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

Part 2

Author: Mr. Luc Gerrits

Tutor: Mr. François Verdier

Supervisor: Mr. Roland Kromes

Science Faculty - Electronics
University Côte d'Azur
France

June 2019

Table of contents

1 Preface and acknowledgement

1.1 The work space

For my first year of the masters degree I did a tutorship at LEAT[5] (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.[26]
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.

Figure 1: Logo of the LEAT and the University Côte D'azur

1.2 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 same year of the electronics masters degree and is in charge of making the link between the electronic and legal challenges of the project.

2 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[19] helped a lot to understand the main concepts of blockchain and the different solutions provided by different companies.

2.1 Definitions

A blockchain[7] is a decentralized, distributed database that is used to maintain a continuously growing list of records, called blocks. Each block are linked (with cryptographic hash) to the previous block. It's also called a distributed ledger. Three main features[1] 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[18], cloud storage[2], preventing voter fraud[4] 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.
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.[20]
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.[21][16]
With those architectural challenges the goal is to implement blockchain access with smart contracts within these IOTs.[22]

We concluded in the first part of this tutorship that only two solutions are possible to implement Hyperledger Sawtooth framework into IOTs.[15] In practice the implementation of this blockchain on IOT devices is new and a lot of tools, programs and libraries needed to be made or needed to be adapted for our use case.

By starting trying building and compiling Sawtooth on the Raspberry Pi 3 we understood immediately that we would need research on processor architecture and operating system (OS) between and the IOT and usual computers. Also because Sawtooth project is young, tools like a data visualisation would be necessary. There is indeed already a project made by Hyperledger to navigate inside Sawtooth blockchain but it misses lots of crucial features. Thus it is required for us to start by solving his issues.

3 Custom tools for Sawtooth

In order to begin practical work on the Raspberry Pie we needed to visualise, search and communicate with Sawtooth. Some of these features already exist but is poor in terms of visualisation and code flexibility. Multiple GIT projects were made to be able to share code between colleagues, which also provide excellent traceability.

3.1 Sawtooth Blockchain Web-Viewer

Sawtooth Explorer[11] is an application that provides visibility into the Sawtooth Blockchain for Node Operators (via a web browser user interface). This project is meant to help scroll throw the Sawtooth blockchain but is missing key-features to be able to correctly study with the blockchain data. One of the issue is a search feature allowing us to find (by querying) blocks, batches and transactions based on deserialize payloads content. Sawtooth Explorer second issue is the absence of deserialization of payloads: Human readable data is essential to view the transaction payloads.
To overcome these issues we decided to make a web based interface with a local copy of the blockchain database (SQLITE file). The code is written in Node JS, with EJS middleware for HTML templating (the client side uses Bootstrap HTML/CSS/JS framework). The database would be filled by scraping Sawtooth's REST API and piping the data to a deserializer depending on the transaction family. This way it will be possible to query for any data inside the blockchain. The query could be a baches IDs to transactions payloads (JSON of even just regex text).

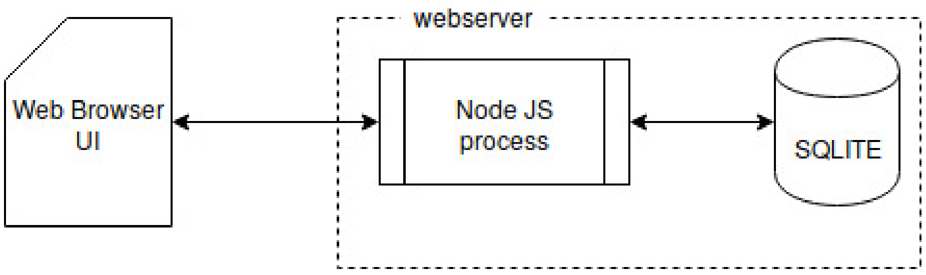
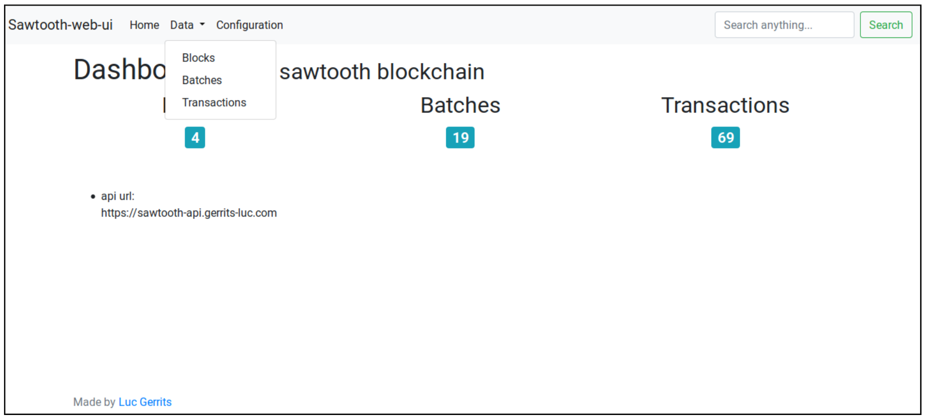


