A True Decentralized Implementation Based on IoT and Blockchain: a Vehicle Accident Use Case

Luc Gerrits
Université Côte d'Azur
LEAT / CNRS UMR 7248
Sophia Antipolis, France
luc.gerrits@etu.univ-cotedazur.fr

Roland Kromes
Université Côte d'Azur
LEAT / CNRS UMR 7248
Sophia Antipolis, France
roland.kromes@univ-cotedazur.fr

François Verdier
Université Côte d'Azur
LEAT / CNRS UMR 7248
Sophia Antipolis, France
francois.verdier@univ-cotedazur.fr

***Abstract*—In this paper, we will present a new model of distributed ledger-based IoT network, in which we combined Hyperledger Sawtooth as blockchain with InterPlanetary File System (IPFS) as a distributed storage system. The combination of these two types of distributed ledger technologies can allow more efficient data storage than in other blockchain implementations. This work was initiated by the automotive manufacturer Renault based on the idea of a new ecosystem of smart vehicles containing IoT devices. We will focus on an accident use case. After the accident, the cars send their data to a dedicated smart contract. We will also describe furthermore our implementation, the characteristics of Hyperledger Sawtooth and IPFS and finally, we demonstrate the realistic feasibility of this implementation by latency measurements.**

***Index Terms*—Blockchain, IoT, Embedded Systems, Hyperledger Sawtooth, Smart Contracts, Latency, IPFS**

I. Introduction

The first blockchain (Bitcoin) was based on the paper written by Satoshi Nakamoto in 2008 [1]. The principal usage of this technology was the creation of an environment to manage cryptocurrencies without having a centralized authority. The concept of blockchain is to have a data structure of growing, immutable, protected and decentralized records.

Since the creation of Ethereum blockchain [2] the execution of smart contracts (i.e. business logic in a form of code) is possible within a blockchain. Using smart contracts gives a possibility to create new – or modify existing – systems to make them autonomous and safe in terms of data privacy.

One of the main reasons to use blockchain and not a standard database is to obtain a distributed record that cannot be altered with. The advantage of using blockchains in this context is to track and share data securely and in complete confidence, to avoid that a centralized and proprietary database of a provider can be modified in all "privacy". So, to counter this, we want a decentralized database shared between partners who are self-monitoring. This is conducive to the generation of an ecosystem and will be essential to determine the responsibility in the case of an accident of an autonomous car (e.g. to investigate driving behaviour).

Today we can find multiple blockchain use cases in various domains like healthcare [3], real estate [4], Smart City and also Smart vehicles. Some of these applications involve Internet of Things (IoT) devices.

IoT is a network of devices (i.e. everyday objects) that contains electronics, programs and connectivity allowing these devices to connect, interact and exchange data. In several cases, the IoT devices are very constrained architectures meaning that they have limited resources such as computational power, small memory size and even low battery lifetime.

This work has two primary aims. The first one is to implement blockchain technology within constrained IoT devices that will be able to interact with blockchain and smart contracts. Depending on the consensus, blockchain needs high computational power and its database size increases infinitely.

Our second aim is to find a more efficient data storage management in the blockchain. An idea of this management is to store only the references of raw data (basically the hashes of the data) on the blockchain. The raw data could be registered on an other distributed system in which the data is not necessarily duplicated in every peer of the network.

The global use-case that we have chosen is represented in Fig. 1 and represents a vehicle infrastructure. In this use-case, Renault's cars are connected to blockchains deployed on several clouds and are able to connect each time an accident occurs. We can observe that Renault's cloud is connected with other types of organizations like an insurance company, expertise, police and car mechanics.

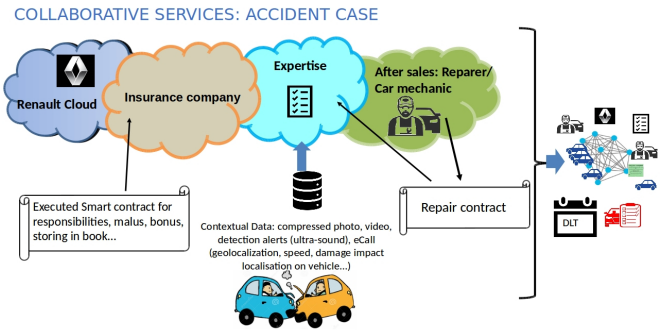


Fig. 1. Renault's use case: an accident occurs

The cars will contain IoT devices, and will be able to establish a connection with Renault's cloud technology based on a blockchain with smart contracts. The cars will contain

