Tutoring Internship Report
Project: Smart IOT for Mobility, Technical challenges

Author: Mr. Luc Gerrits

Tutor: Mr. François Verdier
Supervisor: Mr. Roland Kromes

University Côte d'Azur - Science Faculty - Electronics Department

Summary

I - Preface and acknowledgement

1. The work space

For a couple of months I did a tutorship at LEAT[1] (Electronics, Antennas and Telecommunications Laboratory), a public science and technology institution based in Sophia-Antipolis.
The LEAT laboratory works mainly on :

- Antennas, electromagnetism and microwaves
- Communicating objects, Wireless network optimization, Embedded systems and Systems on Chip (SoC)

Thanks to the laboratory, we had at our disposal a room to work whenever needed.
I worked on the project Smart IOT for Mobility (SIM) a technology for a new economy, using smart contracts (with blockchain technology). The purpose of this tutorship is the implementation of blockchain access with smart contracts within IoT platforms.[2]

Through the assignment, I did not only gain a lot of knowledge but more importantly, it was a great opportunity to sharpen my skills in a professional working environment.

2. Acknowledgement

I am pleased to express my gratitude to Mr. François Verdier, Professor at the University Nice-Sophia Antipolis, member of the LEAT laboratory and my tutor for this tutorship.
Mr. Verdier did give me very valuable in-time instructions and put me in contact with Mr. Roland Kromes who gave me extensive guidance regarding many practical issues.
I worked with Mr. Sylla Sakoba, who is also doing a tutorship on the SIM project. He is a classmate in the first year of the electronics master degree and is in charge of making the link between the electronic and legal challenges of the project.

II - Introduction

For this tutorship I have worked in the LEAT laboratory among with the other tutored. Mr. Sakoba and I were under the supervision of Mr. Verdier. He introduced us to the project and thanks to him we could directly start by reading documentation, reports and white papers provided by Mr. Verdier. The report created by Mr. Massiera[3] helped a lot to understand the main concepts of blockchain and the different solutions provided by different companies.

1. Definitions

A *blockchain*[4] is a decentralized, distributed database that is used to maintain a continuously growing list of records, called blocks. Each blocks are linked (with cryptographic hash) to the previous block. It's also called a *distributed ledger*.
Three main features[5] of blockchains are:

- Decentralized: The blockchain database is shared among potentially untrusted participants and is identical on all nodes in the network. All participants have the same information.
- Immutable: The blockchain database is an unalterable history of all transactions that uses block hashes to make it easy to detect and prevent attempts to alter the history.
- Secure: All changes are performed by transactions that are signed by known identities.

Blockchain is used in many fields: it is a distributed ledger, so it could store any data you want. For example cryptocurrency[6], cloud storage[7], preventing voter fraud[8] and much more.

A *smart contract* is a transaction protocol that executes the terms of a contract. The contract is a program that run inside of the blockchain. Smart contracts allow trustworthy transactions without third parties. These transactions are permanent and can be tracked like any other blockchain transaction.[9]

An *IOT* meaning "Internet of things" is a network of devices (i.e. everyday objects) that contain electronics, programs and connectivity which allows these devices to connect, interact and exchange data.[10] [11]

An IOT has often limited resources. Unlike a computer, IOT processors aren't really powerful, have little or even no user interface, limited storage and limited connectivity. [12]
With those architectural challenges the goal is to implement blockchain access with smart contracts within these IOTs. [13]

2. Interactions and environment

The project has an important social challenge. As explained in the white paper about Smart IoT for Mobility [13] trust must be ensured in order that individuals use a digital service. Mr. Sakoba and Mrs. Marta Ballatore are the people in charge of this part of the SIM project.
There is also a huge legal challenge: blockchain applications are supposed to operate in a world that is regulated by traditional law rules. The legal system will always have a word to say in case there is a technological fail or any other issue.

III - Specifications of the project

As part of the Smart IoT For Mobility project (SIM) I was intended to :

- Read & understand previous reports, white paper and documents related to the project
- Install & test a blockchain technology
- Setup a network with that blockchain technology
- Run Smart Contracts on a blockchain network

A numerous of difficulties presented themselves during the programming of the distributed ledger software:

- Compatibility issue
- Unknown program language
- Software still in beta
- Networking issue

These issues slowed the progress of some tasks, they will be better explained in the following parts.