Figure 2: Sawtooth Blockchain Web-Viewer project flowchart

This is a GIT self-hosted project, tested and currently running. We can now search and display all information about the Sawtooth blockchain.
Some disadvantages we can encounter are:

- It uses a local copy of the database, meaning it is not a decentralized application.
- All data must be deserialized (e.g. CBOR to unicode/utf-8), parsed (stringified JSON to JSON object) and formatted (e.g. JSON indentation for readability) during the importation process. This can take a huge amount of time for a first importation depending on the blockchain size.
- The project needs a password protected proxy pass or an authentication system needs to be implemented to the Node JS server code.



Dashbo sawtooth blockchain

	Batches	Transactions
4	19	69

- api url: https://sawtooth-api.gerrits-luc.com

Made by Luc Gerrits

Figure 3: Sawtooth Blockchain Web-Viewer browser screenshot

Some of the features that could be added is live data importation. It is possible to implement an event listener for almost any data being processed by the Sawtooth blockchain network. As Sawtooth call it, this is done by “subscribing to events“ via a web socket (or using ZMQ protocols [10]). The article of Qigang

Liu and Xiangyang Sun explain perfectly that web socket can be used for real time application like Sawtooth's event listener.[14] Unfortunately Sawtooth's documentation states that it is recommended to use ZMQ because web socket connection to the REST API is limited in terms of supported retrievable data. ZMQ being an alternative to web socket it wouldn't be a problem.

3.2 Sawtooth C++ transaction

We also need a software program to test a real use case scenario. We will take the example used in the SIM report made by Mr. Sylla Sakoba, his use case is a car, more precisely a car a collision. When the car detects a collision, a batch is build by the IOT device inside the car and sends the transactions containing the car information. The transaction builder would be probably coded in python or a C++ program.
This scenario is also used by an article on distributed solution for automotive security and privacy. The author has a table comparing conventional methods and a blockchain implementation showing its advantages for vehicles and related services.[8]
Python is a hight-level general-purpose programming language (scripting language). So even if Sawtooth has an SDK for this language we will not use it because it doesn't allow us to compile the code for specific devices. In the other way C++, who is also a general-purpose programming language, can be implemented in a lot more ways then Python. It can compile the code for ARM, AMD64 and other processors architectures, and could be deployed on a multitude of IOT devices.[9]
For Python it is required to have an OS (like linux) installed on the IOT and for C++ if the code is compiled properly it won't necessary need an OS. Thus the reason to choose C++.

The features our program would have:

- Build secure transactions, send them to one of the Sawtooth REST API.
- Uses as much as possible native C/C++ libraries: so it can be compiled on almost any platform and architecture.
- Be tested on raspberry pi 3: that has a ARMhf processor architecture.

More on this program will be described in section 5.

4 Docker validator inside the raspberry

4.1 The goal and theory

For this a first attempted of implementing a blockchain technology on IOT device we choose to have a complete Sawtooth validator node inside this devices. This means the IOT device should be able to handle:

- Private-public key generation and signature verification
- Hashing (Sawtooth needs the lastest SHA256)
- Execution of a the validator program
- Execution of one or mutltiple Sawtooth transaction processors

It means the IOT should have an operating system compatible with the Sawtooth programs.

