Communicate Vehicle Accident Data to Blockchain for Secure and Reliable Record-Keeping Using Android Automotive Application

Luc Gerrits, Thomas Mabrut, and François Verdier

University Cote d'Azur, LEAT / CNRS UMR 7248, Sophia Antipolis - France
{luc.gerrits, thomas.mabrut, francois.verdier}@univ-cotedazur.fr

***Abstract*—The study presented in this paper demonstrates the potential of blockchain technology to address some of the challenges present in the automotive domain, particularly in terms of providing secure and efficient data handling for real-time vehicle accident data storage. We propose an implementation on Android Automotive OS using a Substrate-based private blockchain and IPFS (InterPlanetary File System). The evaluation of the android application shows successful data communication with IPFS and the blockchain. The paper also exposes the application latency and blockchain metrics analysis.**

***Index Terms*—IoT, BIoT, Android Automotive, Substrate Framework, IPFS**

I. Introduction

The automotive industry is witnessing a shift in the domain of in-vehicle infotainment (IVI) and connectivity, driven by the increasing demand of smart devices, better user experience, additional connected functionalities, and more generally, advanced driving capabilities. Software-Defined Vehicles (SDV) is the result of the gradual transformation of automobiles from *"electromechanical terminals to intelligent, expandable mobile electronic terminals that can be continuously upgraded"* [1]. These new vehicles are powered by a range of operating systems (OS), which are mostly QNX, Android, Linux, and Windows. These new vehicles can be upgraded, thus create value over time. The vehicle value increase by creating links between Original Equipment Manufacturers and automotive/internet software companies. These new vehicles also improve user experience with additional applications to the IVI platform [1]. At present, several vehicle manufacturers are getting ready, if they haven't already done so, to embrace the Android Automotive operating system (an open-source platform developed by Google) as a replacement for their proprietary IVI software. Based on the automotive software and electronics market report, software market is expected to grow 9.1% per year through 2030 [2]. In these modern vehicles, IVI systems are primarily composed of System-On-Chips (SoC) and feature a hardware architecture closely resembling a single-board computer. The overall global IoT market is expected to grow to +43 Billion connected devices and have an $11 trillion impact by 2025 [3] [4]. The automotive industry takes part of this trend. This exponential growth of IoT devices has an forever increasing landscape of challenges that needs to be addressed (centralization, privacy, scalability, security). In the automotive IoT domain, the challenges present significant risks to individuals, including the possibility of severe harm and even include fatalities. Literature shows high interest in blockchain technology as a solution to some of the IoT issues, including for vehicular IoT use cases [5] [6] [7] [8]. Blockchain is a technology that creates a distributed, append-only, ledger maintained by a decentralized peer-to-peer network [9]. Data inside the blockchain is added only after the participating network peers reach a common consensus. Thanks to additional blockchain layer called smart contracts, it is possible to execute pre-defined code in network, thus create applications and automate state changes in the blockchain ledger. Blockchain-IoT (BIoT) provide trust to IoT devices by the intrinsic nature of blockchain. Nevertheless, blockchain is not the ultimate solution for all issues, such as privacy.

In this paper, the IVI hardware is considered as an Internet-Of-Things (IoT): the "vehicle" is a sensing device that is connected to the internet, sends and retrieve data.

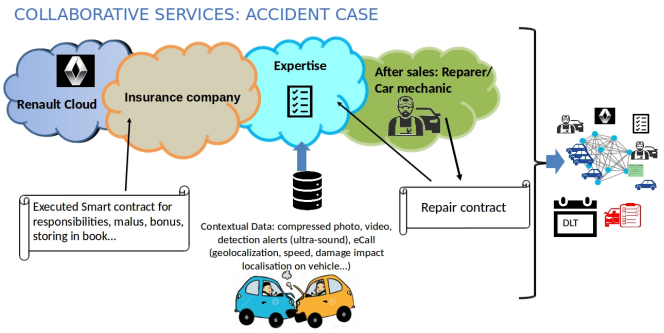

Fig. 1. Renault's use case: an accident occurs

This paper is written in the context of the Smart IoT for Mobility project, which work is driven mainly by the *accident use case*. Connected vehicles are able to send blockchain transactions to the vehicle manufacturer when an accident occurs (Figure 1). The use case focuses on the implementation of blockchain in vehicles (BIoT) and the feasibility of collab-

orative services to use blockchain as a trusted environment to share/or exchange data. For example, when a driver (who previously subscribed to an insurance company), has an accident, the vehicle automatically send its data using blockchain and through the usage of pre-define smart contracts an agreement allows the insurance to view the vehicle accident data. This paper is an attempt to propose and implement blockchain integration inside the vehicle, to communicate its data to the vehicle manufacturer when an accident occurs. Blockchain ensures the data integrity, immutability, identification, and on-chain management through smart contracts.

