Blockchain and NFTs-based Trades of Second-hand Vehicles

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.0092316

Blockchain and NFTs-based Trades of Second-hand Vehicles

ALBERTO BUTERA[1], VALENTINA GATTESCHI[2], *Senior Member, IEEE*, F. GABRIELE PRATTICÒ[3], *Member, IEEE*, DANIELA NOVARO[4] and DEBORAH VIANELLO[5]

[1]Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy (e-mail: alberto.butera@polito.it)
[2]Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy (e-mail: valentina.gatteschi@polito.it)
[3]Dipartimento di Automatica e Informatica, Politecnico di Torino, 10129 Torino, Italy (e-mail: filippogabriele.prattico@polito.it)
[4]REPLY, Milano, 20151 Italy (e-mail: d.novaro@reply.it)
[5]REPLY, Milano, 20151 Italy (e-mail: d.vianello@reply.it)

Corresponding author: Valentina Gatteschi (e-mail: valentina.gatteschi@polito.it).

ABSTRACT Recently, the automotive industry has been characterized by disruptive innovations, like self-driving cars or hybrid/electric engines. Despite this fact, some operations, such as the trade of second-hand vehicles, still continue to be carried out in the "traditional" way, in which the buyer has to trust the seller about the state of the vehicle. Several studies highlighted that odometer fraud alone could cost around 8.9 billion euros per year. In order to overcome these limitations, which are related to information asymmetries between buyers and sellers, in this work we propose to exploit blockchain technology to store a previous vehicle's history in a transparent way. To further explore blockchain advantages, we also present how a decentralized second-hand vehicle market – enabling also automatic transfers of ownership upon monetary transfers – can be built, leveraging on Non-Fungible Tokens (NFTs). We propose an architecture and a practical implementation of a Decentralized Application (Dapp) and discuss the security of the proposed system, its costs, and future developments.

INDEX TERMS Blockchain, decentralized marketplaces, digital passport, ERC Standards, Ethereum, NFTs, second-hand vehicles trading, smart contracts, used vehicles.

I. INTRODUCTION

During recent years, the automotive industry has seen several innovations that are hitting the market, such as hybrid/electric engines, self-driving cars, or IoT-connected cars [1]. Apart from these innovations, another one is related to the exploitation of blockchain technology in the automotive industry. According to researchers, this technology could potentially bring tremendous benefits to the automotive sector [1], [2]. In particular, blockchain technology could be used to support vehicle-to-vehicle communication and secure data transactions, location tracking or component provenance, transparency, security. Importantly, could also be the foundation to support the decentralization of several actors, such as vehicle owners/lenders, fleet management companies, vehicle sharing users, vehicle dealers and retailers, OEM, insurance companies and so on [3].

Some recent research efforts were particularly devoted to investigate how blockchain technology could improve the trading of used vehicles [4]. In fact, this scenario is characterized by a lack of transparency among buyers/sellers/intermediaries, which could potentially end in severe frauds [5]. As a matter of example, a study carried out by the European Parliament in 2017 highlighted that up to 40% of second-hand cars traded across borders have a rolled-back mileage counter, resulting in a loss of around 8.9 euro billion per year [6]. Odometer frauds are not the only problems actors encounter during the trading of second-hand vehicles: other possible areas of improvements, thoroughly detailed in [7] are as follows:

- From the buyer's perspective, the possibility to buy the desired vehicle brand is related to the current second-hand offer (as well as to the buyers' ability to search for information on second-hand vehicles currently on sale). For the buyer, it is also difficult to determine a fair price for a vehicle. Furthermore, as previously mentioned, since the price is related also to odometer data, odometer readings could be tampered, or, even worse, vehicles that had a severe accident could be repaired and re-sold, making it hard to assess the severity of the occurred accident;

- from the seller's perspective, finding the "right" buyer is a complex task. Some sellers feel uncomfortable answering requests from online profiles which do not have a picture. Apart from this aspect, the main drawback for them is that genuine sellers face problems in convincing the buyer that the second-hand vehicle is in mint condition;
- from the intermediary's perspective, assessing the conditions of the vehicle is difficult. In fact, even though intermediaries rely on inspection centers, even if inspections are performed with the latest technology, it is not always possible to assess the condition of the vehicle.