978-1-7281-6371-0/20/$31.00 ©2020 IEEE

sensors such as odometers (that measure the distance travelled), radars (that detect the safe-distance), 360° cameras and many other senors recording the car's environment and the driver behaviour. The data recorded before the accident would be sent to Renault's cloud using a smart contract. Once the data is received, Renault's cloud will process the transaction. Other companies connected to Renault's cloud could get access to the data and process it with their own blockchain and smart contracts. With the help of police and predefined juridically legal smart contracts, the insurance company could, for example, determine who was responsible for the accident situation.

This infrastructure contains a lot of different entities. The obtained structure is rather an ecosystem than a typical vehicle infrastructure. To simplify our research work in the first phase of this project we focus only on a possible solution for the implementation of realizing Renault's cloud.

In Sect. II we will briefly describe blockchain and smart contract technology, including examples of related works about blockchain IoT and vehicle infrastructure systems. Sect. III describes Hyperledger Sawtooth blockchain, IPFS and the structure of our deployed implementation. In Sect. IV we will examine the performance of our solution. We determine the transaction validation rate of the blockchain and the IoT device's time performances. We will finally give in Sect. V a rapid discussion and conclusion about our results.

II. State of the art

A. Blockchain and Smart Contracts

1) Blockchain: A blockchain is a distributed ledger of immutable records. It is protected by private-public cryptographic key signatures and hashes. The blockchain is a decentralized network with duplicated data between all nodes. This novel data structure is protected, if it is set up correctly, against attacks like data altering and identity usurpation [5].

The consensus is a key part of blockchains. It is an algorithm that provides an answer to the validity of a transaction. In our implementation, we have decided to use Hyperledger Sawtooth blockchain. The Proof of Elapsed Time (PoET) is the default consensus rule of Sawtooth. This consensus rule is designed to support large networks. Thus, this consensus can be an ideal solution for our implementation. PBFT (Practical Byzantine Fault Tolerance) is another consensus that is guaranteed safe if the number of faulty nodes is less then $\frac{n-1}{3}$. This consensus is intended to handle small to medium-sized networks. PBFT can also be a solution for our implementation. In Proof of Work consensus (PoW), the members participating in the transaction validation have to resolve a hard cryptographic problem. To resolve this problem the nodes need a high computation power. Thus this consensus is not suitable for IoT devices.

2) Smart Contracts: Smart contracts are programs that can be executed automatically when events happen in the blockchain. These contracts were imagined to replace traditional paper format contracts and be able to avoid the trusted third party confirming the validity of the contracts [6]. These smart contracts are digital programs that can be executed automatically on the blockchain. The smart contracts can realize several applications that are related to the blockchain.

B. Blockchain based vehicle infrastructures

To the best of our knowledge, there are two research projects proposing a new type of vehicle infrastructure based on the blockchain and smart contracts technologies that are similar to our project.

Brousmiche et al. [7] described a new vehicle ecosystem with the primary goal of digitizing the vehicle life-cycles in secure car Smart Service Books realized by the smart contracts. In this system authors use a consortium blockchain with Quorum on Ethereum business model. This infrastructure consists of an automotive manufacturer, an insurance company, the vehicle owners, the vehicle retailer and maintainers. Basically, a vehicle sends its mileage information that is stored directly on the blockchain. The authors also described a different type of storage for their implementation. This hybrid storage allows storing of consortium-based data on the blockchain while the other types of data (member specific business data) are stored locally. A car can also send other types of information as documents (e.g., receipt) that contain a pointer to an external cloud archiving service containing the pointed data. The authors also mention that in their work they do not focus on identity management to have access to the external data.

In the second study [8] authors proposed an infrastructure in which vehicles record information not only by their sensors (e.g., cameras, lidar, brakes, etc.) but also from other vehicles and Road Side Units (e.g. traffic lights). In this work only the hash of the data is sent to the blockchain. The recorded data is treated by different members (Law Enforcement, Insurance Companies, Car Manufacture, Maintenance Service Provider). One of the goals is to be able to determine the vulnerable parties of forensics situations. Every member has access to the hashes set on the blockchain. The raw data is constructed by different types of data (e.g. Maintenance data, Insurance Company Data) that is stored locally in the members. Every member contains its dedicated data.