Section II provides a succinct overview of blockchain technology and a literature review of related works concerning the implementation of blockchain on vehicles. Section III outlines the proposed implementation of an Android Automotive application for the vehicle accident use case. Section IV provides an analysis of the application implementation and its communication latency, while also presenting performance insights of our custom blockchain. Finally, in Section V, we give a brief discussion of our findings and conclude the paper.

II. State of the art

A. Blockchain

The blockchain ledger is the result of an append-only chain of blocks (each linked by the hash of the previous block), that contains cryptographic signed transactions (using asymmetric public-key cryptography and digital signature algorithm). Thus, a user of the blockchain is required to sign, use public-key cryptography to identity himself, and respect the blockchain transaction format to insert valid transaction in the chain.

There are three types of blockchain technology: public, private or consortium. In public blockchains, anyone can participate in the network consensus and read-write data. In private blockchains, the access is restricted and the participants in the network are limited, known and identified. Consortium (or hybrid) blockchains are a combination of public and private blockchain, and the permissions are specific to the use case [10].

Private and consortium blockchains are known to match the technical objectives and expectations of specific type of organization use cases [11]. By controlling the read-write access to the blockchain you removes the economic component from it (i.e. no price volatility). By knowing the blockchain participants, it is also possible to hold someone accountable. Industries are regulated by law, trusted third parties, and require ever more efficient and less costly technical solutions.

Up until 2018, it was necessary to use an existing blockchain and develop smart contracts that mirrors your use case requirements. This forces the application to use the native blockchain cryptocurrency (used as incentive to the network). Blockchain has now evolved to the point that it is possible to create a specific blockchain, that meets your requirements, from scratch using frameworks – especially general-purpose frameworks designed for creating customizable blockchains. At the best of our knowledge, Substrate framework (by Parity Technologies) and Cosmos SDK (by Tendermint Inc.) are the only stable tools that allow creating a specific blockchains. In addition, these two technologies are the result of a common vision, interoperability of blockchain.

Our use case requires extensive customization and the study of the feasibility of interconnected services through blockchain, hence the choice of Substrate framework in this paper. In Substrate, smart contracts can be created using a smart contract virtual machine (executing the contract code on top of the blockchain), or using *modules* (also called pallets). Pallets allows the business logic to be integrated inside the core logic of the blockchain, thus performing better than a traditional smart contract [12].

B. Automotive use case using blockchain

Blockchain has gained sufficient attraction in the automotive industry that manufacturer, researchers, and Original Equipment Manufacturer (OEM) have explored the implementation of this technology on vehicles. In 2016, Leiding et al. introduced the concept of self-managed Vehicle Ad-hoc Networks (VANET) using blockchain (Ethereum blockchain) [5]. The authors proposed an idea of removing centralized structure of VANETs and replace it by Ethereum blockchain, to provide a framework for self-organization and self-management of the network. The paper shows how globally vehicle application and services would communicate with the blockchain without going into technical details or implementations. Blockchain also has been shown to be a potential solution for secure charging Electric Vehicles (EV). Su et al. [13] propose a detailed scheme by only using theoretical models for each blockchain component (network, smart contract, energy, etc.). The resulting simulation discuss the power consumption, costs, and energy demand of their model depending on three schemes. The authors in [14], explore Internet-of-Vehicle (IoV) distributed network architecture and its performance analysis. The authors create a theoretical model of the network and data exchanges between different objects surrounding the vehicle. The resulting simulation express the average throughput, network re-transmissions, and packets sent. To the best of our knowledge, Jabbar et al. have studied the most on the technical implementation of blockchain inside a vehicle. In their first work [5], the authors demonstrate the communication of a vehicle (using Ethereum blockchain) with a prototype using Android application. The application detects driver drowsiness with the purpose of being part of the vehicle Advanced Driver Assistance System (ADAS). The combination of real-time communication between vehicles (V2V) and drowsiness detection could help prevent accidents. The authors of [5] fully implemented the application and evaluated it. In a other work [6], the authors also implemented a new use case (parking management) using blockchain and Android Auto. The authors shows how secure Vehicle-to-Everything (V2X) communication and payment with the parking management use case. They include security analysis, costs, application