Before explaining the advantages of blockchain technology applied to the second-hand vehicle market, a brief overview of this technology could be useful to the reader. A blockchain is a shared ledger, stored on and maintained by a network of nodes, recording transactions executed among nodes [8]. The information stored on the blockchain is publicly available, thus guaranteeing transparency, and cannot be modified or erased, hence guaranteeing immutability. In a simplified view, the blockchain could be imagined as a chain of blocks, which is only incremental and which can only grow in one direction. At given time-frames, a new block is added to the chain by network nodes. The nodes which maintain the chain have an incentive, i.e., they can participate in a lottery to receive some tokens. Each blockchain has its tokens and their price varies based on the value assigned by the market. This technology was devised in 2008, and it is the underlying mechanism of Bitcoin [9]. As time passed, researchers proposed to use the blockchain to store not only information but also executable code. It was the birth of smart contracts, self-executing pieces of code stored on the blockchain, which could act autonomously in case given conditions occur [10].

Blockchain technology, being able to store in an immutable and transparent way a vehicle's previous history (e.g., services, repairs, insurance, pollution check events, etc.), could overcome the above-described threats, characterizing the second-hand vehicle market. Furthermore, focus groups with younger generations (generation Y and Z) showed that they appreciate the potentiality of this technology for second-hand vehicle trading, in particular, if they act as individual buyers or sellers. Intermediaries also appreciate this technology, but perceive it as a threat to their business, as their business is based on the poor information existing in the market of second-hand vehicles [7].

In this paper, we propose a novel architecture that leverages the benefits of blockchain, NFTs, and decentralized storage to create a secure and efficient system for tracking the ownership and the history of vehicles. By associating a dynamic NFT with each vehicle, we enable the tracking of ownership transitions, maintenance, and certificates, which can be easily accessed and verified by users. In addition, this architecture allows users to buy and sell vehicles directly by exchanging the associated NFTs, streamlining the process and reducing the risk of fraud.

In order to achieve this objective, we create a prototypal Decentralized Application (Dapp) by relying on the most famous blockchain supporting smart contracts and Dapps development – Ethereum – and exploiting ERC-721 tokens (i.e., Non-Fungible Tokens or NFTs). NFTs are cryptographically unique, non-replicable tokens, which have been designed to track the ownership of individual assets. Their main characteristics, that make them differ from fungible tokens (such as ERC-20 tokens, the "standard" tokens used on the Ethereum blockchain to transfer value) are that they cannot be divided and that each token contains distinctive information and attributes that make the token unique [11].

NFTs have already been successfully used in several contexts, e.g., in the medical domain to trace the ownership of medical devices [12], [13], in commerce to trace the ownership and shipment of jewels [14], or the state of shipping containers [15], or to support the ownership, trade, and train of AI models [16], [17], or the monetization of private data [18].

It is speculated by the authors that exploiting NFTs for the second-hand vehicle market could bring several advantages:

- the ownership of a vehicle could be easily transferred upon payment;
- NFTs smart contracts could be written adopting existing standards and libraries – such as the OpenZeppelin library [19] – making them more secure;
- several decentralized marketplaces for NFTs already exist, which automatically add NFTs a few seconds after they are minted (i.e., created). In this way, the user would not need to publish the same offer on several websites, instead a single portal would contain all the active vehicle offers;
- since the NFT would track the previous history of the vehicle, the user would only need to fill a few form fields, mainly related to the desired price, or to the end of the selling time-frame. The marketplace would then automatically read the vehicle's previous history from the blockchain and fill in the missing fields;

The rest of the paper is organized as follows: Section II provides an overview of related works in the field of blockchain-based vehicle history tracing and second-hand vehicle trading. Section III describes the architecture of the proposed solution, whereas detailed information on implementation aspects is reported in Section IV. Section V presents the procedure we followed to test and evaluate the proposed system, whereas Section VI discusses the security and costs of the system. Finally, Section VII presents conclusions and future works.

