POLITECNICO DI TORINO
Repository ISTITUZIONALE

Blockchain-based Mobility Verification of Connected Cars

Original
Blockchain-based Mobility Verification of Connected Cars / Chiasserini, C.F., Giaccone, P., Malnati, G., Macagno, M., Sviridov, G.. - STAMPA. - (2020). (IEEE CCNC 2020 Las Vegas (USA) 10-13 Jan. 2020) [10.1109/CCNC46108.2020.9045104].

Availability:
This version is available at: 11583/2756113 since: 2021-10-10T08:44:12Z

Publisher:
IEEE

Published
DOI:10.1109/CCNC46108.2020.9045104

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2020 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

25 November 2026

Blockchain-based Mobility Verification of Connected Cars

Carla Fabiana Chiasserini, Paolo Giaccone, Giovanni Malnati, Michele Macagno, German Sviridov
Dipartimento di Elettronica e Telecomunicazioni - Politecnico di Torino - Torino, Italy

***Abstract*—Several applications for connected cars leverage the mobility information periodically broadcasted by cars through standard vehicle-to-vehicle messages. We propose an architecture in which each car generates and sends reports including the messages received from its neighbors to the access network infrastructure. The infrastructure collects and stores received reports through multiple blockchains, each of them referring to a different geographical area. A smart contract is then executed to verify the spatial coherence among the received mobility data. We implement a proof-of-concept of such a solution using Hyperledger Fabric, and we investigate the scalability of our solution in terms of resource consumption in a MEC system.**

I. Introduction

In next-generation Intelligent Transport Systems (ITS), cars will continuously exchange *notification messages* with their neighbors. Multiple notification messages standards are currently available, e.g., Cooperative Awareness Messages [1] (CAMs) in ETSI standard or BSM [2] according to the SAE standard. Notification messages are destined to carrying safety-related information. For this reason they typically include information such as the position, the speed, the acceleration and the heading of a car. For simplicity and without loss of generality we will consider CAMs as reference standard. While each CAM is used for a one-time validation of the surrounding environment, if stored, the information contained withing each CAM can be used to track the detailed mobility of the cars and enable the development of new applications. In particular, insurance companies could employ this data to identify those responsible of road accidents and to simplify the resolution of conflicts. Similarly, the same information could be used for other purposes such as identifying drivers not obeying to traffic regulation, or locating stolen cars.

Among the main issues related to CAM-based applications is the unreliability of the exchanged information. Indeed, due to malfunctions or poor GPS coverage, transmitted CAMs may include inaccurate information, while in some scenarios (e.g., insurance frauds) drivers may be interested in transferring deliberately forged information. Furthermore, the information carried by CAMs needs to be securely stored so that no one can tamper with it.

Motivated by the above observations, in our work we propose the BIFOCAL (BlockchaIn For cOoperative CAM vALidation) architecture, which enables the validation of the information contained within each CAM and provides a tamper-evident and distributed data storage. This is made possible by the use of a permissioned blockchain storing a log of the exchanged CAMs. The blockchain runs in servers located within the Multi-access Edge Computing (MEC) system. The validation of the logged information is performed by means of a smart contract that checks the coherence across the mutual positions of transmitters and receivers of the CAMs. BIFOCAL leverages the access network infrastructure and (a part from a small persistent memory for temporarily storage) does not require additional resources at each vehicle.

The rest of the paper is organized as follows. Sec. II discusses related work. Sec. III describes the scenario and motivates the use of the blockchain. Sec. IV presents the designed system architecture. The system performance is investigated in Sec. V. Finally, in Sec. VI we draw our conclusions.

II. Related works

Few works have leveraged blockchains in the context of vehicular networks. The work in [3] proposes a general blockchain-based architecture for supporting a wide range of decentralized vehicular applications. It is based on the Ethereum smart contracts, thus requiring also the presence of miners. Each car directly interacts with the blockchain and requires the execution of a set of applications while also paying transaction fees for this purpose. Direct interaction with the blockchain limits the scalability of the approach since it introduces a considerable amount of transactions to the blockchain and it requires a unique geographically global blockchain.