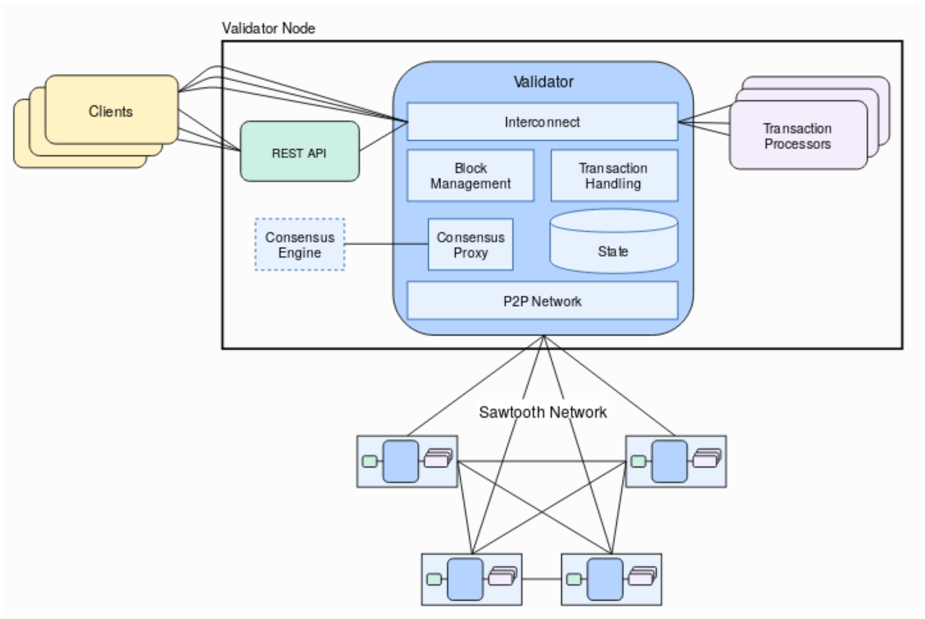


Figure 4: Sawtooth architecture

As you can see on Sawtooth architecture graph (Figure 4), the validator contains almost all the required software to build a node for the blockchain network. This software is specifically compiled for certain architectures using Docker (OS level vitalization).
The transaction processors (TP) handles specific transaction families mostly optional (settings TP is required). An analogy of this component is the Ethereum smart contracts.
Among TPs an consensus engine (using Proof Of Elapsed Time) is required and communicates with the validator.

The figure 5 show how we intent to set up our hardware and software.

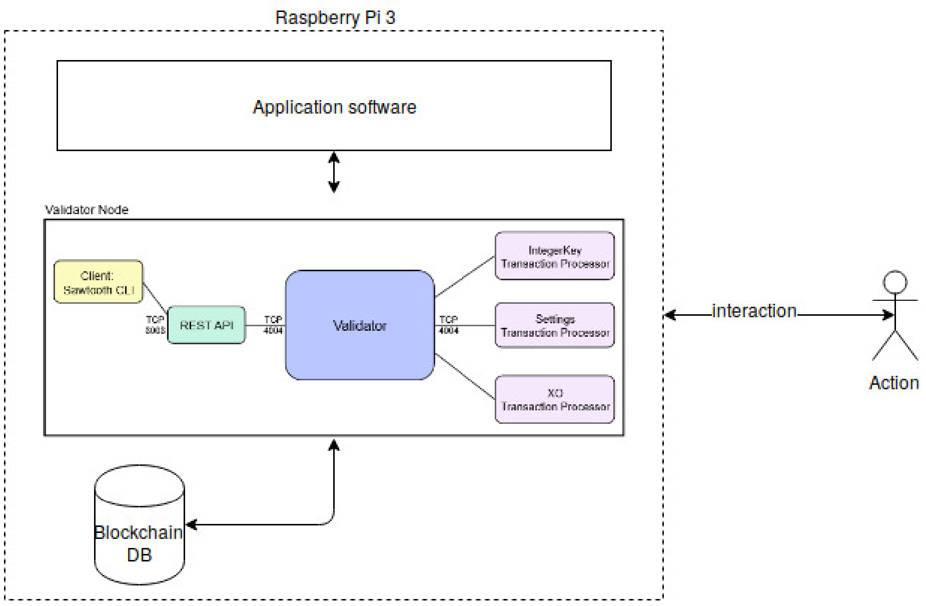


Figure 5: Communication diagram between a client, IOT and Sawtooth network by having a complete Sawtooth validator node inside the IOT device.

This kind of implementation will make all IOT devices part of the blockchain. A multitude of advantages and disadvantages will result of this setup.
Main advantages would be that all IOT devices will be able to participate in the blockchain and that the transaction processing will be distributed over all IOTs. Meaning less centralized data centers, helps maintaining the blockchain decentralized and would result in faster transaction validation.