II. RELATED WORKS

During the last few years, several attempts have been made, both by companies and by universities, on how to store car-related information on the blockchain and make them available to other users. Concerning efforts carried out by companies, Renault group was one of the first companies at performing this task, and it teamed up with Microsoft and

VISEO to develop a digital car maintenance book, based on blockchain technology [20]. Startups such as VINChain [21] and CarVertical [22] were created, with the objective to gather data from multiple sources (e.g. country registries, insurance and leasing companies, police and INTERPOL databases, private registers, paid APIs, and other sources) and store them on the blockchain by linking them to the Vehicle Identification Number (VIN).

Concerning efforts carried out by universities and research institutes, in the literature several works could be found that deal with the storage of the previous vehicle's history. In [23], a system to track a vehicle's previous history is presented. In particular, the system could be used by car owners, manufacturers, trusted third parties, road authorities, insurance companies, charging stations, and car-selling services. All these actors are part of a consortium and, based on their permissions, can write the blockchain. The work described in [24] presents an odometer fraud prevention system. The devised system records on the blockchain mileage and GPS data, in order to avoid odometer fraud. Data are retrieved from a dongle, connected to the on-board diagnostics II (OBD-II) interface. While the presented solution is prototypal (as it sends data to a laptop in the car, which then signs and sends a transaction to the blockchain), the authors discuss the privacy-related aspects and suggest encrypting raw data before sending them to a cloud database, in order to let only the holder of the car to which this data belongs to access them. The authors of [25] present DeMeterA, a system that records on the blockchain odometer readings and is able to track the ownership of a car. In particular, the authors represent a car as a pair private-public key pair, which is created by the authorized seller when he/she registers the asset on the blockchain. The key pair is then transferred to the car owner, that can decide whether to store it on a phone, or on external devices, such as SD-cards or Near Field Communication tokens. The private key is used for updating the odometer readings or to transfer the ownership of the car. This latter operation is performed by creating, from the companion app, a private-public key that can be given to the buyer of the second-hand car. In each moment, the previous history of a car can be accessed by the application. The work described in [26] presents a system to record on the blockchain events that occurred during the vehicle's life, which are used to compute a rating on the health of the vehicle. In particular, the system acquires data from the OBD-II port. These data are used to compute a driver behavior rate, based on previously occurred harsh events. Engine-related information is also recorded on the blockchain to compute a car maintenance rating (based mainly on engine coolant and engine oil temperature). Odometer information is also acquired and stored on the blockchain. All the information is recorded in a smart contract, where a mapping links the VIN to the car's parameters. The work of [27] describes a system relying on the Quorum blockchain to record relevant information related to a car. In particular, when the customer buys the car, the manufacturer creates a digital car book on the blockchain. The access to the car book is initially restricted only to the owner and to the manufacturer, nonetheless, additional grants can be given to other actors (e.g., insurance companies). During the vehicle's life, sensors acquire data related to its usage and periodically store them in the car book. Maintenance operations as well as insurance-related events are also recorded in the car book. When the car is sold, the ownership of the car book is also transferred. The authors of [28] present a system that stores information about the production, the sub-productions, transportation, provenance, and quality of a car on the blockchain. The system can be exploited by different organizations and by the customer. In particular, the blockchain is used to store the hash of quality assurance reports, together with sensors' data gathered during the transportation of the car to the dealers. In this way, the customer can inspect the blockchain to retrieve detailed information on the manufacturing and shipping events related to the car. In [29], a system to trace used cars from the moment in which they enter the market to the instant in which they are sold is proposed. The system relies on Hyperledger Fabric and is meant to be used by an alliance of manufacturers, logistic enterprises, testing institutions, used car dealers and customers. A similar idea is proposed in [30], where a consortium blockchain based on Hyperledger Fabric is defined to ease the interactions among used car buyers and sellers, banks, insurance companies and used car dealers. Information stored on the blockchain is related to test reports, insurance-related data, and certificates such as proof of purchase of insurance certificates. In addition, the system includes an automatic evaluation component, which estimates an appropriate price for the second-hand car. In [31], a system to track used cars' data, relying on the Ethereum blockchain is presented. The system combines smart contracts and decentralized storage (the InterPlanetary File System – IPFS [32]) to store relevant data. In particular, IPFS is used to store documents, whereas the smart contract only stores the hash of each document. The work described in [33] specifically focuses on the bidding process. In particular, it proposes a system for an e-auction methodology on top of the Ethereum blockchain. Documents related to a second-hand car are stored on IPFS, and their hash is recorded on the smart contract. In order to sell a car, an inspector first needs to certify its state. Then, the bidding process is managed by the smart contract, which collects the bids from potential buyers. Finally, the work described in [34] specifically considers the market of second-hand electric vehicles. In fact, the battery of pre-used electric vehicles could be affected by the electric conveyance history, charging capabilities of the battery, history of the charging records, and performance of the driver. Hence, the authors of this work propose to exploit the blockchain to let electric vehicle manufacturers, charging stations, and battery manufacturers record relevant events that occurred during the vehicle's life.