The work in [4] proposes a different blockchain architecture for vehicular applications. Instead of using existent blockchain technologies, the authors propose a new architecture aiming at greater scalability. Cars are assigned a role which can be either a miner node or an ordinary node. The infrastructure devices (e.g., base stations) are assigned a role of controller nodes and have as sole purpose that of providing means for information exchange among different network entities. All the collected data is process and stored locally at each miner node so that, if needed, this information can be distributed to other network participants. Ordinary nodes instead can only access data provided by miner and controller nodes. The main limitations of [4] are the fact that additional hardware devices are required inside the cars in order to process and store data locally at each car. In addition to that, scarce availability of miner nodes may lead to a degradation in the transaction throughput.

III. Blockchain-based mobility verification

Vehicles communicate between each-other by exchanging CAM messages. Although CAM messages are used both for vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications, in this work we focus only on V2V CAMs. Every car periodically broadcasts a CAM including its identifier (randomly generated for privacy reasons), its geographical position alongside with the corresponding time at which the position has been evaluated, and its speed, heading, and acceleration.

V2V CAM transmission does not include delivery guarantee nor any type of security measure as: i) CAMs may be lost due to radio channel congestion or harsh propagation conditions; ii) although [5] specifies an authentication and message integrity standard for CAM messages, it applies for V2I messages only. The combination of the two factors leads to the possibility of performing untraceable CAM forgery.

A. Enabling mobility verification

CAM messages provide mobility information of a car, and allow to reconstruct fine-grained dynamics of urban mobility. Many organizations may be interested in tracing back detailed information related to individual vehicles. As an example, insurance companies may be interested in this information, e.g., for liability reasons, car manufacturers for monitoring purposes, governments for billing. Each organization may be interested in keeping mobility information for themselves and may gain some advantage by being able to modify such information. Storage of CAMs for future analysis raises issues related to trust among organizations. Furthermore, due to possible forgery, blind trust in the information contained in each CAM is inconceivable.

It follows that there is a need for an architecture providing: i) a distributed *immutable* storage, ii) *validation* of the information contained inside each CAM to detect forgery and iii) cooperation among mutually untrusting organizations. All these properties can be satisfied by employing a blockchain since it i) is append-only by construction, ii) involves untrusted entities and iii) allows information validation by means of smart contracts. However, the use of traditional blockchains comes with significant overhead in terms of transaction latency, small throughput, complex commit operations and overall scalability. Nevertheless, recent proposals address these issues by introducing the concept of *permissioned blockchains*.

B. Permissioned blockchain

To understand the benefits provided by the use of permissioned blockchains we will consider as an example Hyperledger Fabric (HF) [6], which is also the platform adopted in our proposed BIFOCAL architecture. In HF,

1) high throughput is achieved in terms of transactions per second, which is necessary for the high amount of CAMs generated in a mobility scenario.
2) the commit latency is small (typically less than 1 s), thanks to the presence of an auxiliary key-value store besides the conventional hashchain. This database permits fast read access, while the hashchain is used as a tamper-evident replica of the data.
3) the access is permissioned, i.e., differently from public blockchains, in HF only authorized users (identified by public key certificates) can access the blockchain, issue transactions and retrieve data. Thus in BIFOCAL, only authorized public/private organizations can access mobility data, thus protecting users' sensitive information. Furthermore, different companies can interact and manage the ledger without the intervention of a trusted authority controlling the whole process. The role of the trusted authority is limited to the issuing of certificates to authorized members of the blockchain.
4) multiple independent ledgers, called *channels*, are possible, each maintained by a different subset of peers. In BIFOCAL this permits to maintain different ledgers, each containing data relative to a given geographical area, thus improving the overall architecture scalability.
5) smart contracts, named *chaincodes*, implement CAM validation algorithms to check the reliability of information contained within transmitted/received CAMs.

The majority of the aforementioned features are achieved thanks to a peculiar architecture employed by HF. This architecture is known as execute-order-validate and it is composed of four agents:

- *clients*: are the actual users requiring the execution of transactions;
- *endorsing peers*: receive transaction execution requests from the clients and perform a pre-commit by simulating its execution. After this, the endorsing peers reply to the clients with a signed outcome of the operation;
- *orderers*: after the endorsement process, clients forward the response of the endorsing peers to the orderers which form an Apache Kafka cluster. These agents are primarily responsible of ordering in a chronological way all the transactions and ultimately creating a block;
- *committing peers*: after the block has been created, orderers forward it to the committing peers. Committing peers verify the correctness of the transactions inside each block and then store them, each of them holding a full replica of the ledger.