The idea of storing only the hashes of data allows a minimal storage on the blockchain and minimizes the processing overhead. Thanks to these advantages we are also using this approach in our work. However, in the described work the raw data are stored locally in the members and the access control mechanism is not yet studied. In our implementation, we are proposing to store the raw data in a distributed file system (IPFS). This implementation has authentication for all IoT devices that wish to participate and store data in the ecosystem.

C. Blockchain based IoT solutions

Because each vehicle contains a connected IoT device the imagined vehicle infrastructure can be considered as a dynamic IoT network. In related works we can find examples of implementations of IoT networks using blockchain and smart contracts. The following works use fairly similar methods as

we also applied to establish and transmit data between the IoT nodes and the blockchain.

In the study [9], the deployed architecture controls the IoT permission to the blockchain. The IoT device can only send the transaction to an interface (called Management Hub, written on JavaScript) that is implemented on the top of the miner node (participating in the transaction validation process) of Ethereum.

In a similar work authors created a system where the IoT nodes can communicate with each other when they are part of the same cluster ("trusted bubble") [10]. The nodes use the blockchain as a communication channel. The communication access between the IoT nodes is controlled by the smart contracts. The end-nodes (Client, IoTs) communicate with the blockchain via a C++ interface that encodes/decodes data toward/from Ethereum. JSON Remote Procedure Calls are used to do these interactions.

To the best of our knowledge, Hyperledger Sawtooth blockchain was used for IoT solutions only by [11]. This implementation realizes the traceability of Agri-Food supply chain management. In this system sensors send data to a gateway or to a related PC that is connected on the blockchain.

We have decided to use Hyperledger Sawtooth because of its advertised modularity, its consensus, its rapidity and the ease of communication between IoT and blockchain.

III. Implementation with Hyperledger and IPFS

A. Hyperledger Sawtooth

Hyperledger Sawtooth is an enterprise blockchain platform, highly modular, designed to keep ledgers distributed and making smart contracts safe. This blockchain separates the core system holding the base of the platform and the application domain where developers can build their own business rules based on enterprise requirements. The core of the system is surrounded by modules that make it possible to choose enterprise-specific transaction rules, permissions and consensus algorithms.

The validator is the core of a Sawtooth node. It is responsible for validating batches of transactions according to the consensus rule that is used. Transaction Processors (TP) are modules containing the business logic (as it is the case in *smart contracts*) that will be applied to the transaction. The Consensus engine is required for the Validator, it is a component that provides consensus-specific functionality for a Sawtooth node. The REST API is a module, built by Sawtooth, communicating with the Validator that allows Client applications (in our implementation the IoTs) to exchange data with the blockchain.

1) Sawtooth message structure and signature process: To be able to understand the permissions and authorization of the system we have built, we will also need to describe the vocabulary and the structure of transactions in Sawtooth. The *transactions* are encapsulated in *batches*. This means a *batch* also has to sign the list of transaction IDs from which the batch is composed. A transaction can be signed with the same or another key than the *batch*.

As we mentioned in Sect.II an IoT device can establish a connection with one of the nodes of the blockchain via an application interface. Thus the device does not take fully part of the blockchain. The Validator node of Sawtooth already has a REST API allowing the communication with Client applications. In our work, we don't have to implement such an interface serving as a bridge between a Client API and the blockchain, as it was the case in some of the related work in Sect. II.

B. InterPlanetary File System (IPFS)

IPFS is a distributed file system based on a peer-to-peer network [12]. This system is similar to a BitTorrent file system in which the shared file can be seen and copied from every peer of the network. The privacy of such a network is based on block storage with content-addressed hyperlinks forming a Merkle Directed Acyclic Graph (DAG). When a file is added to the IPFS system it is hashed, thus the file is identified by its hash. Unlike a blockchain network, data in IPFS is not necessarily duplicated on each peer and does not contain a consensus process. In our implementation we have deployed a private IPFS network with static peers and with a specific swarm key allowing the communication between the peers (the peer who has not the given swarm key could not participate in the file system).

We have deployed a validation mechanism (described in sect. III-C) to be able to validate the provenance of the data and add the data on this file system. The data can be added if and only if the data was sent by the IoT that is a member of our ecosystem.

