannelCapabilities
        Orderer:
            <<: *OrdererDefaults
            Organizations:
                - *OrdererOrg
            Capabilities:
                <<: *OrdererCapabilities
        Consortiums:
            SampleConsortium:
                Organizations:
                    - *Org1
                    - *Org2
    TwoOrgsChannel:
        Consortium: SampleConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
                - *Org2
            Capabilities:
                <<: *ApplicationCapabilities
```

Figure 14: *Finalizing channel participants and consortium*

Chapter 4

Hyperledger Composer

4.1 Overview

The tool aims to simplify the creation of Fabric blockchain applications. Composer lets the user to establish a system without having the knowledge of Go programming language or low-level details about blockchain.

Composer creates blockchain business networks which are compressed into .bna files. They are composed out of three main files - model, logic, permissions - that are discussed in **section 4.4**.

One can test out the .bna network without Fabric nodes, just to see if it is going to work at all. The online tool https://composer-playground.mybluemix.net can provide an environment for testing, fixing and fast deployment of the fixed system. Another great functionality of this online tool is that it can export the system as .bna file. Instead of going through mundane process of manually creating the connection and compression of all files into business network, one can compress them through the playground.

There are two things to keep an eye on when exporting the business network. First is that all files of the system should be present in the composer-playground. Second, when installing the .bna file onto Fabric, the command needs a version of the system. It can be found on the define window on the left bar From: previous version To: current version . The composer cli command requires the current version to be provided.

Unfortunately, IBM will not continue further development on the project. Composer and Fabric drifted away from each other. The features in the latest updates of Fabric are not compatible with the high-level code of Composer.[26]

4.1.1 Composer CLI

Composer command line interface, with primary command in the terminal composer, is used for administrative, development and operational tasks.

This project has been tested and developed with composer cli 20.2 and composer-rest-server 20.7, since crs 20.2 breaks upon using one of the functionalities (multi-user crs).

4.2 Business Network Cards

In order for user to be using the system, they have to be recognised by the system. This identification can be created from the system they are going to be using or import from another one. The importing from another can happen only if the requirements for participant match.

When someone wants to enter a blockchain network, a public and private key-pair is generated. The user then can interact with the system with those two keys. Usually whenever someone wants to send a message to the user, he or she will take the users public key and encrypt the message with it and send it to the user. Then the user will decrypt it with his/her private key. On the other hand, if the user wants to create something and put it to everyone, he will encrypt that creation with his/her private key and will put it open in the network. If someone wants to see what this new message is, they have to take that users public key in order to decrypt the creation and see it. Among other things such as security, this method also proves authentication and integrity of the message.

In the world of Hyperledger Composer, this credentials pair is called business network card. It holds the keys, certificates and connection profile to a particular system. One user can have multiple business cards for different networks. In some for being an end user, for others - business partner.

BNC (blokchain network card) provides all of the information that is needed to connect to a BBN (blockchain business network). You can only access a BBN through a valid BNC. A card contains an Identity for a single Participant within a deployed BN.

Can be stored on: File system, RAM (embedded runtime), database and cloud storage. The safest is local file system.

4.3 Composer Historian

Specialized registry which records successful transactions, including the participants and identities submitted them. The historian stores transactions as HistorianRecord assets, which are defined in the Hyperledger Composer system namespace.

The historian registry is a Hyperledger Composer system-level entity. To refer to the historian registry as a resource for access control the historian must be referenced as: org.hyperledger.composer.system.HistorianRec

Note that: All participants must have the permission to create HistorianRecord assets. If a transaction is submitted by a participant who does not have the permission to create such, the transaction will fail.

4.4 Composer programming

Hyperledger Composer is using several files in order to create a compressed one that would be installed on Fabric.

4.4.1 Model file

Model files contain the structure and types of the system. The modeling language is object-oriented. It supports inheritance, concepts, enumerations and different types of variables. Because of the inheritance, the resource child will take all properties and fields required from the extended resource parent and will add additional from its own definition.

The most important part of a model file is the namespace - defined at the start of the document. In all references to the resources of this particular file, the namespace will be present as part of it. If the file does not contain a namespace, it will be unusable

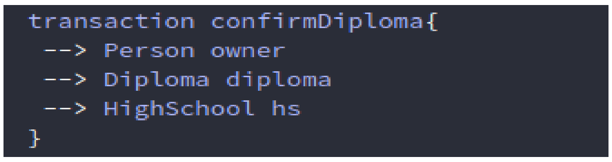
In Composer the system can be modelled by introducing participants, assets, transactions and events.

- Participants - usually can be thought of as a person or some institution.
- Assets - can use as model everything of value
- Transactions - have built in variables for transactionID and timestamp
- Events - have built in variables for eventID and timestamp

The primitive types supported in the language are:

- UTF8 encoded String
- 64-bit numeric value Double
- 32-bit signed whole number Integer
- 64 signed whole number Long
- an ISO-8601 compatible time instance, with optional time zone and UTZ offset DateTime
- Boolean.

Within the model one can also have arrays and relationships. Reference to an asset or participant is indicated by –>and are called relationships.



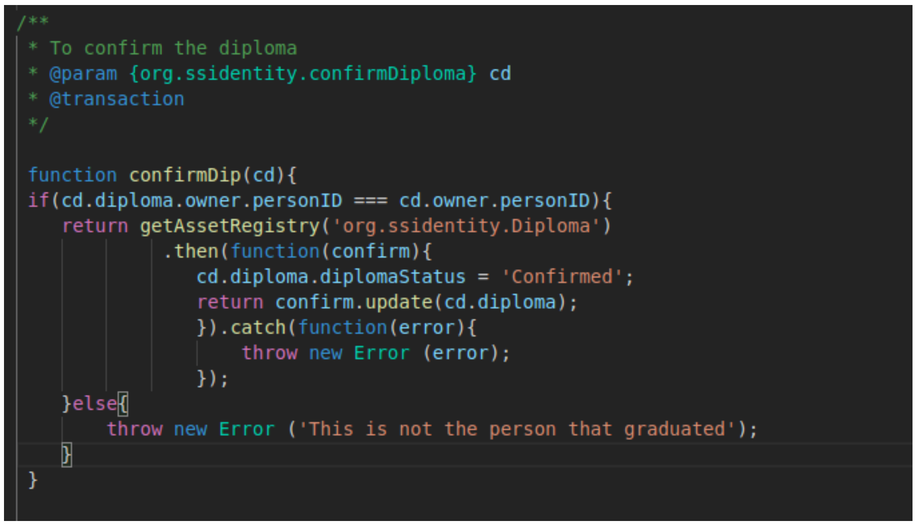
```
transaction confirmDiploma{
 --> Person owner
 --> Diploma diploma
 --> HighSchool hs
}
```

Figure 15: *Example transaction - confirmDiploma*

As seen on **Figure 15** is an ordered set of the objects name and the instance created. The full name of the newly created resource will be namespace.Resource#nameOfTheObject for example org.ssidentity.Person#examplePersonID.

4.4.2 Logic file

The Logic file is where the business logic is written. The programming language used is Type-Script Angular framework. Unusual characteristic is that you have to define the resource and parameter in the comment section before using it in a normal function.