4.2 Practical results

By using a naive approach, we have installed Docker operating-system-level virtualization on the Raspberry Pie 3 and started the same Docker images as on desktop computers. This naturally won't work due to Docker images compilation environment mismatch. The images are not designed to be running on ARM processors.
It is possible to recompile the whole validator image but a lot of dependencies are also pre-compiled software meaning without a proper build documentation it won't be possible. We have also built a complete new docker image by hand and by following all the specification in the documentation but the main issue was still that sub-dependencies of python are not build for the IOT environment. This was confirmed during an exchange on the official Hyperledger chat we had with Dan Anderson who works on Hyperledger Sawtooth at Intel. He is not aware of somebody who implemented this software on a Raspberry Pie device, worldwide.

M. Samaniego and R. Deters article is also tends to point into this conclusion. The paper is more about the challenges that IOT devices are often too constrained regarding computational resources and available bandwidth, meaning even if the Raspberry Pie 3 can run the software, we will have issues regarding processing power and network bandwith. [24]

As a result this solution is not a good fit for this type of project. Blockchain implementation on IOT devices such as an Raspberry Pi 3 using Docker doesn't work at the moment.

5 Only the essential to send transactions

5.1 The goal

Because of the previous issues (section 4) the second attempted of implementing a blockchain technology on IOT device is to build software (in C++) that is capable to send transactions to the Sawtooth network without being a validator node. This means the IOT device should be able to handle:

- Private-public key generation and signature verification
- Hashing (Sawtooth needs SHA256 hashes)
- Sending transaction using HTTP with TLS/SSL

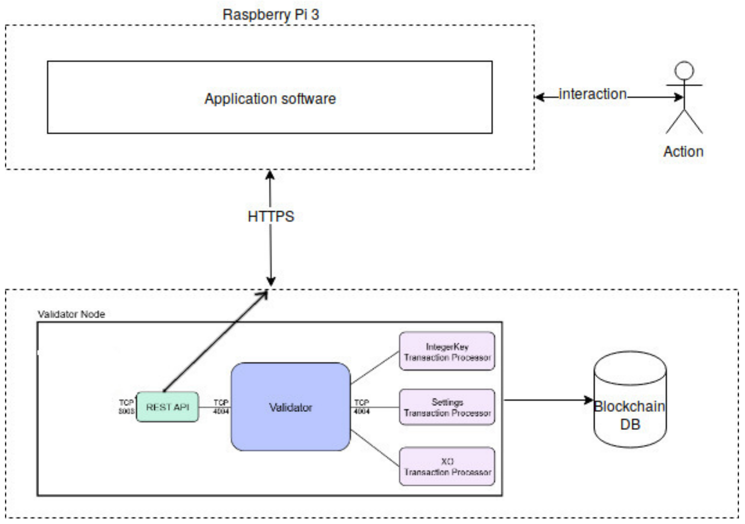


Figure 6: Communication diagram between a client, IOT and Sawtooth network by having only a transaction handler inside the IOT device.

Figure 6 explains that our IOT device will be communicating with validators nodes throw secure HTTP requests (using the REST API). As an example:

If a the IOT device detects a crash (or is instructed so) the device will start building transactions with payloads containing messages/data we want to send. After regrouping all transactions into one batch, the IOT will post the Protobuf formatted data to the Sawtooth REST API.

5.2 The program

Thanks to version 1.1 we are able to build a proper Sawtooth network using PoET consensus. This network was already build in the first part of this tutorship. We used a network with a minimum of 5 nodes created in the first part of this tutorship.
There isn't a C++ Sawtooth SDK implementation. For this reason we used multiple standard open source (or free) libraries to build one project that can handle the required task to send transactions to Sawtooth network.
The program we made is using only stable C++ libraries that have multiple architecture compatibility. With more development the project could come to an stable version. We needed to use multiple components stacked together to archive to a working program, e.g. protobuf, cryptopp, sha and curl.

Protobuf is developed by Google with an open-source licence. Protobuf buffers are a language-neutral, platform-neutral extensible mechanism for serializing structured data. This means data transmission between two parties can be performed regardless the programming language or the communication method. Numerous of studies showed that IOT communication has great performance using Protobuf formatting (compared to XML communications [25]).

During the conception of the program we needed to understand cryptographic algorithm (key and signature formatting was really time consuming).
We used cryptopp (also called Crypto++) a free C++ class library of cryptographic schemes[3]. Crypto++ Library is copyrighted as a compilation and licensed under the Boost Software License 1.0 but all individual files in the compilation are all public domain. Using in our case a Linux based operating system, Cryptopp is pre-compiled for all major system architecture, meaning it's installation is unlikely to be an issue. It allows us to build all the necessary public key (64bytes compressed), private keys (32bytes), signatures (64bytes) using the (required by Sawtooth[13]) Elliptic Curve Digital Signature Algorithms (in our case one named secp256k1).[18]
Cryptopp's ECDCA digital signatures schema follows the US Federal Information Processing Standards and is approved by ANSI[6], Brainpool[17], and NIST[23].