C. Overall network structure

In related work [13], authors combined blockchain with IPFS file system to be able to store IoT data. In this study, a smart contract was used to store smart devices public key and the hash of the IPFS file that stores the smart device's data. In this implementation, the authors used a public IPFS system. Moreover, in this implementation, the blockchain validator node is responsible for updating the data to the IPFS file system. This update process can overload the blockchain system and slow down the validation process.

However still forming a common system, in our implementation, the blockchain and the IPFS are two separate networks. By this separation, the blockchain and the IPFS service can run in a completely different network infrastructure. In addition to the architecture separation, we used a private IPFS network to increase the privacy.

Our implementation is depicted in the Fig. 2. As we mentioned above, this architecture is based on an Hyperledger Sawtooth blockchain and on an IPFS network. In every Validator nodes of the blockchain, we added an *IPFS module* that is connected to the Transaction Processor (TP) via a Python socket.

This module forwards messages to the IPFS network. However, every node of IPFS contains an application layer that we have called *IPFS REST API*. This module receives the

messages simultaneously from the blockchain and from the IoT devices (the functioning of this API is described in ***The protocol of the network*** sub-section). To consider a high scaled network structure a *Distributor* unit helps to forward massages coming from the blockchain to the IPFS network. The *Load Balancer* units serves to forward the IoT device messages to the IPFS and blockchain nodes facilitating communication in the overall network.

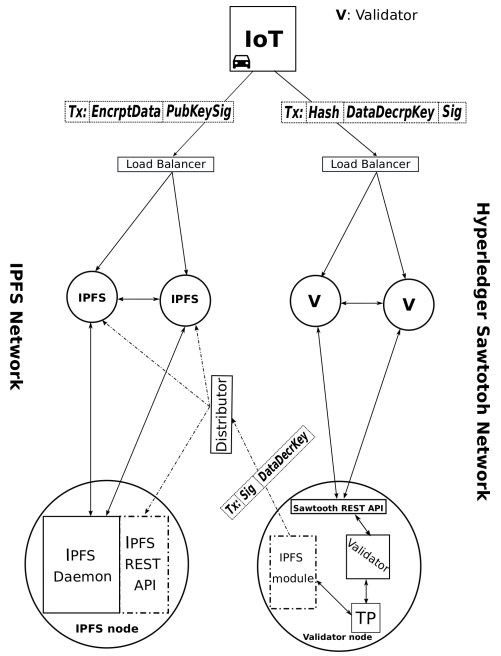
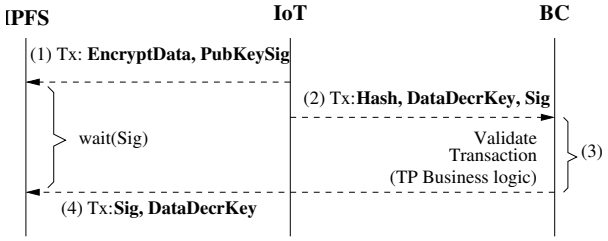


Fig. 2. Implementation containing Hyperledger Sawtooth and IPFS.

1) The digital signature procedure in cryptography: in our study, we are using the public key as a unique ID to identify access and origin of data. The ***private key*** is required to create a digital signature. In private-public key digital signature procedures, the data owner signs data with their private key. After sending the data on the receiver side it is possible to prove the validity of the data signed by the owner associated with the public key. The public key and the signature are used to verify the data (message). This is called the verification process with the public key.

2) The protocol of the network: it is depicted in Fig.3. Before the first step **(1)** the *IoT* generates a private/public key pair. By convenience the public key, called *PubKeySig*, is simultaneously used as the message and as the public key to create the signature. This allows not sending additional data in a request. With the private key it signs the *PubKeySig* to obtain a digital signature called *Sig* (*Sig* = sign(*PrivKeySig*, *PubKeySig*, *PubKeySig*)). The IoT application is written in C++ language, and for the key generation and signature process we have used the ECDSA SECP256K1 algorithm. The signature is required because, thanks to it, the IPFS network can authenticate that the IoT device (sending the data) is a member of the ecosystem.



(5) if verif(Sig, PubKeySig, PublicKeySig) == PublicKeySig than (6) else (7)
(6) decrypt(EncryptData) with DataDecrKey and add Data to IPFS
(7) reject(EcryptData)

Fig. 3. Message sending protocol in our implementation.