metrics, and blockchain metrics. Android Auto[1] is an app running on the driver's phone, thus it has no direct access to the vehicle data.

Gerrits et al. [15], explores and create a prototype application to store vehicle data using a blockchain and IPFS (InterPlanetary File System). The prototype was build on a development board and not directly using a vehicle or vehicle simulation environment. IPFS is a decentralized protocol for sharing and accessing files, based on a peer-to-peer network. IPFS uses content-addressing to ensure that files can be accessed using their unique cryptographic hash (i.e. CID is a hash based on data encapsulated in a special Protobuf format), rather than their location on a specific server. IPFS can help to improve the speed, reliability, and security of file sharing on the internet.

The current challenge lies in integrating blockchain technology more closely with vehicle hardware. With the recent advancements and progressions in the automobile industry, it has become feasible to develop customized applications directly on the vehicle itself. The next two sections present an application solution and its specific blockchain.

III. Automotive blockchain application

In this section we describe the design for a real-time vehicle accident application detection and data storage using Android Automotive and blockchain (Substrate-based). We first describe the application that will run inside the vehicle IVI, next we will explain the blockchain design.

A. Real-time Android Automotive Application

The prototype application aims to retrieve all the vehicle information, sends it to the IPFS file system and stores the data location (a content identifier, CID) in a blockchain (depicted in Figure 2).

Android Automotive is a fully integrated operating system for the IVI. Thus, it has access to various valuable vehicle information, such as speed, acceleration, GPS position, battery capacity, and 136 other sensors. However, access to vehicle sensors is controlled by the Android Automotive framework and the OEM-specific implementation of the platform for security or safety reasons.

When an accident occurs, the application collects the vehicle's data and sends it to IPFS. Although IPFS returns a CID (Content Identifier) when data is uploaded, we create this CID offline in the vehicle to prevent the potential issue of delayed response (or connection loss) from IPFS, which could take several minutes depending on the size of the data. Creating the CID offline ensures that the vehicle does not have to wait for a response before proceeding with the next steps. In the case of a lack of connectivity, the system will retry to connect and send resend the data. Once the offline-generated CID is available, it is sent to a private Substrate-based blockchain belonging to the vehicle manufacturer. This approach allows for more efficient data handling and faster response times in critical situations like accidents.

[1]Android Auto and Android Automotive are not the same

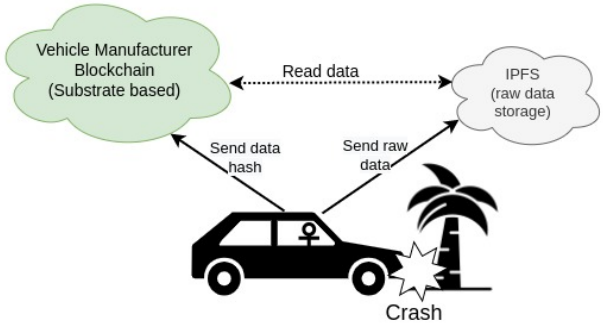

Fig. 2. Accident use case: Android Automotive application connected to a blockchain and IPFS

- The vehicle is equipped with the Android Automotive application, which collects data from various sensors and generates the CID offline.
- The collected data is then sent to IPFS, where it is stored in a decentralized manner.
- The offline-generated CID is transmitted to the Substrate-based blockchain, ensuring secure and efficient storage of the data location. This CID permits also to retrieve the information given to the IPFS by the blockchain.

The application is coded primarily using Android Automotive Java SDK, Substrate API library, and IPFS CID library.

B. Substrate-based blockchain as a secure data registering service

The Substrate framework is used to build a private blockchain belonging to the vehicle manufacturer. This blockchain securely stores the CIDs generated by the application, allowing for easier tracking and retrieval of vehicle data in the event of an accident.

