Advice Report

Windesheim

Spark! Living Lab Conditioned Goods

Version 1.0

Authors:

Name	**Student number**	**Email**
Clément Perucca	s1177124	s1177124@student.windesheim.nl
Florian Paul	s1177122	s1177122@student.windesheim.nl
Thibaut Ribe	s1177132	s1177132@student.windesheim.nl
Nick van de Poel	s1139131	s1139131@student.windesheim.nl
Mischa Heimenberg	s1111178	s1111178@student.windesheim.nl

Version History

Version	When	Who	What
0.1.0	2022-01-02	Mischa Heimenberg	First Setup
0.2.0	2022-01-07	Nick van der Poel	Merged the 2 existing advisory reports
0.2.1	2022-01-10	Clément Perucca	Adding content to the introduction and advices about the configtx file
0.2.2	2022-01-11	Florian Paul	Adding content in every part and finalizing the report
0.2.3	2022-01-11	Thibaut Ribe	Wrote chapter 2: The product
1.0	2022-02-14	All Authors	Final version

Distribution Management

Version	When	Who
1.0	2022-01-14	Windesheim

Table of contents

1. Introduction

We are a group of students from different nationalities and backgrounds who have come together to work on behalf of the Windesheim Zwolle on a project for the external organization Spark! Living Lab.

We are all attending the security engineering minor at Windesheim Zwolle through which we have gotten the opportunity to work with Spark! Living Lab on their “Conditioned goods” project. We are the third iteration of developers working on the blockchain project hoping to deliver an application of blockchain with data-assurance that is usable for Spark’s client LambWeston. We define data-assurance as the process(es) we use to increase the trustability of the data-collected by our application[1][2].

Spark! Living Lab is a research company focused on applying IoT and blockchain to the (cold) supply chain industry. Spark accepts projects from various companies depending on a (cold) supply chain. Spark’s goal is to try and figure out how their clients can benefit from IoT and/or blockchain. Currently a lot of companies are experimenting with applying blockchain and/or IoT and are looking for a way to adopt these technologies, that’s where Spark steps in.

LambWeston-Meijer (LWM) is one of those clients looking to apply IoT and blockchain to their cold chain. LWM is a large manufacturer of french fries and other processed potato products. They distribute their goods all over Europe using coldchain. In this kind of chain, the produced goods are kept in controlled environments targeting cold preservation of goods. From the moment they are being processed in the factory until the moment they are stored in the customers’ fridge, the temperature is to be kept within pre-agreed boundaries. Something that is monitored manually at the moment could be automated.

We have made advisory, design[2] and analysis[1] documentation. This document is meant to be read after all those documents mentioned before in order to fully understand what we have learned from our researches and developments over the past few months. In chapter 2 we will be describing the final product we delivered, in chapter 3 we will discuss the findings we came across during the timespan of the project, in chapter 4 we suggest improvements which could be made and the reasoning behind them and lastly in chapter 5 we voice our advice for the advancement and future of the project.

2. The product

This chapter will briefly describe the final product.

We used hyperledger fabric to make a single node network operational. We made a tutorial video to show how to deploy the network and how to run the API to start the server and execute the chaincodes[3].

We first made our proper vision of the lifecycle based on information gathered from stakeholders which we verified with our product owner, assuring to have the same vision and goals of the project as our stakeholders.

We implemented our design of the lifecycle of a shipment and data trust using the smart contracts with which you can create shipment with an unique ID. It is also possible to create a pool of trusted sensors with unique ID by registering them and saving the list in the blockchain. You can add a sensor from this pool to a shipment. Each sensor will have a measure that will be updated. That way it is possible to keep track of the temperature for example:

Each shipment now has an owner to keep track who's responsible at any moment of it. To change owners a transaction is mandatory. We made a two way agreement for it in which each part of the transaction(current owner and new owner) have to agree. The shipment should also validate the SLA which is currently having a temperature sensor register on the shipment and this temperature should be between a minimal value and a maximal.
Since we do not actually have real sensors we can add manually the measure with the chaincodes. Our API permits the peers to execute the functions of the smart contracts using POST requests. We also clarified some hyperledger fabric notions like the configtx file and made documents to explain them.