With a secret key *DataDecrKey* the data that would be stored on the IPFS file system could be encrypted (e.g. AES encryption, but it is out of scope for our implementation). The encrypted data is called *EncryptData*. In the first step **(1)** *EncryptData* and *PubKeySig* are sent to the IPFS network. This data is received by the *IPFS REST API*, and it remains in a waiting mode. Thus the data is not stored on the IPFS file system yet. Before the second phase **(2)** the device creates the hash of the data that was sent to the IPFS network. Before storing data on IPFS, it has to be formatted as a UnixFS data format. To do so, we have to use UnixFS protobuffer. However, with the standard open-source protbuf we cannot reproduce the required UnixFS data format. In our implementation, we had to modify UnixFS protobuffer to obtain the required format. After formatting the data the IoT device creates the same hash of the data (Hash) as the IPFS system before adding the file. In the second phase **(2)** the IoT sends the Hash the secret key for data decryption (*DataDecrKey*) and the digital signature *(Sig)* to the blockchain. In phase **(3)** a transaction validation process is launched. The validation rules are based on the business logic that is described by the transaction processor (This rule is explained in Sect.III-D). If the transaction is valid, it is stored on the blockchain. The Transaction Processor forwards the *Sig* and the *DataDecrKey*) to the *IPFS module*, that sends this data to the IPFS network. In phase **(4)** the *IPFS REST API* was still in waiting. After receiving the signature and the secret key the signature verification process starts. The data cannot be stored on the file system until the IPFS system is not sure that the data was sent by the trusted IoT. By the signature verification this condition can succeed. If the signature (*Sig*) is verified with the public key (*PubKeySig*) the encrypted data (*EncryptData*) that is referenced by *PubKeySig* can be decrypted with the *DataDecrKey*). After the decryption the raw data can be added to the IPFS file system. The *IPFS Daemon* adds the file.

D. Business logic and emitter authentication

The transaction processor deployed on Python programming language for our use case is called *cartp*. This TP contains business logic that controls but also secures the blockchain state. As shown in the Fig. 4, depending on the transaction command (new_car, new_owner or crash), the transaction will

need different types of permission. Our business logic includes layers of permission: a factory is allowed to create cars, a car can change owner and a car can create a crash with the drivers information. Thus each actor has their own public-private key pairs. The *cartp* TP will modify states that contain different information about a car owners/drivers and crashes. These states have addresses that can only be accessed by the right owner and with the correct permission that is described in Fig. 4.

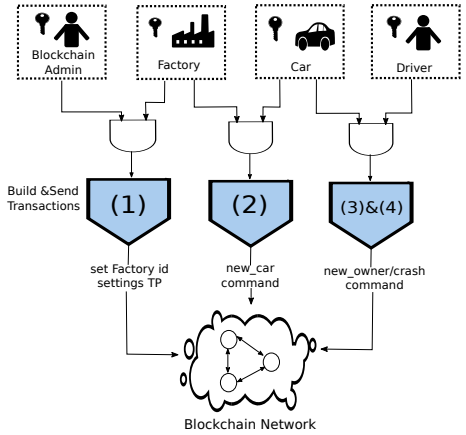


Fig. 4. Car TP (Transaction Processor) business logic.

When a crash occurs, the vehicle will automatically send a transaction. This transaction payload will contain the Hash, *DataDecrKey*, *Sig* and the current owner/driver public key. This information will identify the driver on the blockchain. The transaction will be signed by the driver/owner private key and the transaction batch will be signed with the vehicle's private key.

IV. Experiments and Results

In our experiment, we have measured the transaction validation rate of Hyperledger Sawtooth blockchain with and without the *IPFS module* that we have added in our implementation (see section III-C). These measurements show the impact of this module that serves as a bridge between the blockchain and the IPFS network. Thus, it plays a key role in the IoT authentication process.

On the other hand, we are measuring the execution time of the application that we have implemented in the IoT device. As we mentioned before, the IoT devices are constrained devices with low battery lifetime. Thus, the execution time needs be as short as possible.

A. Validation rate of the blockchain

The test was realized on an Intel Xeon CPU D-1528 @ 1.90GHz server, with 12 CPU and 128.8 GB RAM. The realized blockchain network contains 5 Validator nodes, running in Docker containers and configured with docker-compose. We used Hyperledger Sawtooth version 1.2 for the validator and other modules.