```
/**
 * To confirm the diploma
 * @param {org.ssidentity.confirmDiploma} cd
 * @transaction
 */

function confirmDip(cd){
if(cd.diploma.owner.personID === cd.owner.personID){
    return getAssetRegistry('org.ssidentity.Diploma')
            .then(function(confirm){
                cd.diploma.diplomaStatus = 'Confirmed';
                return confirm.update(cd.diploma);
                }).catch(function(error){
                    throw new Error (error);
                });
    }else{
        throw new Error ('This is not the person that graduated');
    }
}
```

Figure 16: *confirmDiploma transaction coded in Angular*

4.4.3 Access control language file

ACL provides declarative access control over the elements of the domain model. The rules are consecutively executed. Which means that if the first rule is to grant permission to resources to any action, no matter what rules follow, all resources will be grant full access. ACL is constructed by a single file with fixed name - permissions.acl.

The file has rules to restrict the users of action, rather than to permit such. If the file does not exist, then there is no restriction in the system for any resource or user. The rules have an action: allow or deny

Rule controls permission to CRUD operation on resource(s). Unfortunately, each rule can use only one resource. Meaning that in order to give permission to one participant to function with several resources, a rule that allows CRUD operation for each specific resource have to be written.

Simple rule control access to namespace, asset or property of an asset by a participant type or participant instance.

Conditional rule takes boolean JavaScript expression evaluated at runtime to ALLOW or DENY access to the resource by the participant.

Chapter 5

Self-sovereign identity prototype

The main goal of this thesis was to see if Fabric blockchain is going to be up to the task of improving peoples identity recognition process. Securing the private information while proving authenticity. Limiting data provided to the institutions or businesses that require it. Focusing on what the user wants to certificate and authorize. People are not visible to businesses and institutions, unless they initiate an interaction with the respective party, thus providing safety and discretion of the existence of the user.

5.1 Introduction

5.1.1 Identity

Identity is a fluid term. It depends on the context in which this it is used. Through the years the identity of a person was based on family, clan or even community. What's more, a person can identify themselves based on their hobbies, job, belief and others. However, in order to be acknowledged and used, an identification standard has to be accepted.

Nowadays, identity is defined by our passport or any other identity document issued by respective government. This is the so-called bureautic identity. However, because of polit-economic reasons, this identity may prove useless for any purpose in another country. Another problem might occur if the government simply revokes the credentials of a person, as a result that person will cease to exist as far as any institution or business, that requires an ID, is concerned.

A silent threat is when one is required to prove their age for example, they need to show, in most cases, their government issued ID. It not only contains the information of the date of birth, but also more private information, which can be used misappropriately. It might be that the authority that checks for the age of that person to actually be seeking their address.

Christopher Allen, in his paper 'The Path to Self-Sovereign Identity', states that an identity is a uniquely human concept, "one cannot spell and identity without an I" [2]. It is a product of a process build by every second and every choice, conscious or not. The character which forms who we are is in constant shaping. What a person is today, most of the time, is different from what they were ten years ago.

5.1.2 New approach to identity identification

Technology has enabled the possibility for redefining the modern concept of identification document. The idea is to find a procedure, that can give the identity identification back to the individual. In recent years, this process of finding a better possible standard of an ID became known as self-sovereign identity.

Still in its infancy, the new identity concept has not been defined what exactly is. The blockchain technology is giving lots of space to test and experiment, until the final version is set. In order to establish some critical points, Christopher Allen, in the same paper, has outlined ten abstract principles that should be present in every product, that tries to create a self-sovereign system [2]. They are created with the intent to prevent human right abuses and protect the individual. The identity can be a double-edge sword that can be used both for beneficial and maleficent purposes. In the next section, every principle is discussed and point out how it has been implemented into the network.

In contrast to Mr. Allen, who is working on permissionless blockchains to implement his beliefs, the constructed prototype is done in a permissioned one. The main difference is that in permissionless a person will have more freedom of choice, but at the same time, one can abuse the system and create multiple identities.

The permissioned system is less endangered by identity abuse. The trade-off is that a trusted node has to invite a person in the system, thus leaving the final decision to them, whether or not a person is going to enter in the network. In the prototype, only the network administrator can add participants.

IBM and Ethereum Enterprise Alliance have joined into a collaboration to create a cross product between Fabric and Ethereum tools [9]. Hyperledger Burrow, developed by Monax, created special EVM - Ethereum Virtual Machine that can execute Solidity smart contracts in Burrow environment. That EVM is now integrated in Fabric [24]. Eventually there could be an opportunity of the creation of a tool that could connect somehow the two blockchains at some level, that will allow the best of both blockchains to be used.

5.2 Software design

5.2.1 Entities

The prototype is designed to be somewhat close to the reader. Representing a system designed to replicate the real actions that would be done in Bulgaria. The entities included are five participants and three assets, has seven transactions and can emit three events. Person, high school, driving school, university and customs, being participants and diploma, university diploma, driving licence - assets. Transactions - createDiploma, createDrivingLicence, confirmDiploma, confirmDrivingLicence, erollInUniversity, graduateUni, applyToUniversity and events - waitingDiplomaConfirmation, waitingDrivingLicenceConfirmation and applyToUniversityEvent .

The network administrator is going to add only those participants that are meeting the necessary requirements. The system later can be configured to accept participants that are added from another, highly trusted, participant. A class diagram of the prototype can be seen on **Figure 17**.

Participants and assets in the system are designed as:

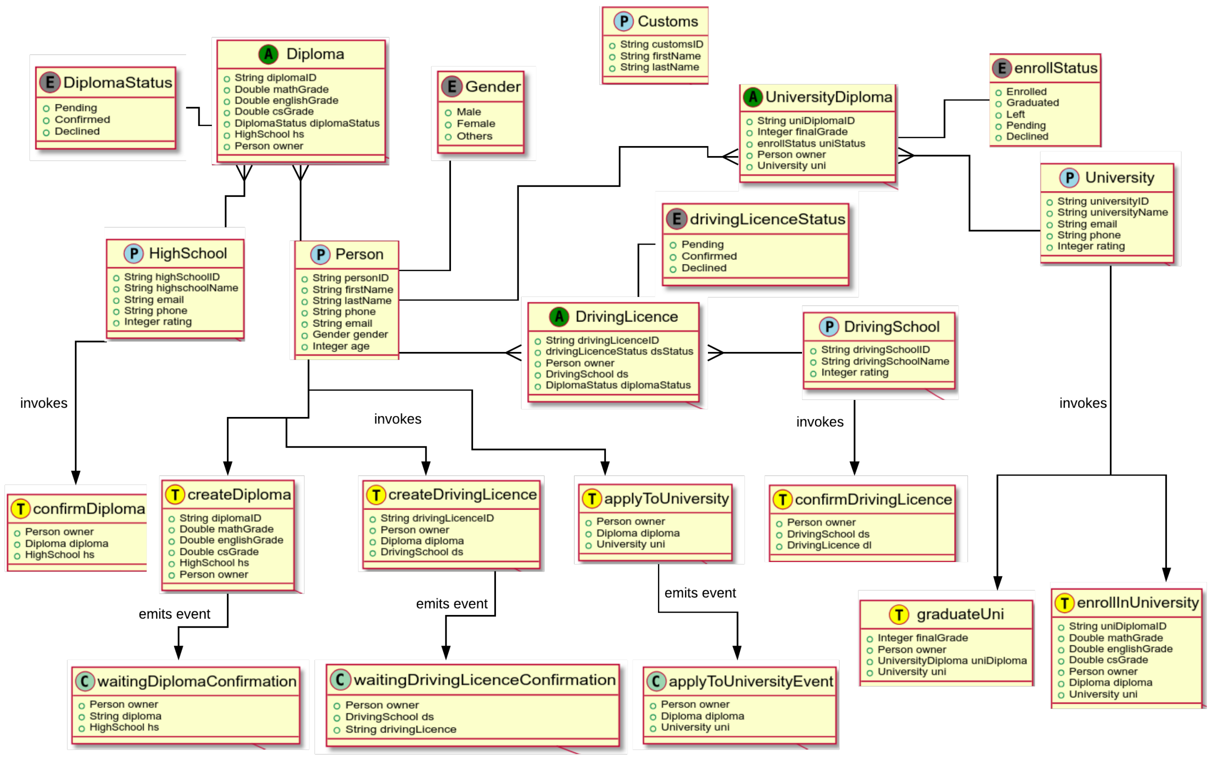


Figure 17: *A class diagram of the prototype*

Person

Has an unique string, personID, that identifies it in the system. In this model the name, age, phone, email and gender are also defining qualities of the person. This participant represents the main end user. He/she can execute the transactions to create their own diploma and driving licence, that then have to be confirmed by the respective participant.

High school

Can be referenced as the administrator of the high school. Since every student will have to know the particular high school's ID. An assumption is made that this will be easily accessible information for the students of the respective school. In the system this entity only confirms the presented document.

Driving school

Can be referenced as the administrator of the driving school. This entity represents those institutions or businesses where one only need to present specific information that is in the form ”Yes, I have/ No, I do not have”. In Bulgaria, in order for a person to start his/her driving lessons, they have to have a certificate for completion of high school and to be at least eighteen year of age.

Customs

Can only see all participants and assets. Do not have the possibility to create, change or update any resource. Monitor for any malevolence.

University

Can be referenced as the administrator of the driving school. The entity represents those institutions that would require more private information. Unlike the driving school, university needs to compare the grade a person has with the one in the conditional offer given. To simulate the common practice, the conditional offers are given to people applied to enroll. So when a person request to enroll in the university, the respective institution will create a transaction. It that will enroll him/her only if that person has a grade equal or more to the one written as part of the transaction.

Having a transaction like that would give some hints about the persons grades, good enough to enroll or not, but ultimately does not provide the exact information. It stays private.

Diploma

Contains information about persons grades and whether it is valid - confirmed - or not. For better security and faster transaction flow, the person, student, is creating his/her own diploma and sends it to the high school for confirmation, instead of the other way around. High schools can only see and have permission to update the students diplomas, where the high schools ID has been referenced. What's more, if the school tries to confirm the diploma of a person, whose referenced school is another, then, because of the access control, the transaction will fail.

Example: if John Doe creates a diploma and references Williams High's unique ID, then WH will be able to see John's diploma. However, if John references Wollas High's, then there are three possible cases.

First, if there is no such school, the transaction will be rejected.

Second, there is such school and they see that he is not a student, they will decline the confirmation.

Third, they confirm, but is the wrong student. This is a violation and Customs are going to demand explanation. The blockchain can be scanned for the necessary facts of this violation. Then the network admin can remove the asset. If the violation has been intentional, the participant high school may be banished as well.

This is possible due to the permissioned nature of the system. In case of malicious intents in the network, the administrator will remove any participant.

Driving licence

Contains information about who are the owner and driving school, as well as whether diploma and driving licence statuses are valid. In the system, similar to the creation of diploma, the person is creating his/her driving licence and sending the digital papers to be confirmed. In contrast to the creation of diploma, in order for the person to be able to submit a driving licence transaction, there are several checks that must be passed.

In the transaction, the user has to provide their participantID, diploma ID and a string for ID for the licence. When the transaction proposal is submitted, the three checks are made. If the transaction is successful, it is added the ledger.

The first check is a comparison between the provided personID and the owner of the provided diploma. Simple check to prevent people from trying to certify their documents with certificate of other people.

The second is a comparison between the persons age and respective country's law. In Bulgaria, the minimum age is eighteen.

The third, and most important, is a check if the diploma is with status ”confirmed”. The importance comes with the fact that it is not needed for the driving school to see all of the information on the diploma or the diploma at all. Thus, securing the user's private data. The system is guaranteeing the authenticity of the authorized asset. Since all participants are screened before allowing them to enter the system. They are assumed to be trusted parties, until the customs catch a malicious one and throw that party out of the system.

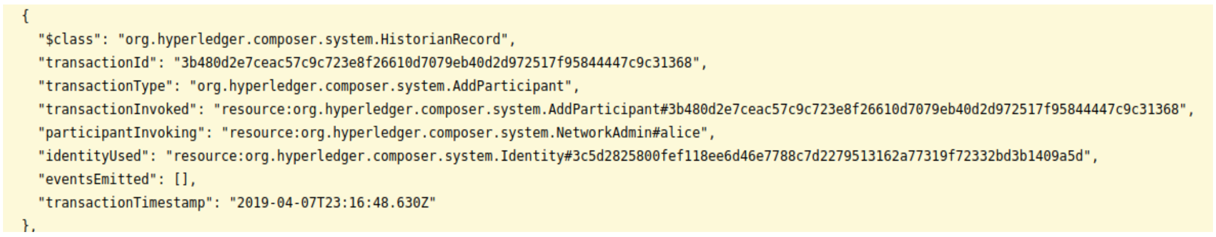
University diploma

It is created by the University. Contains information about the owner and the university, also the status of the student from the university point of view - enrolled, graduated, other. Upon invoking graduateUniversity transaction, an additional field is added - final grade.

5.2.2 Principles followed

The prototype is an experiment to test and conclude whether Fabric is indeed a good base to build upon a self-sovereign identity. This segment will also discuss the ten principles outlined by Christopher Allen.

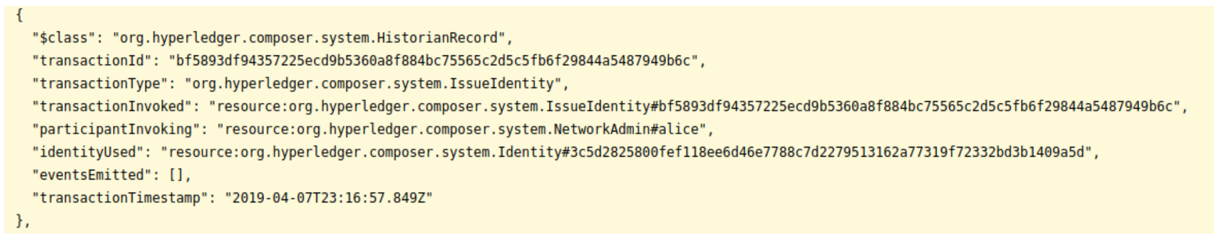
Existence - users must have an independent existence. Once they are accepted in the network no one can invoke their identity. The blockchain does not provide a reference to the newly added participant when it is created as seen on **Figures 18** and 19. The only participants that can observe the identity are Customs. They can observe the whole network, without the possibility to make any changes.