HF permits different organizations to cooperate. Each organization can manage different members (clients, peers, orderers) and interact with other organizations to operate the ledger.

IV. BIFOCAL Architecture

The proposed BIFOCAL architecture is shown in Fig. 1. We consider an urban geographical area served by one or more mobile network operators, where user access is provided through base stations (BSs) such as LTE eNodeBs. We assume that each vehicle is equipped with a 4G USIM (Universal Subscriber Identity Module), permitting a secure communication with a mobile network infrastructure.

Upon transmitting a CAM, a car generates a *report* with contains a copy of the generated CAM alongside with the list

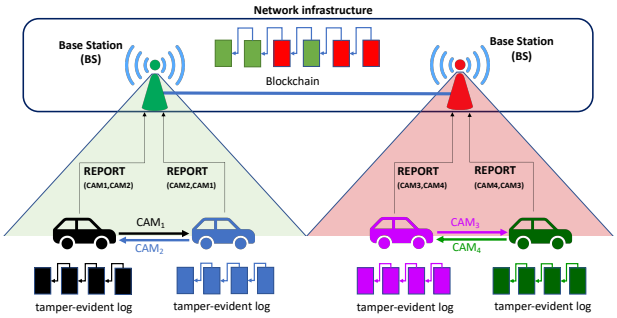


Fig. 1: Scenario and BIFOCAL architecture.

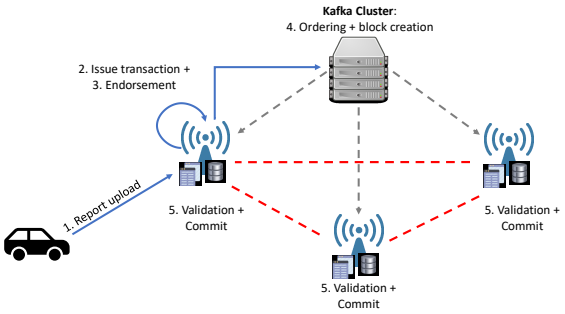


Fig. 2: Interactions between BIFOCAL entities.

of all the CAMs that have been received from neighboring cars since the last report. The car sends the report to the Base Station (BS) to which it is connected. If out of coverage, the car will send it as soon as it gets connected again.

Since cars have limited computational and storage capabilities, they store only a simple tamper-evident log of all its generated reports. On the other side, the network infrastructure leverages the MEC paradigm to host the peers operating the blockchain, which logs all of the reports received from the cars.

Below, we describe the main system components, their role, and, without loss of generality, how they interact with reference to a single mobile network operator.

A. Report log in the cars

The in-car tamper-evident log of the locally generated reports is implemented using a classic hash-chain [7]. In such a chain, the ith commit C_i, after event X_i has been inserted, is computed by applying a hash function $H(\cdot)$ on the combination of X_i and the previous commit C_{i-1}, i.e.,

$$C_i = H(X_i||C_{i-1})$$

. Thus, a membership proof can be easily implemented for any stored event X_i and for any subsequent commit C_j, with $i \leq j$. Note that this hash-chain is similar to the structure of a blockchain, but it is implemented locally by each car in a centralized way, differently from the distributed nature of a blockchain.

Each received CAM is associated with the following additional information: position, speed, acceleration, heading, and timestamp of the receiving car, all evaluated at the instant of the message reception at the physical layer. As explained later in Sec. IV-D, this additional information enables the validation of the data carried by the CAM.

B. Report storage and validation in the blockchain

The overall interaction with the blockchain is managed by the BSs while cars just play the role of generating reports and uploading them to the BSs, and are oblivious to the presence of a blockchain. Likewise, the blockchain is kept unaware of the real identity of cars thus ensuring user privacy, as the mobile network operators are in charge of the authentication of vehicular users.

In BIFOCAL, each BS maintains a local copy of a portion of the ledger (i.e., one or multiple channels) and acts as a client as well as an endorsing and committing peer.

For each report received at a BS, multiple key-value pairs are generated, each corresponding to a different reported CAM. The key includes the time at which the CAM was generated, as well as the sender and the receiver identifier. The value includes the position, speed, acceleration, and heading of the CAM sender and of the CAM receiver, at the time of transmission and reception, respectively.