IV - Discovering the technology

I started by reading documentation to be able to better understand the notions of blockchain, smart contracts, consensus and transactions.
With the report of Mr. Massiera Nicolas, I was able to get a great overview of the main lines of the SIM project. The bibliography of his report helped discovering documents about multiple distributed ledger. His report made me understand that such a technology could be deployed on IOT's and that solutions still need to be discovered to be able to make it work.

After considering a couple of different blockchain technologies, two stood out:

- Ethereum [14] [15] (licensed under the GPLv3, LGPLv3, MIT license)
- Hyperledger Sawtooth[16] [17] (licensed under the Apache License Version 2.0 software license)

Note: Hyperledger Sawtooth is one of the many blockchain solutions provided by Hyperledger. There are other Hyperledger blockchain frameworks.

The reason Mr. Roland Kromes and I decided to work on these two blockchains is because these technologies are flexible and have friendly implementation on IOT devices. Smart Contracts compatibility between these two blockchains is possible because they have Virtual Machine engines (for Ethereum and called transaction processors in sawtooth): this means they can execute code (i.e. the contract) that was deployed into the blockchain. A project (called “Seth”) is a support for running Ethereum Virtual Machine smart contracts to the Hyperledger Sawtooth platform.[18] [19] [20]

V - Structure of a blockchain

I used Hyperledger Sawtooth as a distributed ledger and Mr. Roland Krome worked on Ethereum.

1. Hyperledger Sawtooth

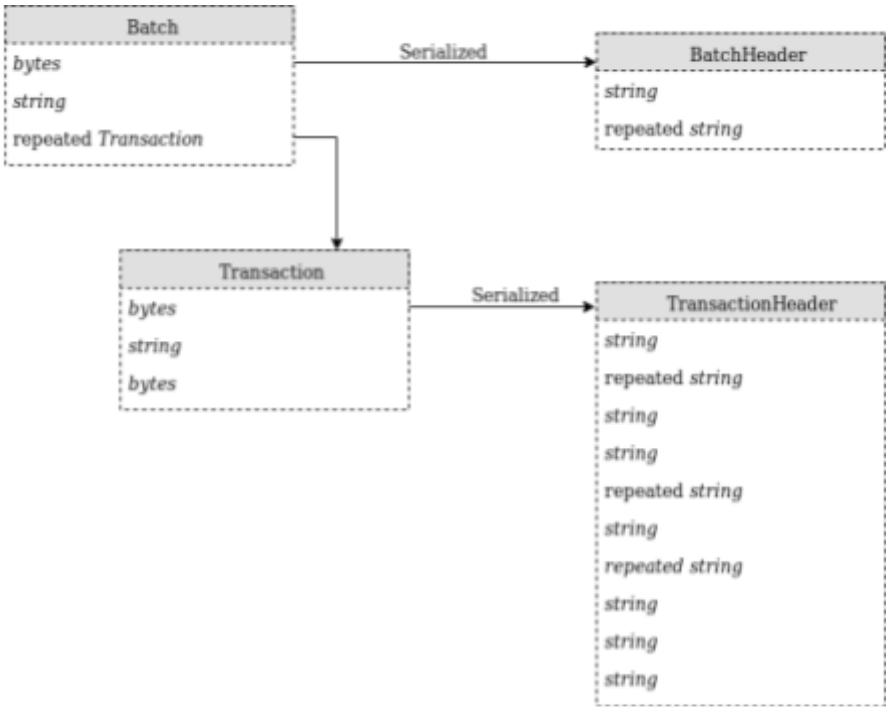
Hyperledger Sawtooth is an open source project developed by Intel under the Hyperledger umbrella. Starting in December 2015 by the Linux Foundation, Hyperledger is an association of projects related to blockchain. [21]
Sawtooth includes:

- A dynamic consensus feature, enabling hot swapping consensus algorithms in a running network
- Unlike Proof of Work[6] (based on computing power), Sawtooth allows Proof of Elapsed Time (PoET), a novel consensus protocol using a lottery-like algorithm. PoET helps to build networks with a large number of nodes with small CPU consumption footprint.
- Smart contracts can be Ethereum compatible.

Important key points to better understand Sawtooth architecture :

1. Any data needing to be persisted is stored in the on-chain key-value storage distributed among the network of validators.
2. This storage allows any binary data to be stored in it.
3. Data is modified with transactions.
4. Every transaction has an associated transaction family and is processed by a corresponding handler. The analogy of contracts in Sawtooth are called Transaction Processors (TPs)[22].
5. A transaction family handler takes the transaction payload, parses it, performs the required business logic (smart contract’s code) and applies corresponding changes to the storage.