```
{
  "$class": "org.hyperledger.composer.system.HistorianRecord",
  "transactionId": "3b480d2e7ceac57c9c723e8f26610d7079eb40d2d972517f95844447c9c31368",
  "transactionType": "org.hyperledger.composer.system.AddParticipant",
  "transactionInvoked": "resource:org.hyperledger.composer.system.AddParticipant#3b480d2e7ceac57c9c723e8f26610d7079eb40d2d972517f95844447c9c31368",
  "participantInvoking": "resource:org.hyperledger.composer.system.NetworkAdmin#alice",
  "identityUsed": "resource:org.hyperledger.composer.system.Identity#3c5d2825800fef118ee6d46e7788c7d2279513162a77319f72332bd3b1409a5d",
  "eventsEmitted": [],
  "transactionTimestamp": "2019-04-07T23:16:48.630Z"
},
```

Figure 18: *Example of record of adding a participant into the system*

Control - As established in the previous section, there would be no need of this product, if it is not focused on the end user. Thus, in the prototype, the person is initiating a connection with institutions



```
{
  "$class": "org.hyperledger.composer.system.HistorianRecord",
  "transactionId": "bf5893df94357225ecd9b5360a8f884bc75565c2d5c5fb6f29844a5487949b6c",
  "transactionType": "org.hyperledger.composer.system.IssueIdentity",
  "transactionInvoked": "resource:org.hyperledger.composer.system.IssueIdentity#bf5893df94357225ecd9b5360a8f884bc75565c2d5c5fb6f29844a5487949b6c",
  "participantInvoking": "resource:org.hyperledger.composer.system.NetworkAdmin#alice",
  "identityUsed": "resource:org.hyperledger.composer.system.Identity#3c5d2825800fef118ee6d46e7788c7d2279513162a77319f72332bd3b1409a5d",
  "eventsEmitted": [],
  "transactionTimestamp": "2019-04-07T23:16:57.849Z"
},
```

Figure 19: *Example of record of issuing an identity to the participant*

and businesses. This decision is also covering the security of the individual. As he or she would not be bothered by anyone, unless the connection between the two parties is made by the user.

Access - users can access any assets that have their initials on. However, the users are denied the permission to change any of the information in the already created identity or assets they possess. This design is actuated by security concerns.

Even though Fabric is permissioned, thus requirements must be met before a user is added to the network, there are still ways to falsify the information provided. The blockchain can only promise for authentication within the system. If the entry is technically correct, there would not be any logical problems. That is why customs that monitor the whole network are added.

The other case is if a user enters the network and then change their details. In the end the outlaw will be noticed, because of the digital footprints left in the blockchain and they will be banned from the network, however it might be too late to be prosecuted as well. Having that a system like this is going to serve thousands of people, if successful, it will be difficult to check upon the network for any malevalous changes in the identity or assets. Even with the police monitoring all the time, it would still be a challenge.

Transparency - systems and algorithms must be transparent. The prototype is using Fabrics default configuration, so the methods and algorithms they can be found and read in detail in Fabrics documentation [16].

Persistence - identities must be long-lived or be able to be forgotten. The identity of a person, when Hyperledger Composer is used, are stored in composer cards. Administrator cards have certificates that are valid for 1 year, along with information about public key, signature and fabric peer, that is associated with. he data about the participant consists of a public, private key and certificate. It does not have explicitly set time when will the certificate expire, if it does at all.

Interoperability - identities should be as widely usable as possible. At this moment in time, the identities can only be used within the Fabric network. However, due to the mentioned collaboration between IBM and EEA, the usability of the identities has an opportunity to increase.

Consent - users must agree to the use of their identity. It is given implicitly by initiating interaction with given business or institution. Minimalization - disclosure of claims must be minimized, the data given must be minimized. Through the access control language, the exposure of data has been limited with great caution.

In the prototype, when the person is requesting a driving licence, he or she sends references to the documents needed. Then a yes or no check is made. Whether they are fulfilling the conditions made by law or respective institution/business.

In the case with requesting an enrollment to the university, several fields are checked, designed by the logic. The university cannot see any data from the user, they can only see if their conditions are satisfied by the respective person.