3. Findings

This chapter will describe all findings the project group made during the project and the realization of the PoC.

The network made by the previous group was not working so we decided to use a single node network which is easier to set up and more appropriate for testing.
Nevertheless, it is theoretically possible to deploy a production network with every organization.

What we have learned about this is:

- Use Fabric-CA in production for certificate generation
Although cryptogen is often provided to be a simpler alternative to fabric-CA it is recommended that it should only be used in development. Cryptogen only generates self-signed certificates and that's why the certificate cannot be verified by a 3rd-party. Fabric-CA also requires new users to be registered and enrolled by the CA-admin of that organization.

-We decided to use Raft instead of Kafka.
Raft is easier to deploy, better designed for large networks (decentralization with these protocols is useless in small networks)
The problem with Kafka is that you will need one Kafka cluster for each organization. However with Raft, each organization can have its own ordering nodes, participating in the ordering service, which leads to a more decentralized system.

- Make sure that all versions match and are up to date
As we implemented the notion of ownership, we also implemented the two-ways agreements. In fact, the transfer of ownership to be agreed between the actual owner and the future owner. Consequently, we created a system where the actual owner needs to send a request to transfer the ownership of the transport and then the future owner accepts that transfer. During all this process, we also check that the SLA is verified.

We also did not have time to properly set the access control by using the configtx file. In fact, we could have done it in the smart contracts nevertheless it would have been a short time solution since in the smart contract we can implement access control by matching MSP ID. Using this solution implies that every time that we want to add an organization to the network we have to redeploy the smart contracts on every peer which takes time and can lead to bugs.

Using the configtx file will be easier since we just have to add the new organization in the Organization part of the configtx. For more details, we can read the HLF Channel configuration that we have done.

4. Improvements

This chapter will describe all improvements that could be made to the final product. We have described our reasoning as to why we believe the improvement should be made.

List of possible security related risks and events

Make a list of possible security related risks and events that could harm the integrity of the data. It was one of our main goals at the beginning but it hasn't been done because of a lack of time. In fact, we thought that we were going to do the securitization of the system but after our second iteration of goals, we chose to put the priority on deploying a system working well instead.

Authentication to the API

API would also need some authentication because otherwise anybody would be able to invoke chaincode on someone else's behalf. Everybody could invoke chaincode as an admin and so to have access to all the functions for example. Consequently, the data has no integrity making it untrustworthy and unusable in legal situations.

Investigate more in the configtx.yaml file

We found that the configtx file is managing the channel configuration and consequently the classification of each organization. We configured it in our code but not completely because of a lack of time. Thanks to that, we could add access control with the system of classes and reach the data integrity and security.

Docker upgrade

Upgrade the current single node system into the entire system making it work on Docker because single node systems are just usable for tests and research but not for real production systems.

Test of visibility from outside the chain

Try to add another organization in the system but out of the initial supply chain to see if it can read the information inside the supply chain => to test if the data is readable from the outside and verify our work.

Add the sensors

Implement the sensors to the system when they will be available. It would bring more automation, more speed and more trust in the system because sensor data can be secured and not employee's notes. However the complexity of connecting and adding sensors to the production environment should be seriously considered to become a project on its own. We foresee the research and knowledge needed for a first sensor integration to be quite extensive and could quite possibly lead to some complications before getting it to function correctly.

GPS upgrade

Use GPS to know when transports have arrived at their destination in order to validate the transport and conclude the transaction (to implement functions managing it in smart contracts); It would make the system more automated but also let the companies follow the progress of the transport on its way.

Generalization

Currently, parts of the applications are specifically tailored to this use case. For example, temperature measurement or validating the SLA. To make it smoother to implement additional measurements or features It would be beneficial to make this process more generic to also make it fit other use cases easier if this POC would be reused as a basis for future projects. This would also make this project more valuable for companies that do not have the resources to develop this from scratch. Because their application could be based on this current POC.

Indication of a completed shipment

There is currently no indication whether a shipment has truly arrived at its final destination or that the shipment is stuck on a specific location. In order to fix this a new property will need to be added to shipments.