Fig. 2 describes a typical interaction between the different entities in BIFOCAL. Upon the reception of a report (step 1), a BS issues a transaction with the goal of storing the received report inside the blockchain (step 2). In accordance with the behavior of the HF endorsing peers, the BSs execute the chaincode functions that are necessary to store and validate reports previously logged in the blockchain (step 3). The transaction is then transmitted to the orderer, which orders the transaction and creates the corresponding block (step 4). Finally, the created block is distributed back to the committing peers so that it can be validated and committed (step 5). For every newly committed block, the ledger state database is updated with the addition of as many new key-value pairs as the number of CAMs included in the report for which the block was generated.

Since committing a transaction to the ledger involves considerable processing burden for all of the agents, we consider bulk report commits. That is, after a predefined number of reports has been collected, each BS issues a transaction triggering the storage of multiple reports simultaneously into the blockchain at the cost of a single commit. This strategy permits us to reduce the storage overhead since the execution of transactions requires to store the digital signatures of peers that endorsed the transaction.

At last, we remark that the interactions and the transaction (and related validation) described above are solely aimed at storing the reports, hence the CAMs and the information related to their reception, in a tamper-evident manner. As detailed below, subsequently to the storage of reports, another kind of transaction and validation occur in the system, which

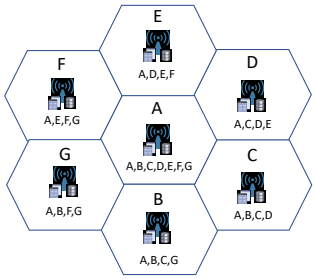


Fig. 3: Architecture composed of several channels, each storing data related to a different geographical area.

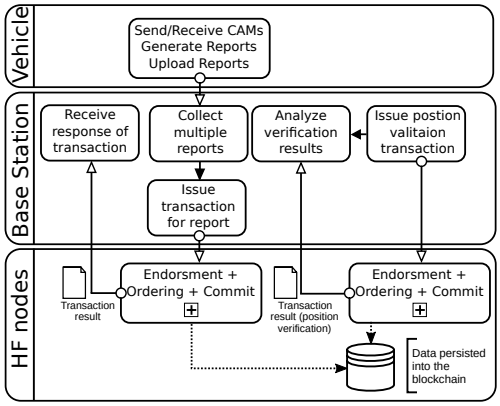


Fig. 4: Process of storage and validation of the CAMs

are aimed at the verification of the information carried by the reported CAMs.

C. Geodistributed blockchain architecture

As mentioned, HF permits the creation of different *channels*, i.e., separated ledgers that increase the system scalability and the privacy of stored information: peers may store only some channels, and only peers belonging to a given channel can access and retrieve the data stored therein. Importantly, peers can access simultaneously different channels.

In BIFOCAL, channels are used to create different ledgers, each referring to a well-defined geographical area, which naturally maps into the coverage area of a BS. The reports collected by a BS are stored only in the channel of its coverage area. Fig. 3 shows the case where, depending on their geographical position, BSs store data of different channels. The letters *A, B, C, D, E, F, G* identify the channels and the relative geographical areas. A BS peer must join only the channels corresponding to the areas covered by its own cell site and by those of its neighbouring BSs (which may belong to a different mobile operator). In this way, on the one hand, each BS will maintain a replica of a reduced portion of data, on the other hand it will have visibility of the reports sent by all cars under its coverage as well as of those sent by cars (potentially under the coverage of neighboring BSs) that received such CAMs. The latter permits the BS to have a broader view of the geographical area, enabling an effective execution of position verification algorithms. However, the validation of reports may require to retrieve data stored in different channels.

It is important, however, to highlight that the management of different separated ledgers leads to a higher system complexity. This is because the algorithms for the verification of the information carried by the CAMs need to retrieve the CAMs stored in different channels, increasing the number of interactions with the blockchain. Another problem related to the management of different channels is the identification of cars launching a Sybil attack. Indeed, in the presence of one channel only, it is possible to execute a query to understand that the victim of the attack is simultaneously located in different geographical regions, while the use of different channels makes it difficult to identify this condition. In our case, this problem is overcome by the adoption of the in-car tamper-evident data structure of reports, as previously discussed in Sec. IV-A.