To be able to use Ethereum smart contract we would need to implement a Solidity compiler. An existing Sawtooth transaction processor named "seth" is allowing this compatibility.[12] Solidity is a statically typed, contract-oriented, high-level language for implementing smart contracts on the Ethereum platform (so also on Sawtooth).

5.3 Practical results

Important note : When compilation is stated as working (i.e the program compiled), it means the compilation has always been tested on AMD64 processor architecture (e.g. desktop computer) and on ARMhf architecture (in our case a Raspberry Pie 3).
The program is capable of building any kind of transaction(s) and combine them into one batch following the cryptographic constrains defined in section 5.2. The program correctly compiles and executes.
Nonetheless some major difficulties (but solved) where encountered:

- Public key generation with the correct format
- Encoding data (payloads and signatures)
- Sending transactions using libcurl library

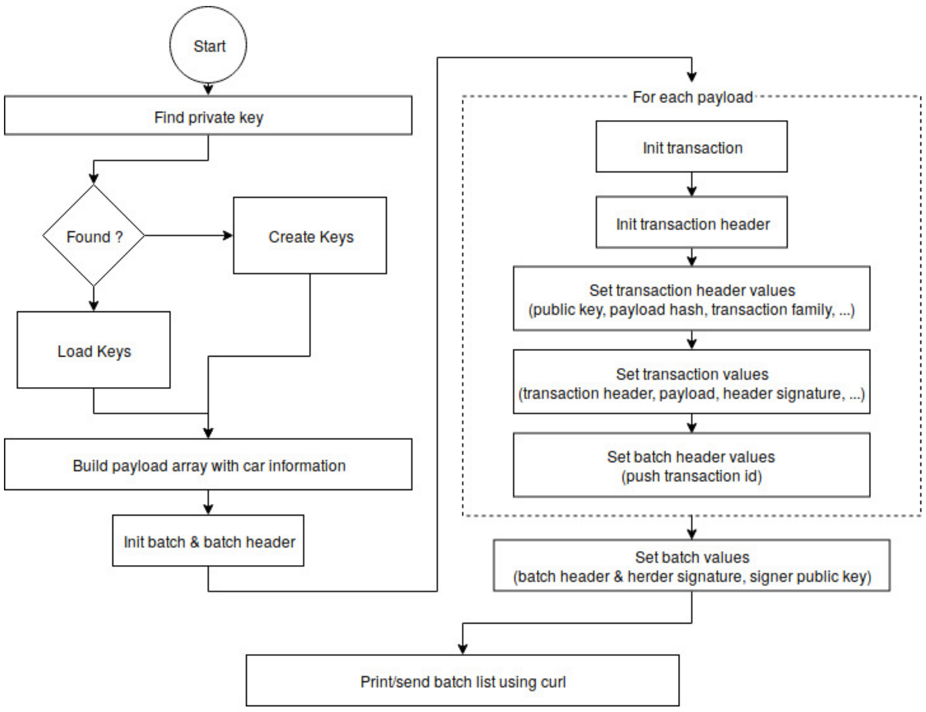

Start
Find private key
Found ?
Create Keys
Load Keys
Build payload array with car information
Init batch & batch header
For each payload
Init transaction
Init transaction header
Set transaction header values
(public key, payload hash, transaction family, ...)
Set transaction values
(transaction header, payload, header signature, ...)
Set batch header values
(push transaction id)
Set batch values
(batch header & herder signature, signer public key)
Print/send batch list using curl

Figure 7: Program flowchart.

An important information you can't see on figure 7 is that multiple data values are serialized before being set. All you see on this flowchart is coded and functional.
Unfortunately Sawtooth REST API response is that the batch list is poorly formatted or that a signature isn't valid. There are only two signatures in a batch

list (containing one transaction). These two signatures have the correct format but it is possible there is an issue during a manipulation of the data (encode, de-serialize and/or re-serialize). It is an error that I would need to discuss with the project maintainers.
After this issue is solved, real application with a car (or car simulation) could be done.
To better understand the impact of a C++ program implementation for Sawtooth, it will be essential to make benchmarks. This will allow us to see practical results about power consumption, transaction speed, network usage and computing time.