5. Advice

This chapter will describe the project group's advice based on findings made during the project and the realization of the PoC. The longer version of the abstract.

During the project, the team quickly became aware of the size and complexity of Hyperledger Fabric. The complexity is one of the main downsides of using Hyperledger Fabric and the main source of the issues found.

As the project group had no previous experience with blockchain development, the development heavily relied on documentation. This documentation was often very high level and did not provide a lot of information about the infrastructure on a technical level. This made development, designing, and deploying of the network remarkably harder, and slowed down the progression significantly.

Combined with the time constraints on the project caused some issues. When others, such as an IT-department, try to implement a Hyperledger fabric network without previous experience with blockchain or Hyperledger Fabric the lack of infrastructure technical documentation will almost certainly impact the development duration and quality.

To overcome this problem, a team of Hyperledger Fabric experts could be hired to design and deploy the network according to best practices.

The previous group thought that another possible solution was to use other platforms for blockchain but the theoretical best way to ensure data security and integrity is based on the execution order within Hyperledger which is proper to it.

An important part to improve is about the authentication to the API. The reason is obviously because we don't want anybody to invoke chaincode on another user's behalf. So the solution could be to create separate users for everybody in order to limit the functionalities to the dashboard. For example, we could give a user only the right to read the data but not to modify it.

There are also some improvements that could be made to improve the current proof of concept. For example, the PoC right now uses pre-generated certificates, which is not safe in a production environment. These certificates need to be generated by the companies themselves. It is also wise to use an off-chain database for the dashboard to improve performance and limit the number of queries to the blockchain. Query operators should be avoided if possible because they scan the whole state database which will decrease the performance.

Finally, we implemented access control in order to have a level of authorization and authentication and it could be upgraded to be used on sensor data directly.

The project team believes that using blockchain could be a viable solution and the integrity of data is of great importance. If the data integrity is well ensured by experts, this system could be used as a base to a generalization of this type of security and would be very useful for the blockchain technology in the future, which could be very interesting for Spark! Living Lab.

6. Lexicon

Nodes: Nodes are the communication entities of the blockchain. A “node” is only a logical function in the sense that multiple nodes of different types can run on the same physical server. What counts is how nodes are grouped in “trust domains” and associated to logical entities that control them.

There are three types of nodes:

- Client or submitting-client: a client that submits an actual transaction-invocation to the endorsers, and broadcasts transaction-proposals to the ordering service.

- Peer: a node that commits transactions and maintains the state and a copy of the ledger. Besides, peers can have a special endorser role.

- Ordering-service-node or orderer: a node running the communication service that implements a delivery guarantee, such as atomic or total order broadcast.

Orderer : The Orderer is responsible for packaging transactions into Blocks and distributing them to Anchor Peers across the network.

Anchor peers: A peer node on a channel that all other peers can discover and communicate with. It makes sure peers in different organizations know about each other.

Raft: The Fabric implementation of the established Raft protocol uses a “leader and follower” model, in which a leader is dynamically elected among the ordering nodes in a channel (“consenter set”) and that leader replicates messages to the follower nodes. So, the data is sent only to the leader nodes and then replicated in all the nodes. Because the system can sustain the loss of nodes, including leader nodes, as long as there is a majority of ordering nodes remaining, Raft is said to be crash fault tolerant.

Kafka: Definitely similar to Raft, uses the same protocol of leader and follower and is also CFT but the deployment is really more complicated.

7. References

[1] Heimenberg, M., Van de Poel, N., Ribe, T., Paul, F., & Perucca, C. (2022, januari). Analysis Report. https://docs.google.com/document/d/1I2iwNtwg_bNyPvLr5PIG7Ox-zUSQkyoX0faKSvo2Xe4/edit?usp=sharing

[2] Heimenberg, M., Van de Poel, N., Ribe, T., Paul, F., & Perucca, C. (2022, januari). Design Document. https://docs.google.com/document/d/1nG2Hw0pzG_eeM_X2_piNJOSg0uGKVBvjOzWDhyqo5cE/edit?usp=sharing

[3] set up the network and start the api demo. (2022, 11 januari). [Video]. YouTube. https://www.youtube.com/watch?v=n1Wl3cuGX5U