Protection - the rights of users must be protected. In the prototype customs are monitoring the network and transactions. They are responsible to observe and send reports for improvement, if there is any security hole, that needs to be patched. In order for one to connect to the system, they have to show their composer card. In section 4.2 is explained the different type of storage of composer identity cards. Even though saving the card on your local is considered to be the safest, it is still not completely secure practice. That is why in eventual steal of those files, the identity can be revoked by a trusted entity.

5.2.3 Sequence diagrams

The prototype has three interaction scenarios. Person with high school, driving school and university. The following are sequence diagrams representing them. For this prototype it is of an essence that the person is having confirmed diploma, otherwise they cannot create a driving licence or apply for university.

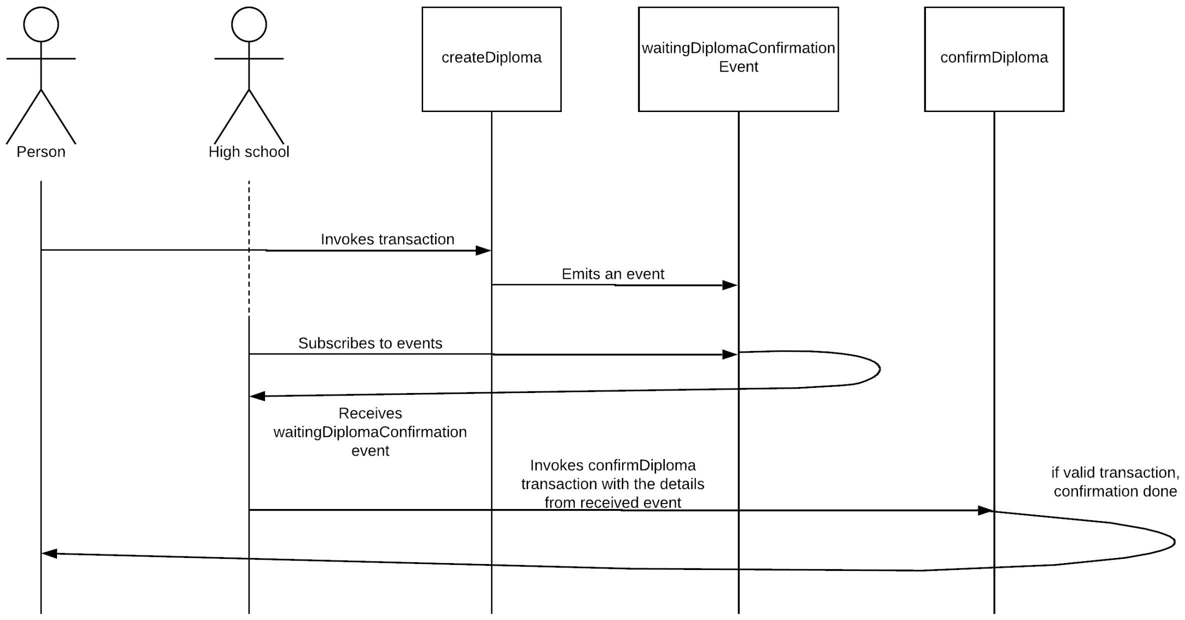

Person
High school
createDiploma
waitingDiplomaConfirmation Event
confirmDiploma
Invokes transaction
Emits an event
Subscribes to events
Receives waitingDiplomaConfirmation event
Invokes confirmDiploma transaction with the details from received event
if valid transaction, confirmation done

Figure 20: *Creating a diploma*

The first diagram, shown on **Figure 20** is representing the interaction between a person and a high school. The person invokes transaction createDiploma with the grades they have from their graduation. Upon submitting, an event is emitted to the network. The high school, which had subscribed for listening on waitingDiplomaConfirmation events, receives it. The high school can either manually execute confirmDiploma transaction or automate it. In the end the person can check his Diploma asset.

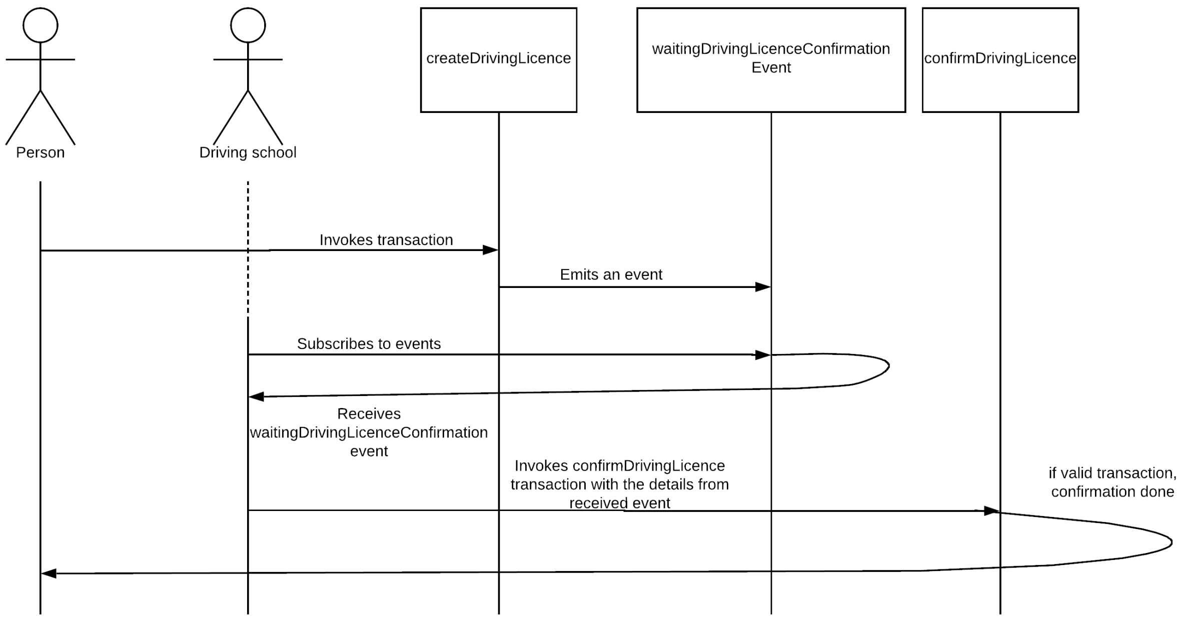

Person
Driving school
createDrivingLicence
waitingDrivingLicenceConfirmation Event
confirmDrivingLicence
Invokes transaction
Emits an event
Subscribes to events
Receives waitingDrivingLicenceConfirmation event
Invokes confirmDrivingLicence transaction with the details from received event
if valid transaction, confirmation done

Figure 21: *Creating a driving licence*

Figure 21 represents the interaction between a person and a driving school. For person to be able to execute createDrivingLicence they must have confirmed diploma and to be older than seventeen years of age. The driving school does not have access or visibility to the diploma asset. However, since the initiator is the person itself, he can see the resource and compare the value. Driving school can trust whether the diploma is truly confirmed by the high school, since only the referenced school in the createDiploma transaction can change this field. Furthermore, the permissioned nature of the system proves for the authenticity of the participant.

If conditions are fulfilled and the transaction successfully submitted, an event will be emitted. The driving school, analogous to high school, can manually execute transaction confirmDrivingLicence, or can automate this process. As a result of a successful transaction, the value of the specific Driving Licence asset is changed.

Shown on **Figure 22** is the interaction between a person and university. The person executes a transaction applyToUniversity. If successful, an event is emitted. The university will receive information about the one who wants to apply for further studies.

University is then taking the participantID provided and must fill out the conditional degrees. Simulating a conditional offer, without knowing anything about the student. If the university has a strict minimum for student's grade, it could automate the transaction with the specific grades.

Once the student has finished their studies, it is assumed, that the university will know the student's participantID and will update their university diploma to 'graduated' with an additional field for the

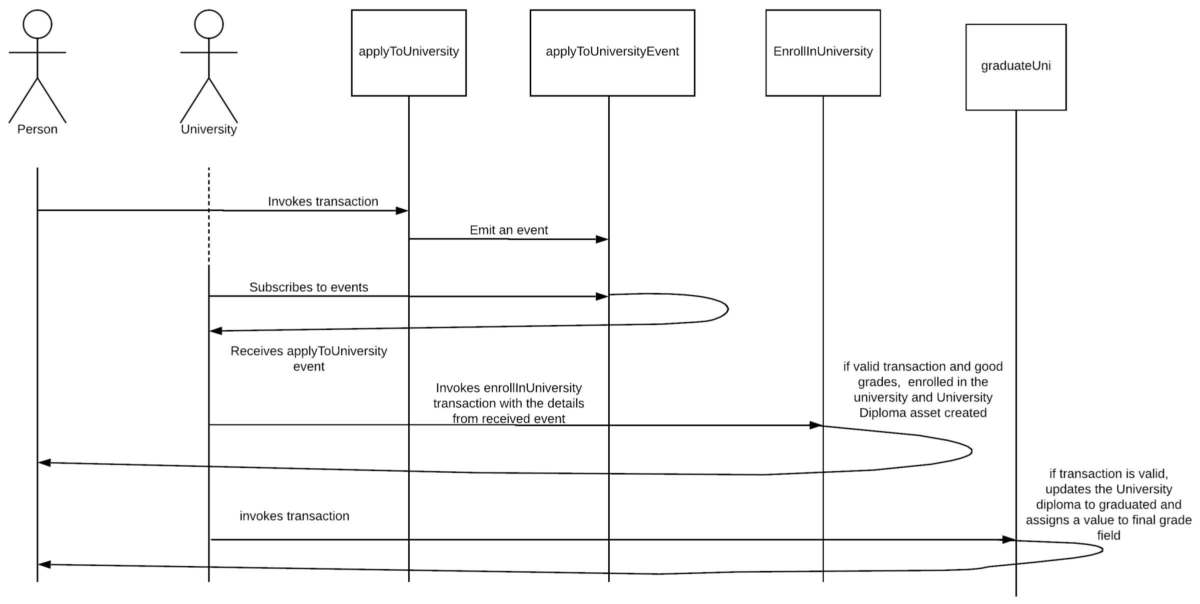

Person
University
applyToUniversity
applyToUniversityEvent
EnrollInUniversity
graduateUni
Invokes transaction
Emit an event
Subscribes to events
Receives applyToUniversity event
Invokes enrollInUniversity transaction with the details from received event
if valid transaction and good grades, enrolled in the university and University Diploma asset created
invokes transaction
if transaction is valid, updates the University diploma to graduated and assigns a value to final grade field

Figure 22: *Interaction between a person and university*

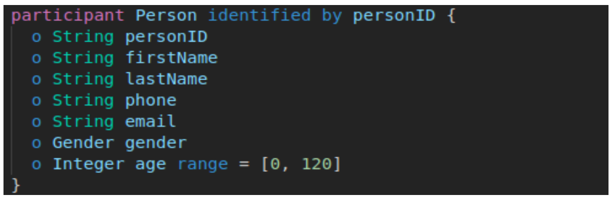
final grade in the range 0-100.

5.3 Implementation

The prototype is using the default configuration of Fabric - two organizations with two peers each.

5.3.1 Adding new unique set of participants

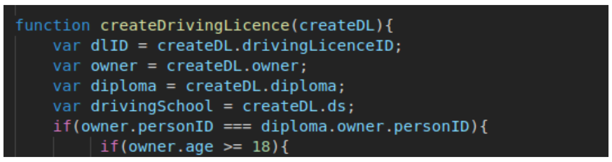
Adding new unique set of participants or assets is done by modifying the model, logic and permissions files. As described in **section 4.4** the model file will contain the data structure, transactions and events of the newly introduced participant or asset. An example of participant data structure can be seen of **Figure 23**.