Unlike Ethereum, the code of handlers (smart contracts) can be written in any language (you only need support for a couple of languages) and/or stored off-chain. *Off-chain* means that when a transaction is executed, changes happen locally (on-chain refers to changes happening on the entire blockchain).[23]



Batch-Transaction structure

Using batches has its strengths : a batch is the atomic unit of change in the system. If a batch has been applied, all transactions inside the batch will be applied. If a batch has not been applied (probably because one of the transactions is invalid), then none of the transactions in the batch will be applied.[21]

Hyperledger Sawtooth is an ongoing project with one stable version available. We did our best to install, deploy and to run this software.
Note: a new version was just released on december 2018.

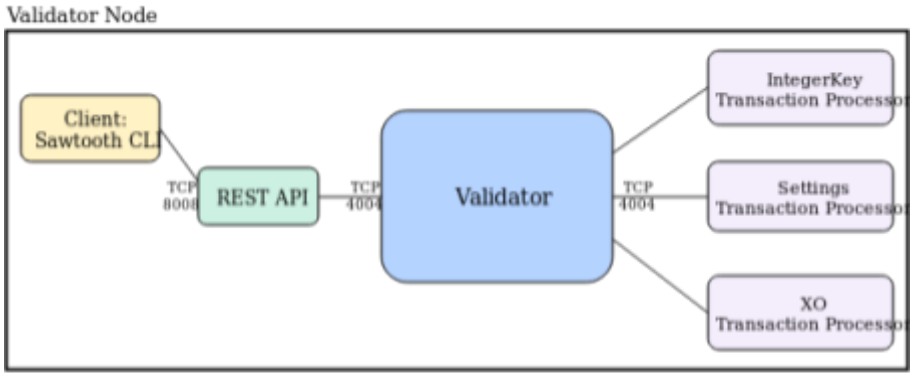
2. Installing Sawtooth: single node

There are three ways to install a Sawtooth blockchain node, it can be done with: Docker, on Ubuntu 16.04 or via Amazon Web Service.

The issue executing sawtooth on Ubuntu 16.04 is that we would need that exact operating system version. With Amazon, the issue is that we would need to use there pre-build

systems, i.e. manual tweaking will be difficult. This is why we use Docker to run the blockchain.

Docker is a program performing containerization, it is an Operating-system-level virtualization tool. It allows a container to use the operating system features. Programs running inside can only see the container's contents and devices assigned to the container.
Docker is a powerful tool that handles many network configurations. It fits perfectly to build a development blockchain environment. [24]



One validator

The version 1.0 of Sawtooth works perfectly to run a single node. A validator node (or peer) is a blending of six different containers, each one having its own purpose. The containers are called a :

- “Validator”, using dev-mode consensus (doesn’t require multiple nodes)
- “REST API” connected to the validator
- “Settings” transaction processor
- “IntegerKey” transaction processor
- “XO” transaction processor
- “Client”, a container that can run Sawtooth’s command line program.

Note: Not all of these containers are essential to run the node. TP’s can be swapped in and out.

The Settings transaction processor allows to store blockchain settings.
The IntegerKey transaction processor container, having a program (named intkey) that can store key-value variables on the deployed ledger.
The XO transaction processor contains an implementation of the popular game *Tic-tac-toe* available through the xo command. By running this game, it checks if the XO transaction processors run correctly.