Based on the previous state of art and analysis of organizational requirements, we design a private blockchain with the following key components:

Custom Pallet: A pallet is a modular component in Substrate that contains the logic for specific functionality. In our use case, a custom pallet is created to manage the storage and retrieval of accident data. This pallet includes actions such as adding and querying the stored accident CIDs.

Access Control: To ensure secure data management, a signed transaction is used to store the accident information in the blockchain. Moreover, a simple hierarchical authorization system is implemented to permit only authorized vehicles to store data. The authorized vehicles list is managed by the vehicle manufacturer.

Consensus Mechanism: A consensus mechanism is required to validate and agree upon transactions within the blockchain network. For our use case, a suitable consensus mechanism is Proof of Authority (PoA), given the private nature of the blockchain. In the PoA-based system, a limited number of trusted authorities is responsible for validating transactions. In some cases only one trusted organization is controlling the blockchain validators (possible geographical

decentralization), and in other cases a limited group of organizations can each participate in the consensus (i.e. more decentralized).

IV. Evaluation of the implementation

To examine the proposed implementation, we undertake a comprehensive analysis of the application, the communication protocol with the blockchain, and the interpretation of the created custom blockchain that was created.

A. Android Automotive Application

The application interacts only with IPFS and the blockchain (Figure 3).

1) Implementation details: Android Automotive offers access to a total of 136 sensors, each of which yields data in the form of 32-bit integers (total of 4672 bits). We format the data using a custom Protobuf [16] format. Protobuf allows data formatting with efficient serialization/deserialization. The vehicle application records two categories of sensor data (SD), which depend on their event-driven properties. The first category of sensors is periodically recorded every seconds (e.g. speed, acceleration, GPS position) and only the last ten records are sent when an accident occurs. The second category of sensors is recorded according to the occurrence of events (e.g. we send the last 10 events of the engine state, associated with its timestamp). All timestamps (TS) are standard UNIX 32-bit numbers.

With the vehicle sensors information and record configuration we can estimate the maximum storage size each accident will produce.

$$\begin{aligned} &10 \text{ records} * (64 \text{ TS bits} + 5 \text{ SD} * 32 \text{ SD bits}) \\ &+10 \text{ records} * 131 \text{ SD} * (64 \text{ TS bits} + 32 \text{ SD bits}) \\ &= 128000 \text{ bits} \approx 128.0 \text{ kbits} \end{aligned} \quad (1)$$

Adding pictures or video camera data would significantly increase the storage requirements to several gigabytes. Naturally, with today bandwidth speed, sending 128 kbits (over the internet) is relatively fast, but increasing the data will result in significant latency.

Although the primary focus of the application is not on accident detection, a simple test was performed to simulate an accident by abruptly changing the vehicle speed from 100 km/h to 0 km/h. The application successfully detected the simulated accident and initiated the data recovery and transmission process.

Once the data has been recorded and an accident has been detected, the data is sent to IPFS (step ① on Figure 3). The application start a new thread to POST an HTTP request towards the IPFS API, which returns the CID upon successful upload. To prevent any upload latency, we use an offline generated CID to send an accident report to the blockchain.

To be totally compatible with the IPFS content-addressing, we mimic the CID format standard created by IPFS. A CID is a self-describing content identifier composed of four parts: a multibase, a version, a multicodec and a multihash. The multibase is a self identifying base encoding (base58btc, base64, ...), the version is the binary representation of the CID version (CIDv0, CIDv1, ...), the multicodec is the data format (MerkleDAG, MerkleDAG cbor, ...), and the multihash is the hash function (SHA-256, SHA-512, ...) of the content being addressed.