D. Validation of CAM messages

The application of position verification algorithms to CAMs contained in reports is possible only after the reports have been committed to the ledger (i.e., they have been stored in the blockchain). Most importantly, the position verification process may require to access the content of transactions initiated by different peers (i.e., BSs). We therefore introduce a *processing time window*, W_p, between the time when the reports are stored at a BS and the time when the CAMs are validated therein. Clearly, the value of W_p should be carefully tailored in order to establish the best trade-off between latency and reliability. Small values of W_p imply a low latency before a CAM is validated, but decrease the reliability of the CAM verification since only a subset of the generated reports may be already available in the blockchain and, hence, used for verification. Conversely, large values of W_p increase the latency but also the reliability of the process.

The overall validation process is shown in Fig. 4 and can be summarized as follows:

1) The BS, acting as a client of the blockchain, issues a transaction to store the reports (each typically including multiple CAMs), received from the cars in the blockchain.
2) After the transaction has been committed, the BS waits W_p to ensure that its neighbouring BSs have stored their collected reports in the blockchain.
3) The BS issues one or more transactions to apply the position verification algorithm to the key-value pairs (each associated with a CAM) stored in the blockchain.
4) The BS collects and analyses the result of the validation as soon as the transaction is committed.

We implement a simple position verification algorithm, which is prototypal with respect to more advanced algorithms, such as those in [8], [9]. The algorithm, named Simple Position Validation (SPV), works as follows.

Consider a CAM that is transmitted by car TX located in position $(x_{\text{TX}}, y_{\text{TX}})$ at time t_{TX} and it is received by car RX

Algorithm 1 Simple Position Validation (SPV) algorithm

Input: $\mathcal{K}$: set of all the transmitted CAMs
Input: ϵ: maximum location error [m]
Input: $d_{\max}$: maximum radio range [m]
Input: $n_{\min}$: minimum number of witness cars
Output: Verification Information

1: **for all** $k \in \mathcal{K}$ **do** ▷ For each CAM
2: $c \leftarrow$ queryCAM(k) ▷ Query the blockchain to retrieve the CAM
3: TX is the transmitter of CAM c
4: $n_V \leftarrow 0$ ▷ Init number of positive votes
5: $n_I \leftarrow 0$ ▷ Init number of negative votes
6: $\mathcal{N}_c \leftarrow$ queryReceiversCAM(k) ▷ Query the blockchain to get the reports of cars that received c
7: **if** $|\mathcal{N}_c| < n_{\min}$ **then** ▷ Check minimum number of witness cars
8: c.status $\leftarrow$ undecided ▷ Insufficient number of witness cars
9: updateBlockchain(c)
10: **continue** ▷ Consider a new CAM
11: **end if**
12: **for all** RX $\in \mathcal{N}_c$ **do** ▷ For all cars that received c
13: **if** both equations (1) and (2) hold **then**
14: $n_V \leftarrow n_V + 1$ ▷ Vote for valid CAM
15: **else**
16: $n_I \leftarrow n_I + 1$ ▷ Vote for invalid CAM
17: **end if**
18: **end for**
19: **if** $n_I \geq n_V$ **then** ▷ Check if majority is for invalid CAM
20: c.status $\leftarrow$ invalid ▷ Invalidate the CAM
21: **else**
22: c.status $\leftarrow$ valid ▷ Validated CAM
23: **end if**
24: updateBlockchain(c)
25: **end for**

located in position $(x_{\text{RX}}, y_{\text{RX}})$ at time t_{TX}. RX is denoted as *witness car*. Let $d_{\text{TX,RX}}$ be the physical distance between TX and RX based on their positions (evaluated at the time of, respectively, transmission and reception at the physical layer) declared in the report[1]:

$$d_{\text{TX,RX}} = \sqrt{(x_{\text{TX}} - x_{\text{RX}})^2 + (y_{\text{TX}} - y_{\text{RX}})^2}$$

To validate the CAM, the propagation delay between the cars must be compatible with their relative distance as well as with the maximum radio range $d_{\max}$, i.e.,

$$c \cdot (t_{\text{TX}} - t_{\text{RX}}) - \epsilon \leq d_{\text{TX,RX}} \leq c \cdot (t_{\text{TX}} - t_{\text{RX}}) + \epsilon \quad (1)$$