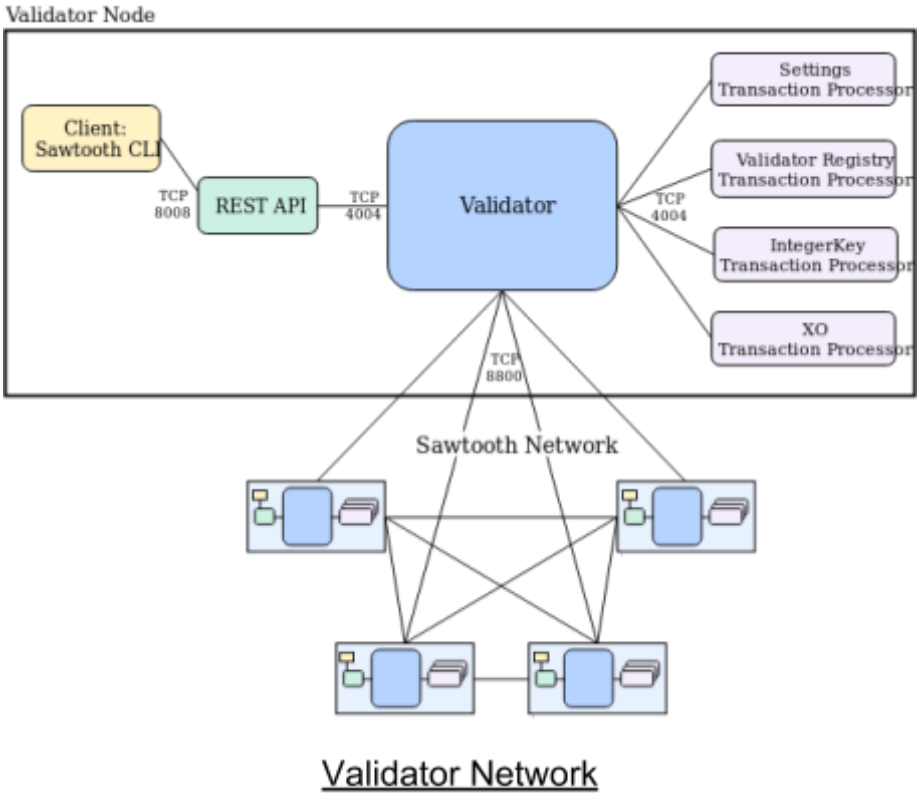
VI - Building a blockchain network

1. Default configurations: all nodes are validators

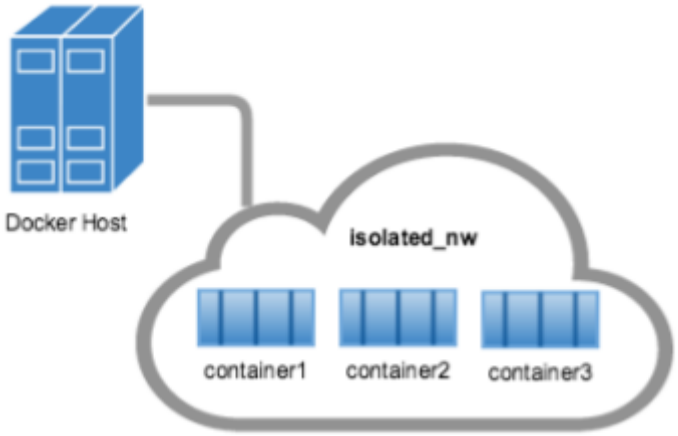
I have had a lot of issues starting a Sawtooth network due to Docker containers version not being available.
The version 1.1 was released in december 2018. After that, I could use the Docker files version 1.1. [25]

1.1. A stable and functional network

The configuration for a Sawtooth network, used in our development environment, is as following :



The sawtooth network is the assembly of multiple single validator nodes. Each node runs a single validator, a REST API, and one or more transaction processors. It uses a consensus initialized by one of the validators.[26]
There are, for each node, six different Docker containers. So for five nodes, there are a total of thirty containers running simultaneously. There is an additional transaction processor and a PoET Validator Registry that handles PoET settings for this multiple-node network. There is only one shell container for this Docker environment.
Each node has at least four peers (=node validator) connected to himself.
The Docker environment allows to separate the containers network from the machine system.



Host - Docker network

It is interesting to have an isolated network because it simulates perfectly the blockchain network environnement. Thus, it is also possible to open certain ports of the isolated network to the host machine. In other words, we can access from the host machine, the REST API of one of the validators node to be able to send requests to the Sawtooth network.[27]

Important: Because this blockchain is on an isolated network, we need to define each port and IP. This was the origin of many problems. Consequently, a simple network misconfiguration had blocked the success of one of the node validator to connect to the other nodes.
In fact, when opening the same port linking, the Docker isolated network and the Docker host (i.e. my machine), the isolated network couldn't have anymore access to that port. The node in question was supposed to initialize the consensus between all peers : Nothing could even be initialized from the start.
After solving this issue, the configuration worked with success. All nodes had, listed in their peers, at least four other nodes. We tested, with the IntegerKey transaction processor tool, to send batches of transactions to the Sawtooth development network.
By accessing the REST API in the browser (with a specific port), we also viewed the created batches and transaction list.