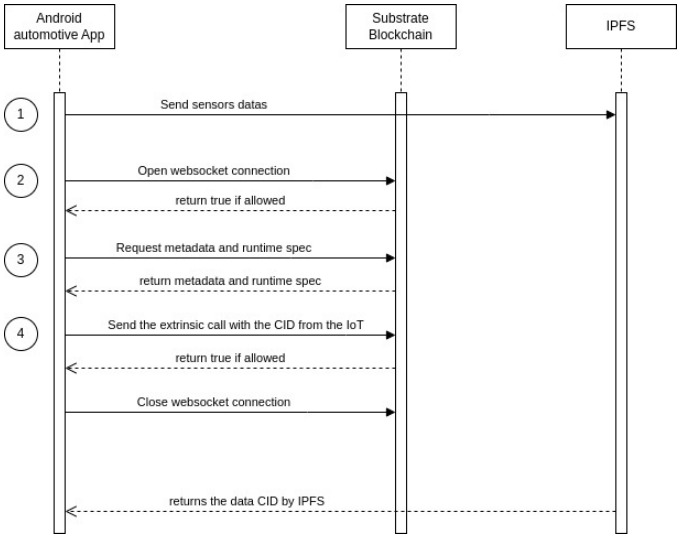

Fig. 3. Interactions between vehicle-blockchain-IPFS after accident occurs

The size of a CID, can vary depending on the version of CID being used and the length of the multihash (256 bits, 512 bits, ...). For example, a typical CID using version 0 with a SHA-256 multihash will be 34 bytes long when encoded in base58. Using CIDv1 with a SHA-256 multihash will be 46 bytes long when encoded in base58. Using different base encodings, such as base16 or base32, will result in different CID sizes. Overall, the size of a CID is relatively small compared to the size of the content it identifies, making it a lightweight and efficient way to reference data in IPFS.

Sending a transaction (to store the accident data) to the blockchain is divided in 3 steps. First, initiate a Websocket TCP connection to a blockchain node (step ② on Figure 3). WebSocket is a protocol used for real-time bidirectional communication between a client and a server over a single TCP connection.

Now that we have open the Websocket connection we need to retrieve elements from the blockchain state that will be incorporated in the transaction. The transaction that will execute the smart contract (called *extrinsic execution* in Substrate) requires to know the chain metadata, (step ③ on Figure 3), specification, latest block number, and call indexes (found in the metadata). Extrinsics are indexed in the blockchain core runtime, along with its actions (e.g. report accident extrinsic n°98 and report accident action n°0 corresponds to *callindex* 0x6200). In summary, our call is composed of the extrinsic version, the size of the transaction, the public key of the account that sign the data, the signature, the era (the number of blocks after the checkpoint for which a transaction is valid. If zero, the transaction is immortal), the nonce (number of transactions sent by a specific account), the tip, the callindex,

and the extrinsic action call arguments.

The signature is performed on the combination of 3 parts previously packed, and sign it using SR25519. These 3 parts are the callindex, the call arguments (in our case the argument is the CID), the extra compose of era, nonce, tip, additional compose of the runtime specversion, the runtime transaction version, the genesis hash, and the hash where we want to add our data. The last step is to send the signed extrinsic call containing the data CID (step ④ on Figure 3).

2) Simulation: We simulate the application using Android open-source software which emulate an Automotive OS with Android Virtual Devices (AVD). In our case we use an OEM-specific (by Volvo) system image from the Polestar 2 vehicle (API v29.2, Android 10.0, X86_64). The emulator includes a dashboard that we use to display information about the vehicle (e.g. speed, GPS, ...) and blockchain (e.g. metadata, transaction status, ...) and additional debug buttons (e.g. force accident detection).

We get the execution latency (Table I) of the application depending on the chosen endpoint (local or cloud). These latency is the time it takes to send a transaction and the data when the vehicle detected an accident. This application is running on a laptop (Intel® Core™ i7-10875H CPU @ 2.30GHz × 16, 32,0 GiB RAM) with a Gigabit Ethernet connection, and communicates to locally installed IPFS and blockchain or with a distant cloud from a cloud provider.

Latency (second)	Local	Cloud
Send transaction to blockchain	1.065	4.971
Send raw data to IPFS	0.023	2.134
Total	1.088	7.105

TABLE I
ANDOID AUTOMOTIVE APPLICATION LATENCY

The results from the local setup experimentation revealed that both blockchain and IPFS could efficiently process requests within a timespan of less than two seconds. Moreover, the difference in latency observed between the local and cloud setups emphasizes the importance of network conditions and can significantly impacts performance. These results imply that vehicle manufacturers must consider the type of communication device utilized within the vehicle to ensure optimal system operation.