The tests consist in sending 3tx/sec with a limit of 10k transactions altogether. Each transaction is encapsulated in one batch (i.e. one REST API request to the blockchain). We aim to analyze the behavior of the blockchain under a multitude of clients. Thus sending one REST API request with only one transaction. In this paper, a transaction is considered successful when it is committed on the blockchain, meaning that the transaction rate is calculated as the number of successful transaction divided by the elapsed time. Transactions with a status different from committed will be considered as rejected.

Firstly we experimented Hyperledger Sawtooth using our TP without IPFS. We started the experiments with **PoET consensus**. It is possible to configure the consensus. Each parameter can be optimized to achieve a faster network and adapt its network size. Nonetheless, our experiments on Hyperledger Sawtooth using PoET consensus appear to never gain a higher transaction validation speed of 1.5tx/sec with a network size of 5 nodes. Also this consensus seems to stop working after 2.5k transactions, after that reject rate is almost 100%. In the studies of Ampel *et al.* [14] and Benahmed *et al.* [15], their experiments have the same outcome. The PoET consensus project is probably not developed enough at this time.

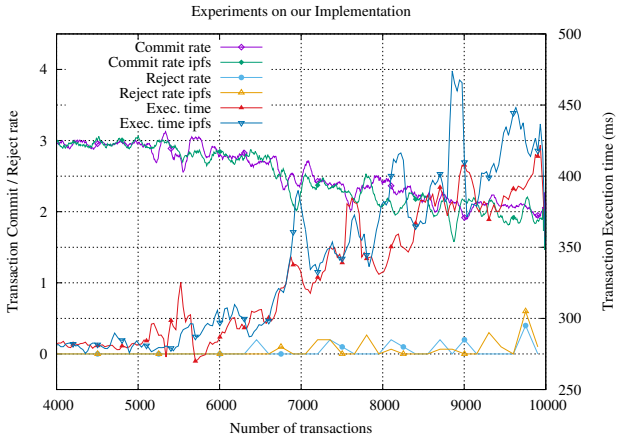


Fig. 5. Experimentations on our implementation using PBFT consensus.

These issues lead us to use Hyperledger Sawtooth **PBFT consensus**. With 5 nodes, this consensus works without difficulty at a rate of 2.7tx/sec. The PBFT algorithm is deterministic compared to PoET which is more probabilistic. With more than 10k transactions in this small blockchain network (5 nodes), PBFT consensus test on Sawtooth has significant performances. With 92% success on 10k transactions and 100% up to 6k transactions (low pass filtered results in Fig.5).

Secondly, in the next measurements, we observe the influence on the transaction validation rate and the number of rejected transactions of our IPFS implementation (the *IPFS_module* is installed). The result of this implementation is an increase in transaction execution time of approximately 10-100ms. The transaction rate is still at 2.7tx/sec.

B. Latency in the IoT device

In the experiments, we used a Raspberry Pi 3B+ model. It should be noted that the latency measurement of the communication phase via HTPP is not taken into account in our experiments because it depends on the given communication protocol. The results of the latency measurements are depicted in Fig. 6.

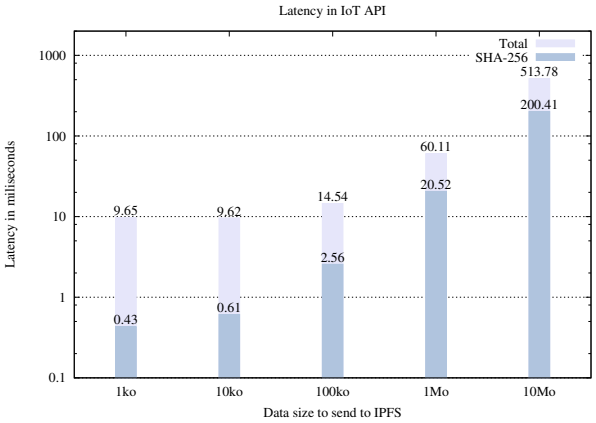


Fig. 6. Latency in IoT device. This figure represents the total execution time and the time that is occupied by the hash creation.

To reproduce the reference of the data that will be stored on the IPFS system the IoT device has to run the SHA-256 cryptographic hash function over the whole data. The latency of this procedure increases significantly when the size of the data grows. For example, in the case of 1Mbytes data, the hash reproduction procedure takes 34.1% of the total execution time.