The above overview, which has been summarized in Table 1 highlights that several attempts have already been made in research, to propose blockchain-based solutions for the

Ref.	Focus on	(Blockchain) Tech.	NFTs
[20]	car previous history	N/A	No
[21]	car previous history	Graphene	No
[22]	car previous history	Ethereum	No
[23]	car previous history	Ethereum	No
[24]	used cars	Ethereum, NoSQL DB	No
[25]	used cars	BigchainDb	No
[26]	used cars	Ethereum	No
[27]	used cars	Quorum	No
[28]	car supply chain	Hyperledger Fabric	No
[29]	used cars	Hyperledger Fabric	No
[30]	used cars	Hyperledger Fabric, IPFS	No
[3]	used cars	N/A	No
[31]	used cars	Ethereum, IPFS	No
[33]	used cars	Ethereum, IPFS	No
[34]	used electric cars	Ethereum	No

TABLE 1. Overview of related works

Ref.	NFT type	Locking	Trading system	Payment	Dispute manag.
[12]	static	Yes	Yes	Cryptocurr.	No
[13]	static	No	Yes	Cryptocurr.	Yes
[14]	static	No	Yes	Cryptocurr.	Yes
[15]	static	No	Yes	Cryptocurr.	No
[16]	static	No	Yes	Cryptocurr.	No
[17]	dynamic	No	Yes	Cryptocurr.	No
[18]	static	No	Yes	Cryptocurr.	Yes
Our	**dynamic**	**Yes**	**Yes**	**Traditional**	**Yes**

TABLE 2. Comparison with other works based on NFTs

traceability of a vehicle's previous history, or for the certification of the vehicle's state in the second-hand car market. The majority of the developed systems exploit the Ethereum blockchain (in some cases, with a decentralized storage like IPFS, or with a centralized one). Some systems, instead, rely on Hyperledger Fabric for permissioned access.

The majority of the analyzed works have the objective to store the vehicle's story from multiple sources [20]–[24], [28]–[31], [34]. Some works also propose to rely on the blockchain to transfer the property of the car [25], [27]. One work takes a step forward and uses the information recorded on the blockchain to automatically compute the status of a vehicle [26]. Finally, one work further exploits the decentralized nature of the blockchain, to create a decentralized bidding platform [33].

The aim of our work is a summarization of the objectives just reported: i.e., we aim to store a vehicle's previous history, but at the same time to create a decentralized market in which users can bid for a second-hand vehicle (eventually, by having an indication on the right price for the vehicle), and in which the ownership of a vehicle can be automatically transferred.