B. Blockchain and IPFS

A custom private blockchain is built using Proof-of-Authority consensus algorithm and a custom module that match our use case interaction requirements was created. The blockchain is a fork of the substrate-node-template[2] version *polkadot-v0.9.35*. We deploy the blockchain in two ways: a local *dev* setup (single node, simulated Aura consensus), and in a cloud with 3 blockchain validator nodes setup. The local setup is used to debug and prototype the implementation in the first place, next we use the cloud to get more real-life scenario metrics.

[2]https://github.com/substrate-developer-hub/substrate-node-template/

In the cloud setup we can estimate the maximum throughput by sending multiple accident reports (10k reports) at a constant rate (from 100 to 2k transaction per second). We name the *Input TPS*, the sent transactions per second to the blockchain, and the *Output TPS* the number of transaction included in the blockchain that are finalized (i.e. the transaction is included in a block and immutable).

To generate these results we create a multi-threaded benchmark program to generate identities, initiate the identities in the blockchain, and finally mass-send accident reports transactions to the blockchain. We configure the benchmark to send a total of 10k accident reports.

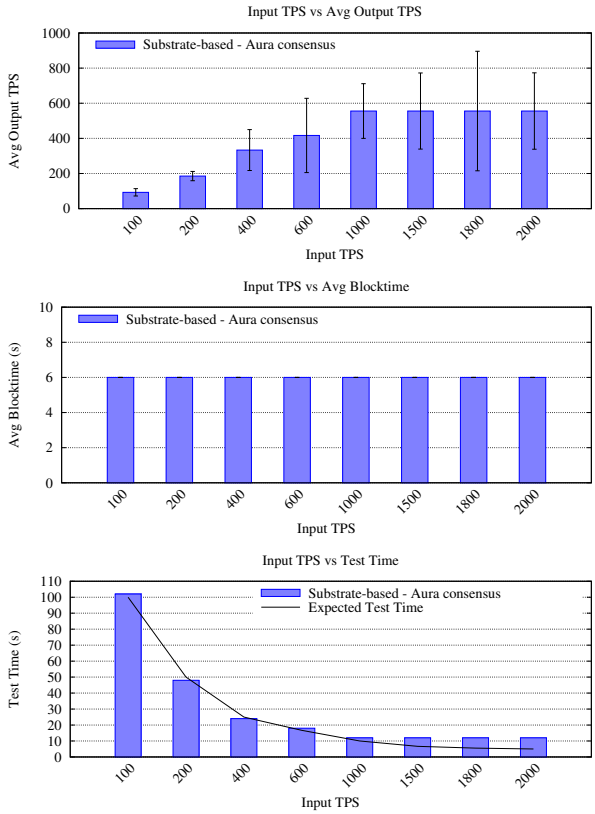

Fig. 4. Blockchain maximum throughput (the average Output TPS is limited by a maximum 560 TPS), blockchain blocktime (6 sec.), benchmark test time (minimum 12 sec.), and expected test time

First, we can observe (Figure 4) that the transaction per seconds (TPS) executed by the blockchain can reach 400 TPS. Secondly, the blockchain consensus is configured to create a block every 6 seconds and the results shows no variation in the block production time if Input TPS increase up to 2k. Thirdly, the test time indicates how much the blockchain uses its transaction processing queue. The expected test time is the theoretical time that the transactions should be executed.

We observe that the test time tends to a minimum threshold of 12 seconds. This is first due to blocks being created every 6 seconds, thus a test can never be faster than 12 seconds (minimum 2 blocks to retrieve performance metrics). However, sending more than 10k transactions with more than 1k TPS,

can't result in a higher Output TPS due to pending transaction queue limitation.

The configuration of our blockchain node software is set to 10k transactions in the pending queue. The blockchain node has a pending queue limit of 10k transactions and can process on average 560 TPS. Thus a continuous Input TPS can't be higher than 560 TPS, otherwise the pending queue will fill, reach 10k, and will start rejecting transactions. A solution could be to implement temporizing transactions in a memory pool before being processed.

V. Discussion and Conclusion

The primary objective of this research paper is to present an implementation of a vehicular use case, which ensures the security of vehicle accident-related data through the utilization of blockchain technology and a decentralized storage file system. Our practical implementation has successfully established communication between the blockchain and IPFS (all code available here [17]). The blockchain includes a smart contract that facilitates a basic authorization check. Our implementation differs from Jabbar et al. [6] work by creating an Android Automotive application which has direct access to OEM-specific vehicle data. Jabbar et al. implement an Android Auto application which is installed on the driver's smartphone.