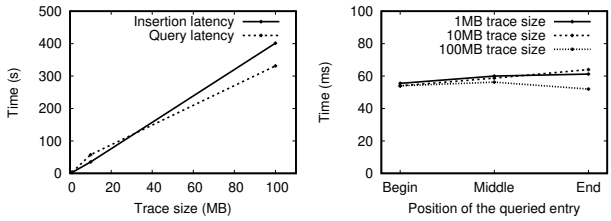
$$d_{\text{TX,RX}} \leq d_{\max} \quad (2)$$

where c is the speed of light and ϵ is the maximum location error taking into account factors such as clock synchronization, non-instantaneous transmissions (due to radio channel congestion at MAC layer), and processing delays.

The validation process, verifying the conditions in (1) and (2), has been implemented through a chaincode function, whose pseudo-code is reported in Algorithm 1. The position verification algorithm is based on a voting scheme (lines 12–21), which evaluates the number of consistent and inconsistent reports with respect to the information available in a CAM. The voting is executed only if enough witness cars have received the CAM (lines 7–10). Due to the append-property of the blockchain, for performance reasons, in SPV the ledger is updated only if the CAM is considered invalid (lines 20–21).

As a final remark, HF requires that the committing peers of the blockchain maintain a complete replica of the ledger. No other alternatives like sharding are possible. This poses some scalability issues due to the always growing record of transactions and blocks. Notably, it is not possible to remove "intermediate" blocks, otherwise the tamper-evident property of the blockchain is violated.

[1]More accurate formulas could be used as the Harvesine one.



(a) Transaction and query latency as functions of the trace size (b) Query latency as a function of the CAM entry position

Fig. 5: Transaction and query latencies

V. Simulation results

We deployed HF version 1.1.0 inside multiple Docker containers (one for each peer and orderer) running on Ubuntu 16.04 in a private cloud system available at our university. To validate the proposed architecture and solution, we followed a two steps approach. As first step, we created synthetic traces of the reports uploaded to the BS, based on a vehicular mobility simulator able to model the CAM exchanges between the cars and from the cars to the network infrastructure. The full traces carry 100 MB of CAM data, comprising approximately 60,000 reports. As second step, we emulated a real scenario by feeding to HF the report trace. Thus, we could test the operations of insertion, search, and validation of each individual CAM.

A. Numerical results

We conducted an exhaustive set of experiments to test the scalability and the overall performance of the proposed architecture.

1) HF interaction latency: We first focus on the time to perform the insertion and retrieval of the CAM data traces inside the blockchain. We consider the case of one organization (i.e., one mobile network operator) with two peers and one orderer. Fig. 5a depicts the latency related to the insertion and query of a full trace as the trace size varies. As expected, for both insertion and query, the latency grows linearly with the amount of data to insert/query, since, thanks to the presence of the auxiliary database, there is no need to scan the whole blockchain to retrieve a particular entry. The latter fact is highlighted in Fig. 5b, which shows almost no variability in accessing entries that are stored at the beginning of the blockchain, in the middle, or at the end of it. Such behavior essentially translates into a deterministic processing time of new reports, and it provides latency guarantees for the query of CAMs needed during the position verification execution.

2) Transaction overhead: As previously mentioned, in order to avoid significant storage and processing overhead we

TABLE I: Storage and performance to store 10 MB of report data, equivalent to 6000 reports.

Trans. size [kB]	Num. blocks	Num. trans.	Total storage overhead [%]	Total storage latency [s]	Reports per sec.
10.88	90	894	30.02	122.94	48.8
98.74	10	99	4.99	35.34	169
244.95	5	40	3.33	31.83	188
489.75	2	20	2.42	29.10	206
977.65	1	10	2.29	29.10	206

employed bulk commit, which involves transactions containing more than one report at a time. We executed a set of experiments by performing multiple storage transactions and by varying their size while maintaining constant the number of per-block transactions, equal to 10 transactions per block. The experimental results are reported in Table I.