```
participant Person identified by personID {
  o String personID
  o String firstName
  o String lastName
  o String phone
  o String email
  o Gender gender
  o Integer age range = [0, 120]
}
```

Figure 23: *Person participant identified by personID*

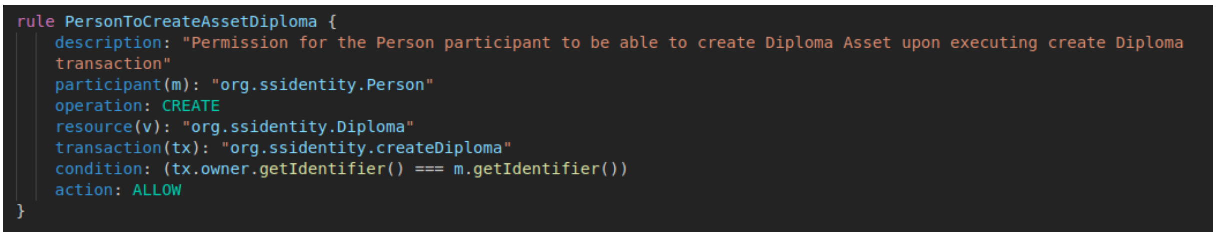
The logic file contains how this asset or participant is going to be used. Maybe one or more transactions and enough events will be constructed depending on the requirements. An example part of function in the logic file that is describing an action with the example participant can be seen on **Figure 24**.