In particular, for the latter two objectives – decentralized market and transfer of ownership – we rely on NFTs to ease the bidding phase and the trading of the vehicle. To the best of our knowledge, no other works already investigated the practical application of NFTs to second-hand vehicle trading, by proposing an architecture and a working solution.

We summarize our key contributions as follows:

- we aim to store a vehicle's previous history, but at the same time to create a decentralized market in which users can bid for a second-hand vehicle (eventually, by having an indication on the right price for the vehicle), and in which the ownership of a vehicle can be automatically transferred;
- we use NFTs to facilitate the handling of vehicle processes, i.e., tracking history, updating vehicle information, and the trading phase;
- to keep the history of the vehicle updated, we propose a dynamic NFT that contains updated hashes of the data stored on the IPFS;
- to protect the negotiation phase from fraud, the ownership of the NFT is first transferred to the smart contract of the marketplace, which automatically transfers it to the buyer if he/she correctly completes the purchase, otherwise, it returns it to the seller.

Table 2 shows a comparison between our solution and several NFT-based solutions already reported in the literature [12]–[18]. In particular, we have compared the NFT architecture and the trading system. First, our solution is based on a dynamic NFT structure where some data can be updated, such as the status of the NFT and the IPFS hashes. A similar approach was implemented in [17], whereas all other solutions developed a static NFT structure that cannot be modified once it is created. In addition, our solution implements a "locking" system that prevents certain actions on the NFT (e.g., transferring ownership) while its status is "locked". This feature guarantees that no malicious/unintended actions can be performed during the trading phase. Most of the other solutions do not have this kind of protection, except for [12], which implements a "locking" system that is checked only on its transfer function instead of the whole trading process. Regarding the trading system, all the solutions mentioned have implemented a smart contract to manage the transfer of NFTs. Our solution is the only one based on traditional currencies. We decided not to rely on cryptocurrencies because trading vehicles require certain amounts of money that most people would not trust to convert into cryptocurrencies right now. Finally, we proposed a dispute management feature based on transferring the ownership of an NFT during the trading phase to the marketplace smart contract, which automatically transfers it to the buyer or seller depending on the purchase outcome. Some of the other solutions, such as [13], [14], [18], developed dispute management based on security deposits.

It is worth remarking that, apart from NFTs, other state-of-the-art techniques in the field of blockchain and distributed ledger technologies have been devised, so far. To mention a few of them, together with permissionless blockchains (e.g., Bitcoin, Ethereum, etc.) and permissioned blockchains (e.g., Hyperledger fabric [35]), ledger databases (e.g., LedgerDB [36]), Directed Acyclic Graphs (e.g, IOTA [37]), Hashgraphs (e.g., Hedera [38]), Holochain (e.g., Holochain [39]) and Tempo (e.g., Radix [40]) have been devised. Some of

the above alternatives provide full decentralization (Bitcoin, Ethereum, IOTA, Holochain, Radix), whereas others were devised to/resulted in centralized solutions (Hyperledger Fabric, LedgerDB, Hedera). In order to develop our architecture, we evaluated the pros and cons of each alternative. In particular, we decided that, although solutions based on permissioned blockchains and ledger databases offer advantages such as low storage costs and high throughput, we needed the benefits of a fully decentralized system. In fact, our system had to guarantee to users the immutability of the stored data, without the possibility of a central entity arbitrarily changing it. In addition, it had to be publicly accessible to anyone who wanted to examine vehicle details. Among decentralized systems, we selected those potentially supporting NFTs (Bitcoin, Ethereum, Holochain, Radix). Finally, among them, we decided to rely on the Ethereum blockchain, due to its widespread adoption, its community, the number of successfully NFT-based projects and platforms (such as Opensea[1] or LooksRare[2]), and the availability of well-tested and optimized libraries/standards to reduce costs and attacks.

III. PROPOSED SOLUTION

In this Section, we describe our NFT-based solution by starting with a general overview and then going into the system design details, such as the architecture, the stakeholders involved, and the sequence diagrams that outline the interactions with the platform.