By construction, the number of stored reports for each transaction is proportional to the transaction size, thus the total number of transactions is inversely proportional to the transaction size, hence to the required number of blocks, given a fixed amount of reports to store. The results show that, for minimum-size transactions, a considerable overhead is experienced in terms of storage, equal to about 30% of the total size of the blockchain. Indeed, each transaction contains the digital signatures necessary for the identification of the peers that executed the endorsement of the transaction. Notably, it is sufficient to increase the size of each transaction by one order of magnitude, to reduce this overhead by almost a factor 6. Further increase in the transaction size provides marginal decrease in the introduced storage overhead. A similar behavior is exhibited by the overall time required to execute all transactions, which decreases roughly by a factor 3.5. Furthermore, the latency for each transaction increases with the size of each transaction, as expected. The throughput in terms of stored reports over time varies from (roughly) 50 to 200 reports/s. We can conclude that bulk commits increases the throughput, at the expenses of the increased latency to collect all the required reports.

3) Scalability analysis: We assessed the scalability of our approach by varying the number of peers, organizations, and orderers, in the scenario with 6000 reports to store (i.e., 10 MB trace). Fig. 6 shows that the time required to store the reports grows linearly with the numbers of peers ≤ 8, after which it remains stable. Fig. 7 instead depicts the overall CPU and RAM resources utilization of all peers in the cloud system. The linear behavior as a function of the number of peers highlights that there is no significant increase in terms of CPU and RAM consumption per peer, as the number of peers grows.

The number of organizations does not impact either the scalability of the system. Indeed, by fixing the total number of peers and by equally splitting them across multiple organizations (whose number is also varied), no change in terms of storage time can be observed. We do not report here the results for the sake of space. Similarly, the impact of the number of orderers is negligible: by increasing the number of orderers from 1 to 12, it can be shown that the processing time for the insertion of a block increases by a mere 11%.

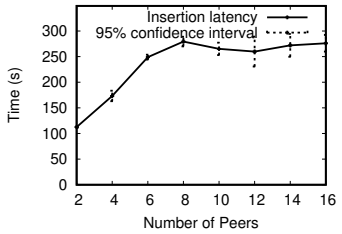


Fig. 6: Total time to store 6000 reports

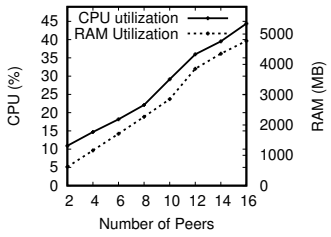


Fig. 7: Overall CPU and RAM consumption to store 6000 reports

VI. Conclusions

We proposed a novel architecture to store and validate the CAMs exchanged among connected cars. Our solution is based on a local tamper-evident log of the CAMs received by a car and a collective CAM report sent by each car to the network infrastructure. The network infrastructure runs a distributed ledger through a multi-channel blockchain technology. A smart contract discovers invalid CAMs, i.e., the ones transmitted with incoherent positions with respect to the positions of the receiving cars. We implemented a prototype of the solution on Hyperledger Fabric and tested using synthetic CAM traces, and we assess the performance in terms of transaction latency and computation and storage resources.

References

[1] "ETSI EN 302 637-2 V1.4.1," http://www.etsi.org/, Apr. 2014.

[2] J. B. Kenney, "Dedicated short-range communications (dsrc) standards in the united states," *Proceedings of the IEEE*, 2011.

[3] B. Leiding, P. Memarmoshrefi, and D. Hogrefe, "Self-managed and blockchain-based vehicular ad-hoc networks," in *ACM UbiComp*, 2016.

[4] P. K. Sharma, S. Y. Moon, and J. H. Park, "Block-VN: A distributed blockchain based vehicular network architecture in smart city," *Journal of Information Processing Systems*, vol. 13, no. 1, p. 84, 2017.

[5] "ETSI TS 103 097 V1.3.1," http://www.etsi.org/, Oct. 2017.

[6] "Hyperledger fabric," https://www.hyperledger.org/.

[7] S. A. Crosby and D. S. Wallach, "Efficient data structures for tamper-evident logging." in *USENIX Security Symposium*, 2009, pp. 317–334.

[8] M. Fiore, C. E. Casetti, C. F. Chiasserini, and P. Papadimitratos, "Discovery and verification of neighbor positions in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, Feb 2013.

[9] F. Malandrino, C. Borgiattino, C. Casetti, C. F. Chiasserini, M. Fiore, and R. Sadao, "Verification and inference of positions in vehicular networks through anonymous beaconing," *IEEE Transactions on Mobile Computing*, vol. 13, no. 10, pp. 2415–2428, Oct 2014.