```
function createDrivingLicence(createDL){
    var dlID = createDL.drivingLicenceID;
    var owner = createDL.owner;
    var diploma = createDL.diploma;
    var drivingSchool = createDL.ds;
    if(owner.personID === diploma.owner.personID){
        if(owner.age >= 18){
```

Figure 24: *Example how the Person participant is used for conditions*

The third is important file that needs to be modified is the permissions one. Specifications about who can interact with and see the new resource should be explicitly described rule by rule. On **Figure 25**.



```
rule PersonToCreateAssetDiploma {
    description: "Permission for the Person participant to be able to create Diploma Asset upon executing create Diploma
    transaction"
    participant(m): "org.ssidentity.Person"
    operation: CREATE
    resource(v): "org.ssidentity.Diploma"
    transaction(tx): "org.ssidentity.createDiploma"
    condition: (tx.owner.getIdentifier() === m.getIdentifier())
    action: ALLOW
}
```

Figure 25: *ACL for Person invoking createDiploma transaction*

The experience gathered from this thesis proves that modifying your files in VS Code, with the composer plugin on, is easy and more error resistant. Once the changes are made, the recommended action is to copy and paste them in the online tool - composer-playground. It can serve as easy deployment and test if all the changes are successfully made. Upon successful testing, the advised method to get a business network archive is to export the working local online system from the export button. It is assumed that at this point, the prototype system will be running. To upgrade it with the new unique type of participant, once the correct new .bna file is ready, it has to be installed by the command:

```
composer network install -a NETWORK-FILENAME.bna -c PEERADMIN@SYSTEM
```

If there are multiple admin peers, all of them have to have the new version installed, before upgrading the system:

```
composer network install --card PeerAdmin@byfn-network-org1
--archiveFile ssidentity181.bna

composer network install --card PeerAdmin@byfn-network-org2
--archiveFile ssidentity181.bna
```

And then only one peer is needed to be upgraded by the command :

```
composer network upgrade -c PEERADMIN@SYSTEM -n NETWORK-NAME
-V NETWORK-VERSION
```

```
composer network upgrade -c PeerAdmin@byfn-network-org1
-n ssidentity -V 0.0.2-deploy.181
```

5.3.2 Adding a participant to the system

There are two ways in which a participant entry can be added to the system.First, through composer CLI. Second, through REST API.

The steps to add a new user into the network through CLI are the following. First, adding it as an entry in the terminal.

```
composer participant add -d '{"$class":"org.ssidentity.Person",
"personID":"Daka","firstName":"Daka",
"lastName":"Gospodinov","phone":"4324325",
"email":"daka.gospodinov@stirling.co","gender":"Male",
"age":23 }' -c admin@ssidentity
```

Next step is to issue an identity to this participant, so that the user can use the newly created entry.

```
composer identity issue -u Daka -a
org.ssidentity.Person#Daka -c admin@ssidentity -x
```

As a result, from the command above there will be an output composer card file called Daka@ssidentity.card. There are two cases to continue from this point: First is by importing the card through the CLI by the following command:

```
composer card import -f Daka@ssidentity.card
```

This scenario is easy and good for testing purposes, however, it does not create public and private keys for the new participant. Only a secret is used as a 'mock' participant. Note, when the developer wants to restart the system, all participants will be lost. Upon new start up, all composer cards must be deleted from the previous run. The command to check all existing composer cards is:

```
composer card list
```

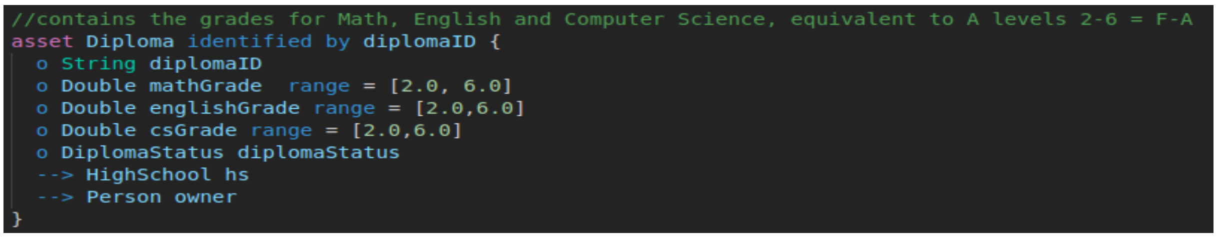
An example of command to delete card:

```
composer card delete -c Daka@ssidentity
```

Second, by having a multi-user REST API. It will give additional field, to manage one's wallet. From then on it depends on the architect which passport strategy is the best to be used. More detail in **section 5.4**.

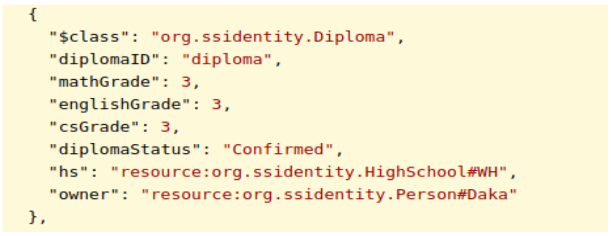
5.3.3 Asset associated with an identity

All assets in the prototype have some kind of relationship, shown by –>explained in **section 4.4.1**, this is a reference to other asset or participant in the system. Through this relationship variable, the logic function can get to the current values of the respective asset or participant reference. The owner of a specific asset is indicated in the 'owner' field as shown in **Figure 26**. **Figure 27** shows how the raw data from the REST API looks like and **Figure 28** shows how it looks like in the generated application by composer.