6 Conclusion

To conclude this tutoring, we have seen this semester that we had a lot more practical work by applying the ideas we have presented in the previous part of our work. We are now sure that the first solution (presented in section 5) is not going to work at the moment, but the second solution is. If we do some deeper research on the Docker images and the validator's software dependencies, it will work (hardware like disk space, ram, etc. needs to be considered).
Sawtooth has a good potential for this blockchain and IOTs implementation with the second solution we provided (section 5). The program is compact and can be deployed on a lot environments.
The biggest compromise would be that IOT devices won't entirely be part on the blockchain. Therefore, a decision will have to be made as to when the IOT should or should not have the technology of validating transactions.

References

[1] About blockchain by Hyperledger. https://github.com/hyperledger/sawtooth-core/blob/master/docs/source/introduction.rst. Accessed: 2019.

[2] Blockchain data storage. https://lisk.io/academy/blockchain-basics/use-cases/blockchain-data-storage. Accessed: 2019.

[3] Crypto++® library 8.2. https://www.cryptopp.com/. Accessed: 2019.

[4] How Blockchain Technology Can Prevent Voter Fraud. https://www.investopedia.com/news/how-blockchain-technology-can-prevent-voter-fraud/. Accessed: 2019.

[5] LEAT Website. http://leat.unice.fr/. Accessed: 2019.

[6] Inc. Financial Industry Standards Accredited Standards Committee. Public key cryptography for the financial services industry - key agreement and key transport using elliptic curve cryptography. 2011.

[7] A. Shanti Bruyn. Blockchain an introduction. August 26, 2017.

[8] A. Dorri, M. Steger, S. S. Kanhere, and R. Jurdak. Blockchain: A distributed solution to automotive security and privacy. *IEEE Communications Magazine*, 55(12):119–125, Dec 2017.

[9] James O Hamblen, Zachery C Smith, and Winne W Woo. Introducing embedded systems in the first c/c++ programming class. In *2013 IEEE International Conference on Microelectronic Systems Education (MSE)*, pages 1–4. IEEE, 2013.

[10] Pieter Hintjens. *ZeroMQ: messaging for many applications.* " O'Reilly Media, Inc.", 2013.

[11] Sawtooth Hyperledger Intel. Hyperledger sawtooth - Explorer Git Project. 2019.

[12] Sawtooth Hyperledger Intel. Hyperledger sawtooth - Seth Git Project. 2019.

[13] Sawtooth Hyperledger Intel. *Sawtooth Creating Private and Public Keys.* Hyperledger, 2019.

[14] Qigang Liu and Xiangyang Sun. Research of web real-time communication based on web socket. *International Journal of Communications, Network and System Sciences*, 5(12):797, 2012.

[15] Mr. Gerrits Luc. Tutoring Internship Report Project: Smart IOT for Mobility, Technical challenges. 2019.

[16] HK Merchant and DD Ahire. Industrial automation using iot with raspberry pi. *International Journal of Computer Applications*, 168(1), 2017.

[17] Johannes Merkle and Manfred Lochter. Elliptic curve cryptography (ecc) brainpool standard curves and curve generation. 2010.

[18] Satoshi Nakamoto. A peer-to-peer electronic cash system. 2008.

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

[20] Keyur K Patel, Sunil M Patel, et al. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing*, 6(5), 2016.

[21] Dinkar R Patnaik Patnaikuni. A comparative study of arduino, raspberry pi and esp8266 as iot development board. *International Journal of Advanced Research in Computer Science*, 8(5), 2017.

[22] Guitton Patricia, François Verdier, primavera de filippi, Thierry Marteu, Frédéric Mallet, Philippe Collet, Lise Arena, Amel Attour, Marta Ballatore, Michela Chessa, Agnès Festré, Raphael Bernhard, and Benoit Miramond. Smart iot for mobility: Automating of mobility value chain through the adoption of smart contracts within iot platforms. 09 2018.

[23] Federal Information Processing Standards Publication. *FIPS PUB 186-4 Digital Signature Standard*. U.S. Department of Commerce, 2013.

[24] M. Samaniego and R. Deters. Blockchain as a service for iot. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 433–436, Dec 2016.

[25] P. Urien. *Towards secure elements for trusted transactions in blockchain and blochchain IoT (BIoT) Platforms. Invited paper.* Feb 2018.

[26] F. Verdier. *Research Project Smart IOT for Mobility.* University Côte D'azur, 2019.