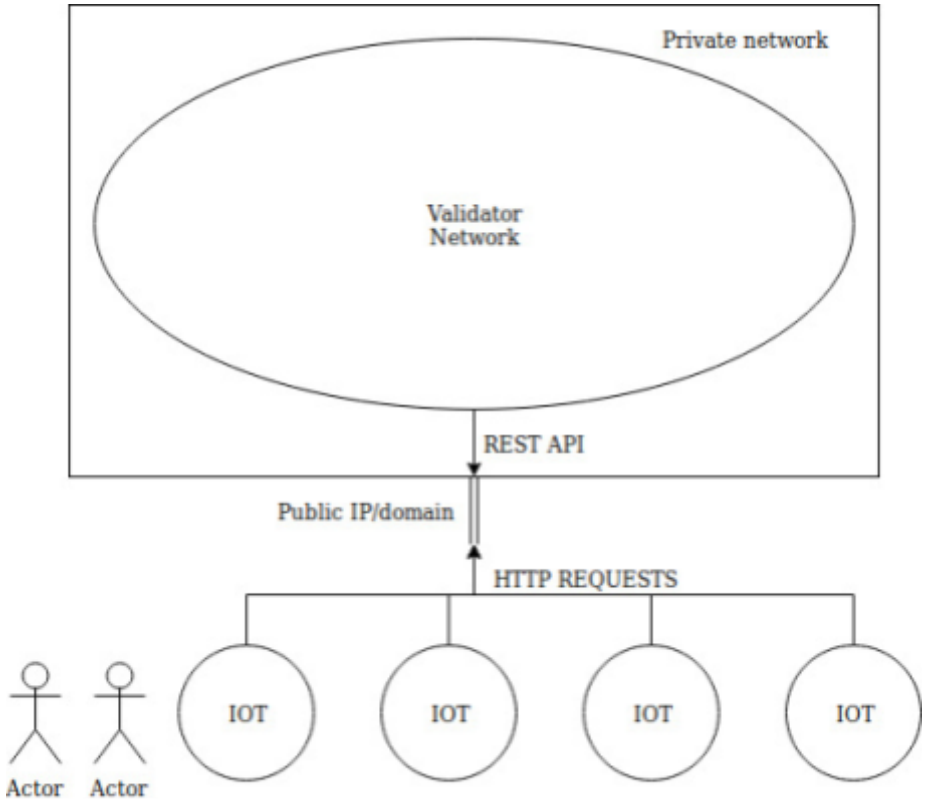
Our setup was five validator nodes, with a minimum of 3 nodes (otherwise a node risks to win to many transactions). The documentation states that 2 nodes per identity is a good start.[30]
Testings on this environment are the same as the single node environment.
The default smart contract worked correctly. The test consists into playing the tic-tac-toe game. The game tests if transaction processors are functional by sending, reading transactions and communicating with the validator.
I could always view the sawtooth blockchain state by viewing the validated blocks via a provided REST API.

1.2. Going further

To go further, we should build a simple custom smart contract with this Sawtooth network environnement. Interaction with the smart contract should be, for example, a simulation of a car changing its owner, driving (i.e. record the mileage) etc. For this, we would need to build a transaction processor following a set of rules describing what a car can record on the blockchain.
By building a custom made transaction processor, the Sawtooth network would have the perfect smart contract that we want for the SIM project. Interaction and implementation of such smart contract would lead to a big step forward to start IOT-blockchain application.

2. Alternative approach: Not all nodes are validators

Sawtooth has an in-build REST API that provides all the necessary commands to remote control the blockchain. We could create nodes (actually just an IOT with some basic requirement to build a transaction) that would only build and submit transactions to the blockchain.
The following configuration, would describe such an environment :



Alternative IOT-Sawtooth configuration

So in this configuration, we would have a private network completely separated from the public world. Each IOT device can access and use the blockchain network via RSA authentication. This is a private blockchain, so all devices would need to be registered in the network.

I started to develop my own program in C that builds and submits transactions directly to the REST API.

The program has 5 steps :

1. Creating Private and Public Keys
2. Encoding Your Payload
3. Building the Transaction
4. Building the Batch
5. Submitting Batches to the Validator

I got to understand exactly how keys are created, and how transaction are built. I used the secp256k1 library to use standard cryptographic validation and key creation.[28] [29] I could create/read the files containing the hexadecimal values of the random bytes. Also, I started building the transaction payload. Unfortunately, I didn't finish this program in time. Anyhow, I did have a great learning experience by starting this program.