```
//contains the grades for Math, English and Computer Science, equivalent to A levels 2-6 = F-A
asset Diploma identified by diplomaID {
  o String diplomaID
  o Double mathGrade  range = [2.0, 6.0]
  o Double englishGrade range = [2.0,6.0]
  o Double csGrade range = [2.0,6.0]
  o DiplomaStatus diplomaStatus
  --> HighSchool hs
  --> Person owner
}
```

Figure 26: *An example of asset - Diploma, on the last line is specified a reference to owner of type Person*



```
{
  "$class": "org.ssidentity.Diploma",
  "diplomaID": "diploma",
  "mathGrade": 3,
  "englishGrade": 3,
  "csGrade": 3,
  "diplomaStatus": "Confirmed",
  "hs": "resource:org.ssidentity.HighSchool#WH",
  "owner": "resource:org.ssidentity.Person#Daka"
},
```

Figure 27: *An example of asset - Diploma, shown in REST*

5.3.4 ACL

The access control language is of great importance. This use case is possible largely due to the fact that the control to resources is really strict. A developer using Composer has to be very careful with ACL.

ACL is the place where participants get permission to do CRUD operations on resources. If a participant is not given explicit allowance to do specified operations on a specified asset, participant or transaction, then it will be denied.

A specific complex rules, like the one shown on **Figure 25**.

5.4 REST Interactions

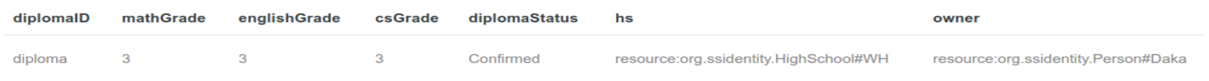
REST, representational state transfer, server is exporting API to be used by web applications.

There are several options that a developer can choose from when starting the REST-server. The more notable are, option for authentication, multi-user, tls and websockets.

In the future, if Composer complete version is released, multi-user REST-server will be the preffered option. It allows users to import their cards, credentials, into the system creating more scalability. However, because of the security flaw, multi-user option is only available if authentiaction is added.

Ernesto Lee and Sudip Ghosh recommend the usage of Passport.js. Passport is authentication middleware for Node.js. It is modular and flexible for different requirement. The official tutorial is recommending github authentication strategy [14].

The multi-user option, adds Wallet field to the API. This way, if a user is both an end user and a representative of an institution, they can easily switch. Security flaw is that, the importing of credentials is still needed. Thus, if a user connects to a malicious rest-server, they will lose their credentials.



diplomaID	mathGrade	englishGrade	csGrade	diplomaStatus	hs	owner
diploma	3	3	3	Confirmed	resource:org.ssidentity.HighSchool#WH	resource:org.ssidentity.Person#Daka

Figure 28: *An example of asset - Diploma, shown in generated by composer app*

5.4.1 Automation

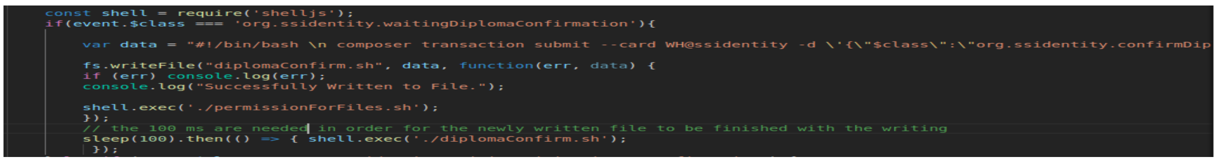
REST API supports websockets for events. When participants subscribe to events, as seen in Section 5.2.3, they can automate some redundant transactions. To do that, they need a web application that allows them to listen to events and be able to configurate the automated response.

The automation is done in several steps:

First, the app listens to the server.

Second, upon receiving an even, it filters it. If the event is one of the expected, the app stores the data and moves to the next step.

Third, a composer transaction command is written in a file with the data collected from the event as shown on **Figure 29**. The reason behind this complex and not efficient solution is because Composer JavaScript API is not clear how to create a runtime transaction when passing relationship values, references to assets and participants.



```
const shell = require('shelljs');
if(event.$class === 'org.ssidentity.waitingDiplomaConfirmation'){

    var data = "#!/bin/bash \n composer transaction submit --card WH@ssidentity -d \'{\"$class\":\"org.ssidentity.confirmDip

    fs.writeFile("diplomaConfirm.sh", data, function(err, data) {
    if (err) console.log(err);
    console.log("Successfully Written to File.");