However, there exists potential for enhancement in crucial areas, namely security, privacy, and confidentiality. Security can be improved by protecting the private keys of the vehicle using hardware security modules (e.g. using physical unclonable functions). Privacy and confidentiality are closely intertwined in fulfilling regulatory requirements, such as the European General Data Protection Regulation (GDPR), which strives to enhance individuals with control and rights over their personal data. While our implementation did not account for privacy and confidentiality concerns, literature suggests that blockchain technology holds immense potential in addressing these issues [18] [19].

A use case specific blockchain was developed using the Substrate blockchain framework. The performance analysis results (560 transactions per seconds) indicate the possibility of the implementation being applicable to a practical scenario. Furthermore, the modular architecture of the Substrate Framework suggests that organizations can leverage its flexibility to their advantage. In addition, we have also presented latency analysis using a vehicle emulation environment to investigate the delays required after an accident occurs (less than 8 seconds). In our further research, we will use the Substrate framework to create a more complex use case to enable multiple interoperable services to communicate together.

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] "Software-defined vehicles – a forthcoming industrial evolution," https://www2.deloitte.com/cn/en/pages/consumer-business/articles/software-defined-cars-industrial-revolution-on-the-arrow.html, accessed: 2023-03-30.

[2] "Outlook on the automotive software and electronics market through 2030," https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/mapping-the-automotive-software-and-electronics-landscape-through-2030, accessed: 2023-03-30.

[3] "Growing opportunities in the internet of things," https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things, accessed: 2023-03-30.

[4] "By 2025, internet of things applications could have $11 trillion impact," https://www.mckinsey.com/mgi/overview/in-the-news/by-2025-internet-of-things-applications-could-have-11-trillion-impact, accessed: 2023-03-30.

[5] R. Jabbar, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, "Blockchain for the internet of vehicles: A decentralized iot solution for vehicles communication using ethereum," *Sensors*, vol. 20, no. 14, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/14/3928

[6] R. Jabbar, N. Fetais, M. Kharbeche, M. Krichen, K. Barkaoui, and M. Shinoy, "Blockchain for the internet of vehicles: How to use blockchain to secure vehicle-to-everything (v2x) communication and payment?" *IEEE Sensors Journal*, vol. 21, no. 14, pp. 15 807–15 823, 2021.

[7] 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.

[8] 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.

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

[10] K. Wüst and A. Gervais, "Do you need a blockchain?" in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2018, pp. 45–54.

[11] M. Jo, K. Hu, R. Yu, L. Sun, M. Conti, and Q. Du, "Private blockchain in industrial iot," *IEEE Network*, vol. 34, no. 5, pp. 76–77, 2020.

[12] "Substrate framework documentation," https://docs.substrate.io/, accessed: 2023-03-30.

[13] Z. Su, Y. Wang, Q. Xu, M. Fei, Y.-C. Tian, and N. Zhang, "A secure charging scheme for electric vehicles with smart communities in energy blockchain," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4601–4613, 2019.

[14] T. Jiang, H. Fang, and H. Wang, "Blockchain-based internet of vehicles: Distributed network architecture and performance analysis," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4640–4649, 2019.

[15] L. Gerrits, R. Kromes, and F. Verdier, "A true decentralized implementation based on iot and blockchain: a vehicle accident use case," in *2020 International Conference on Omni-layer Intelligent Systems (COINS)*, 2020, pp. 1–6.

[16] "Protocol buffers documentation," https://protobuf.dev, accessed: 2023-03-30.

[17] "All code for this paper," https://github.com/projet-SIM/android-automotive-blockchain-2023, accessed: 2023-03-30.

[18] T. Salman, M. Zolanvari, A. Erbad, R. Jain, and M. Samaka, "Security services using blockchains: A state of the art survey," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 858–880, 2019.

[19] D. Wang, J. Zhao, and Y. Wang, "A survey on privacy protection of blockchain: The technology and application," *IEEE Access*, vol. 8, pp. 108 766–108 781, 2020.