As previously mentioned, the proposed framework operates on the Ethereum blockchain, which provides the ability to execute smart contracts and consequently leverage NFTs for tracking the vehicle's lifecycle and change of ownership. Ethereum supports the standard ERC-721, an open standard that describes how to build non-fungible or unique tokens. This standard defines a minimum interface that a smart contract must implement to allow unique tokens to be managed, owned, and traded.

The proposed solution benefits from the built-in advantages of NFTs and blockchains, such as security, reliability, traceability, transparency, and proof of ownership. In addition, it exploits IPFS as a decentralized storage system. IPFS grants the availability and immutability of the vehicles' data uploaded through the platform.

A. GENERAL SYSTEM OVERVIEW

The developed system aims to provide Original Equipment Manufacturers (OEMs) with a secure and reliable system through which their customers can sell and purchase vehicles in the second-hand market by leveraging NFTs. In particular, NFTs track the life cycle of vehicles by storing up-to-date information about them and their condition (manufacturing metadata, repair works, insurance certificates, and inspection reports).

[1]https://opensea.io
[2]https://looksrare.org

The core of our framework resides within two smart contracts developed to provide functions for the NFTs management system and the second-hand market operations. Figure 1[3] illustrates the overall system architecture, consisting of the back-end that includes the two smart contracts (i.e., *the NFT smart contract* and the *Martketplace smart contract*) deployed on the Ethereum blockchain as well as the IPFS storage, and the front-end Dapp through which actors interact with the other components. The Figure also shows the actors involved in the system processes, i.e., the OEM, repair shops, insurance companies, buyers, sellers, and the three external modules that provide support functionalities, i.e., Metamask, the market price evaluator, and the payment gateway.

The first smart contract is the NFTs smart contract developed according to the Ethereum ERC-721 standard and includes functions for minting, transfer approvals, and transfer of NFTs. In addition to ERC-721 functions, the smart contract allows updating vehicle metadata and managing role-based access control. The second smart contract is the Marketplace smart contract, which contains functions that handle the listing and purchasing processes of NFTs. Through it, buyers can bargain NFT prices set by sellers if they differ from those established by the market price evaluation.

The front-end Dapp serves as the interface for users to interact with the back-end and the external modules. Users must authenticate themselves through the register/login phase before interacting with the application. Moreover, operations involving sending and receiving transactions require users to have an Ethereum address with public-private key pair used to sign and verify transactions on the blockchain. However, browsing the NFTs for sale does not require an account.

The decentralized storage contains vehicle metadata, such as images, properties, and updates. It ensures the immutability of uploaded data and preserves the availability of all data as long as at least one node in the network has a copy of it.

B. SYSTEM DESIGN

The system architecture, depicted in Figure 1, includes different actors, components, and external modules that interact with each other. The role of each actor/component/external module is described below:

Actors:

- **OEM**: OEMs produce the vehicles. When a new vehicle is produced, the OEM mints a new NFT containing a graphical representation of the physical object and production metadata (both stored on IPFS). When a customer wants to buy a new vehicle, the OEM transfers the ownership of the associated NFT to the customer. In addition, the OEM accepts or rejects updates to a vehicle and may request updates himself.
- **Insurance companies**: insurance companies release a new insurance certificate when the vehicle owners renew the policy. They upload the certificates to IPFS and