    shell.exec('./permissionForFiles.sh');
    });
    // the 100 ms are needed in order for the newly written file to be finished with the writing
    sleep(100).then(() => { shell.exec('./diplomaConfirm.sh');
     });
```

Figure 29: *Code showing action upon receiving an event*

Forth, executes the transaction with the help of shelljs node package.

Chapter 6

Conclusion

6.1 Evaluation

This project started as introduction into Fabric, the blockchain modular framework created specifically for business. The milestones of this thesis consisted of gathering knowledge how Fabric works, and how can smart contracts - chaincode - be written. The choice was made to use Hyperledger Composer for the business network, instead of writing low-level code on pure fabric.

By the end of this work, a Fabric system with two organizations with two peers per each is running. If needed, without stopping and losing data, the network can be upgraded with the new changes. When the development of the prototype started, it was known that Composer was not yet finished product. Many different bugs and errors came and went. Until in the end of December it was not announced that IBM are going to stop their development on Composer. Since that decision, the community dropped drastically and instead of updates, there were small patches here and there.

Composer has a lot of bugs. The REST Server is behaving strangely when invoked by the system field. In the prototype there are ACL, which are restricting the visibility of all participants, however, if a developer has left a door for a malicious person to get to all fields in the RESTful server, the attacker may steal a lot of data, that should not be visible.

Composer proved to be slow. A transaction whether through the API or cli, takes about two seconds. The perfect achievement would be if the system is made worldwide and everybody can have their own identity. It is impossible for Composer to ever achieve even a small portion with such speed.

Another error that occurred often was when a new version of the .bna file has been mounted on the Fabric system. The default configuration is for this action to take less than 300 seconds, whereas, Composer always crosses that line on the first run. Potentially could produce a lot of errors and misfortunes when such a crash happens upon upgrading the network. Since it accepted the bna and only failed because of time, Composer throws a new error when the upgrade command is executed again, leaving the developer with two choices. First, have a small little upgrade in order to have a new procedure. The trade off is that it might throw a new time error. Second, put down the system, install the new business network and put it up. If the developer is not keeping any backups, the system will be lost.

Fabric is extremely resourceful tool. There are monthly announcements of new projects with the modular blockchain. In September a system in collaboration with Walmart should be able to give information about all items in the multinational retail corporation. Fabric is also proliferating. Every three to four months a new version is up and running.

The idea of digital identity, that would be focused solely on the end-user is still not clearly defined. However, this thesis proves that Fabric may have an answer to some of the more clearly defined problems. Having a digital identity that cannot be revoked by anyone and used worldwide, can save millions of lives.

6.2 Future Work

Was there a lot more time after the announcement of Composers EOL, this project would be using pure Fabric to achieve its goals. Another interesting experiment would be to see how two machines are interacting on a Fabric network and have a scenario where one is permitting the entry to the other machine.

For future development, Composer should be left outside the scope of useful tools. Although a great proof of concept tool, it may be misleading. On the other hand, if a student learns how to boot up and manage pure Fabric it will be a great feat. It will be especially interesting how the identities issued in the new version of Fabric.

6.3 Further readings

The following books and documentation will improve your understanding of blockchain, Hyperledger Fabric and Hyperledger Composer.

- ”Enterprise blockchain development with hyperledger Fabric and Composer” by Ernesto Lee and Sudip Ghosh
- ”Distributed systems: principles and paradigms” by Andrew S. Tanenbaum, Maarten van Steen
- ”Blockchain for dummies” by Manav Gupta
- IBM Red Books series - ”Developing a Blockchain Business Network with Hyperledger Composer using the IBM Blockchain Platform Starter Plan”
- ”Mastering Ethereum: Building Smart Contracts and DApps Book” by Andreas Antonopoulos and Gavin Wood Ph.D.
- Fabric’s documentation - https://hyperledger-fabric.readthedocs.io/en/latest/
- Composer’s documentation - https://hyperledger.github.io/composer/latest/introduction/introduction.html

References

[1] S. Alboaie and D. Cosovan. Private data system enabling self-sovereign storage managed by executable choreographies. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 83–98. Springer, 2017.

[2] C. Allen. The path to self-sovereign identity. *LIFE WITH ALACRITY BLOG (Apr. 25, 2016), http://www. lifewithalacrity. com/2016/04/the-path-to-self-soverereignidentity. html*, 18, 2016.

[3] Altoros staff. A Blockchain-Based Platform for Automating Bond Issuing Worth 10M. [Online]. Available: *https://www.altoros.com/portfolio/blockchain-based-platform-automating-bond-issuing-worth-10m*.

[4] Altoros staff. Decentralized P2P Securities Demo Using Hyperledger Fabric. [Online]. Available: *https://www.youtube.com/watch?v=-CoS751N8Qc*.

[5] Blockgeeks staff. What is Blockchain Technology? A Step-by-Step Guide For Beginners. [Online]. Available: *https://blockgeeks.com/guides/what-is-blockchain-technology/*.

[6] Blockgeeks staff. What is Cryptocurrency: Everything You Must Need To Know! [Online]. Available: *https://blockgeeks.com/guides/what-is-cryptocurrency/*.

[7] C. Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on distributed cryptocurrencies and consensus ledgers*, volume 310, 2016.

[8] Christina Majaski. Distributed Ledgers Definition. [Online]. Available: *https://www.investopedia.com/terms/d/distributed-ledgers.asp*.

[9] Colby Murphy. Joining forces for the advancement of blockchain technologies. [Online]. Available: *https://www.ibm.com/blogs/blockchain/2018/10/joining-forces-for-the-advancement-of-blockchain-technologies/*.

[10] COTI Staff. Ledgers over the years - from ancient Egypt to blockchain, DAG and beyond. [Online]. Available: *https://medium.com/@COTInetwork/ledgers-over-the-years-from-ancient-egypt-to-blockchain-dag-and-beyond-47924175cb97*.

[11] Docker staff. What is a Container. [Online]. Available: *https://www.docker.com/resources/what-container*.

[12] Dr Deepayan Bhowmik. CSCU9YS: Cryptography-I. Lecture from University of Stirling 2018.

[13] V. V. P. A. S. T. V. Elli Androulaki, Angelo De Caro. Blockchain crypto service provider.

[14] Hyperledger Composer staff. Enabling authentication for the REST server. [Online]. Available: *https://hyperledger.github.io/composer/latest/integrating/enabling-rest-authentication*.

[15] Hyperledger Fabric staff. Fabric Architecture. [Online]. Available: *https://hyperledger-fabric.readthedocs.io/en/release-1.4/arch-deep-dive.html*.

[16] Hyperledger Fabric staff. Fabric CA Users Guide. [Online]. Available: *https://hyperledger-fabric-ca.readthedocs.io/en/release-1.4/users-guide.html*.

[17] Hyperledger Fabric staff. Ledger. [Online]. Available: *https://hyperledger-fabric.readthedocs.io/en/latest/ledger/ledger.html*.

[18] IBM staff. Maersk and IBM Introduce TradeLens Blockchain Shipping Solution. [Online]. Available: *https://newsroom.ibm.com/2018-08-09-Maersk-and-IBM-Introduce-TradeLens-Blockchain-Shipping-Solution*.

[19] Ivan Vankov. Genesis block configuration. [Online]. Available: *https://www.youtube.com/watch?v=nF9fkBuBNpE&t=4s*.

[20] B. Li, Y. Wang, P. Shi, H. Chen, and L. Cheng. Fppb: A fast and privacy-preserving method based on the permissioned blockchain for fair transactions in sharing economy. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1368–1373. IEEE, 2018.

[21] A. N. Mencias, D. Dillenberger, P. Novotny, F. Toth, and Morris. An optimized blockchain solution for the ibm z14. *IBM Journal of Research and Development*, 62(2/3):4–1, 2018.

[22] D. C. Mills, K. Wang, B. Malone, A. Ravi, J. Marquardt, A. I. Badev, T. Brezinski, L. Fahy, K. Liao, V. Kargenian, et al. Distributed ledger technology in payments, clearing, and settlement. 2016.

[23] Nolan Bauerle. What is Blockchain Technology? [Online]. Available: *https://www.coindesk.com/information/what-is-blockchain-technology/*.

[24] Raheel Zubairy. Develop a blockchain application with an Ethereum smart contract deployed on a Hyperledger Fabric network. [Online]. Available: *https://developer.ibm.com/announcements/loyalty-points-fabric-evm/*.

[25] SecureKey CIO Andre Boysen. How blockchain is changing digital identity. [Online]. Available: *https://www.youtube.com/watch?v=EQ5PGPIjrtI*.

[26] Simon Stone. Composer TSC update. [Online]. Available: *https://lists.hyperledger.org/g/composer/message/125?fbclid=IwAR1kS_QyYLjrKovql_PU6UGHf_fjsmZKyHvk4GjUHFHx-laO9AF1T72TsoA*.

[27] SSL2Buy staff. Symmetric vs. Asymmetric Encryption What are differences? [Online]. Available: *https://www.ssl2buy.com/wiki/symmetric-vs-asymmetric-encryption-what-are-differences*.

[28] Stack Overflow staff. Developer survey results 2018. [Online]. Available: *https://insights.stackoverflow.com/survey/2018/*.

[29] Techworld Staff. The most infamous data breaches . [Online]. Available: *https://www.techworld.com/security/uks-most-infamous-data-breaches-3604586/*.

[30] Verify.Me staff. Website of the company. [Online]. Available: *https://verified.me/*.

[31] W3Schools staff. What is npm? [Online]. Available: *https://www.w3schools.com/whatis/whatis_npm.asp*.