The SHA function of the standard CryptoPP C++ library that was executed is already optimized for ARM processors (the Raspberry Pi contains a quad-core ARM Cortex-A53). Thus the hash reproduction time could not be reduced by an algorithm optimization. However, it is possible to use a hardware accelerator computing the SHA-256 hash function.

In a previous work, we used hardware accelerators computing the SHA hash functions to obtain a significant speedup (x112.75) of its application [16]. In this work, we modified a BCM2837 (the heart of the Raspberry Pi) by adding these accelerators. In this work, we could use the same approach to decrease the execution time, and it can also be possible to reduce the dynamic power consumption of the entire architecture.

V. Discussion and Conclusion

In this paper, we have realized an implementation of a vehicle infrastructure by using Hyperledger Sawtooth blockchain, IPFS file system, and IoT devices. It also has an IoT authentication protocol, which allows transactions to be rejected throughout the system. We are giving technological results and limits relying on the blockchain. By using the IPFS system we can also optimize the amount of data that the blockchain has to record. Contrary to [8], the data in our work remains completely distributed. We also give a possible solution to speed up the execution time of the IoT API and reduce the device's power consumption. To conclude this study, our implementation is operational using Hyperledger Sawtooth with PBFT consensus. However, because of the limited transaction validation rate of Sawtooth, the blockchain is only powerful enough for a limited number of validator nodes and transactions rate. In future works, other blockchains will be studied, competing with Sawtooth.

Acknowledgment

This work has been supported by the French government, through the UCA^{JEDI} and EUR DS4H Investments in the Future projects managed by the National Research Agency (ANR) with the reference number ANR-15-IDEX-0001 and ANR-17-EURE-0004.

References

[1] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, 2014.

[3] S. Ghaffaripour and A. Miri, "Application of blockchain to patient-centric access control in medical data management systems," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Oct 2019, pp. 0190–0196.

[4] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[5] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *2017 IEEE International Congress on Big Data (BigData Congress)*, 2017.

[6] N. Szabo, "Smart contracts: building blocks for digital markets," *EXTROPY: The Journal of Transhumanist Thought,(16)*, vol. 18, 1996.

[7] K. L. Brousmiche, T. Heno, C. Poulain, A. Dalmieres, and E. Ben Hamida, "Digitizing, Securing and Sharing Vehicles Life-cycle over a Consortium Blockchain: Lessons Learned," in *9th IFIP Conference on NTMS*, 2018.

[8] M. Cebe, E. Erdin, K. Akkaya, H. Aksu, and S. Uluagac, "Block4forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles," *IEEE Communications Magazine*, vol. 56, no. 10, 2018.

[9] O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT," *IEEE Internet of Things Journal*, vol. 5, no. 2, 2018.

[10] M. T. Hammi, B. Hammi, P. Bellot, and A. Serhrouchni, "Bubbles of Trust: A decentralized blockchain-based authentication system for IoT," *Computers & Security*, vol. 78, pp. 126 – 142, 2018.

[11] M. P. Caro, M. S. Ali, M. Vecchio, and R. Giaffreda, "Blockchain-based traceability in agri-food supply chain management: A practical implementation," in *2018 IoT Vertical and Topical Summit on Agriculture - Tuscany (IOT Tuscany)*, May 2018, pp. 1–4.

[12] J. Benet, "IPFS - content addressed, versioned, P2P file system," *CoRR*, vol. abs/1407.3561, 2014.

[13] M. S. Ali, K. Dolui, and F. Antonelli, "IoT Data Privacy via Blockchains and IPFS," in *Proceedings of the Seventh International Conference on the Internet of Things*, ser. IoT '17. New York, NY, USA: Association for Computing Machinery, 2017.

[14] B. Ampel, M. Patton, and H. Chen, "Performance modeling of hyperledger sawtooth blockchain," in *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2019, pp. 59–61.

[15] S. Benahmed *et al.*, "A comparative analysis of distributed ledger technologies for smart contract development," in *IEEE 30th Annual International Symposium PIMRC*, Sep. 2019, pp. 1–6.

[16] R. Kromes, L. Gerrits, and F. Verdier, "Adaptation of an embedded architecture to run hyperledger sawtooth application," in *IEEE 10th Annual IEMCON*, 2019.