VII - Conclusion

I had a great experience during this tutorship. I gained a considerable amount of knowledge about blockchain technology.
Personally, I think blockchain can be implemented on IOT devices, but distributed ledger frameworks need to make progress to help the IOT integration (maybe lower-level programs).

Hyperledger Sawtooth is a promising distributed ledger for the SIM project. We still need to build a custom smart contracts and deploy it on a real IOT. After such investigation, we could really deploy a simulation for the SIM project using the right permissions configuration of Hyperledger Sawtooth. This would lead to a great example of IOT blockchain implementation with smart contract. By using sensors and building a basic web interface using the REST API, we could display summarized and condensed data to logged-in users.

Bibliography

[1] LEAT: http://leat.unice.fr/

[2] Smart IOT for mobility project website: http://univ-cotedazur.fr/en/idex/academies/networks-information-and-digital-society/recherche/liste-des-projets-finances/smart-iot-for-mobility

[3] Mr. Massiera Nicolas, Internship Report Project Smart IoT for Mobility, 2018

[4] Blockchain an introduction by A. Shanti Bruyn : https://beta.vu.nl/nl/Images/werkstuk-bruyn_tcm235-862258.pdf

[5] About blockchain by Hyperledger : https://github.com/hyperledger/sawtooth-core/blob/master/docs/source/introduction.rst

[6] Bitcoin Cryptocurrency: https://bitcoin.org/bitcoin.pdf

[7] Blockchain Data Storage : https://lisk.io/academy/blockchain-basics/use-cases/blockchain-data-storage

[8] How Blockchain Technology Can Prevent Voter Fraud : https://www.investopedia.com/news/how-blockchain-technology-can-prevent-voter-fraud/

[9] The Idea of Smart Contracts : http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_idea.html

[10] Internet of Things-IOT (white paper) : http://ijesc.org/upload/8e9af2eca2e1119b895544fd60c3b857.Internet%20of%20Things-IOT%20Definition,%20Characteristics,%20Architecture,%20Enabling%20Technologies,%20Application%20&%20Future%20Challenges.pdf

[11] IOT definition : https://techterms.com/definition/internet_of_things

[12] Raspberry-pi specifications : https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/

[13] Smart IOT for mobility white paper : https://www.researchgate.net/publication/327860801_Smart_IoT_for_Mobility_Automating_of_Mobility_Value_Chain_through_the_Adoption_of_Smart_Contracts_within_IoT_Platforms

[14] Ethereum website: https://www.ethereum.org/

[15] Ethereum image source : https://news4c.com/ethereum-price-analysis-the-trend-against-usd-and-bitcoin/

[16] Hyperledger Sawtooth : https://sawtooth.hyperledger.org/

[17] Sawtooth project github : https://github.com/hyperledger/sawtooth-core

[18] Sawtooth-Ethereum smart contract integration : https://sawtooth.hyperledger.org/docs/seth/releases/latest/introduction.html

[19] Hyperledger-Ethereum comparaison : https://espeoblockchain.com/blog/hyperledger-vs-ethereum/

[20] Ethereum-sawtooth smart contact : https://github.com/hyperledger/sawtooth-core/pull/415

[21] Hyperledger Sawtooth documentation : https://sawtooth.hyperledger.org/docs/core/releases/latest/

[22] Contracts with hyperledger sawtooth : https://stackoverflow.com/questions/51375195/how-to-deploy-contracts-with-hyperledger-sawtooth

[23] Hyperledger Sawtooth overview : https://medium.com/remme/hyperledger-sawtooth-as-a-development-framework-architecture-overview-4947ec81fa50
[24] Docker company inforamtion: https://www.docker.com/company
[25] New version 1.1 of Sawtooth : https://www.hyperledger.org/blog/2018/12/06/announcing-hyperledger-sawtooth-1-1
[26] Sawtooth network : https://sawtooth.hyperledger.org/docs/core/releases/latest/app_developers_guide/creating_sawtooth_network.html
[27] Docker network information: https://docs.docker.com/v17.09/engine/userguide/networking/#bridge-networks
[28] scep256k1 github: https://github.com/bitcoin-core/secp256k1
[29] screp256k1 documentation : https://godoc.org/github.com/ethereum/go-ethereum/crypto/secp256k1
[30] Sawtooth consensus https://sawtooth.hyperledger.org/faq/consensus/