[3]Figures in the paper have been designed using images from Flaticon.com

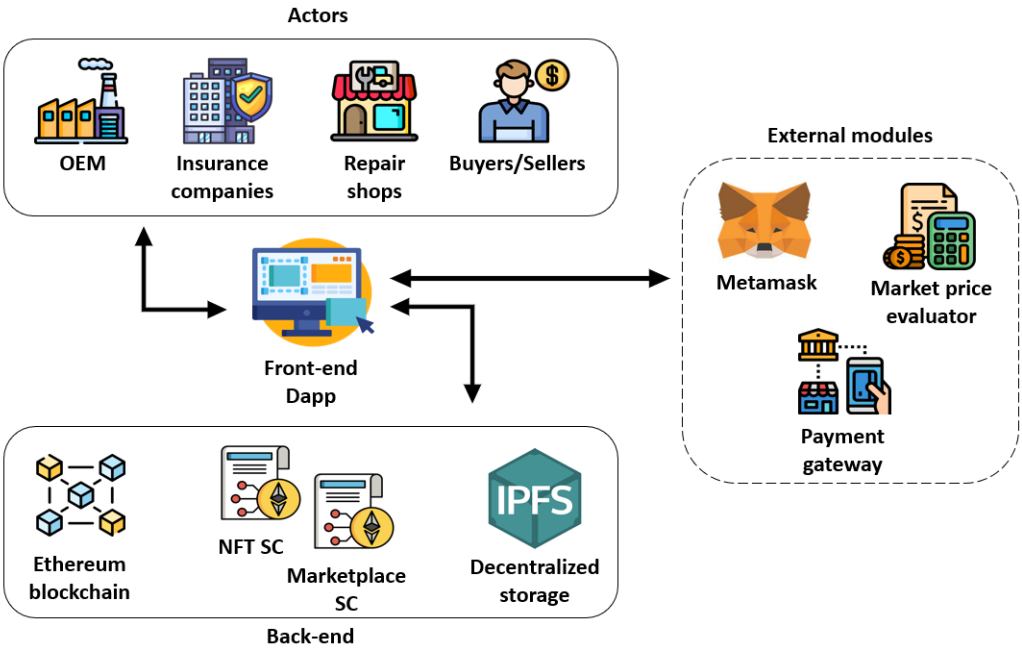

FIGURE 1. System architecture

update the NFTs. In addition, insurance companies can change the state of NFTs to "insurance required", if needed.

- **Repair shops**: repair shops perform maintenance operations on vehicles. They add their work to the repair log files, upload them to IPFS and update the NFTs. Repair shops can change the state of NFTs to "maintenance required", if needed.
- **Buyers**: buyers submit a price proposal to the NFTs they wish to buy. If the proposal is accepted, they execute the payment and receive the ownership of the NFT bought.
- **Sellers**: sellers offer their NFT (thus including the associated vehicle) for sale in the marketplace. They can accept or reject price offers made by interested buyers or remove their listed NFT at any time before the sale.

Components:

- **Ethereum blockchain**: The Ethereum blockchain is a decentralized platform that allows users to execute code and transactions in a decentralized manner. Changes to the blockchain are initiated by transactions and recorded permanently. In this system, the Ethereum blockchain stores smart contracts, NFTs, and all associated transactions.
- **NFT smart contract**: The NFT smart contract inherits the standard ERC-721, which outlines the must-have functions to manage NFTs, and implements the logic for updating the NFTs when the associated vehicles undergo updates. Along with the Marketplace smart contract and IPFS, this smart contract forms the back-end portion of the system.
- **Marketplace smart contract**: the used vehicle market resides in a second smart contract that provides the functions for listing and selling NFTs. The Marketplace smart contract collects the offers made by buyers and provides the capability to accept or reject them to sellers. Along with the NFT's smart contract and IPFS, this smart contract forms the back-end portion of the system.
- **IPFS**: IPFS is used to store files in a decentralized way. Files stored on IPFS can be located using the hash of their content (i.e., their Uniform Resource Identifiers, or URIs) and are immutable, meaning they cannot be changed. They are also always accessible as long as at least one node in the network holds a copy of the file. In the case of NFTs, IPFS is used to store the images of a vehicle and the metadata associated with it.
- **Dapp**: the decentralized application provides the interface that allows users to interact with smart contracts. It implements the authentication procedure with the traditional registration/login process, but many functionalities also require Ethereum address authentication. Users can consult their profile page to interact with the NFTs or the marketplace homepage to purchase/sell them.

External modules:

- **Market price evaluator**: the market price evaluator is an external module that automatically computes the market price for a vehicle by taking into consideration the metadata of the associated NFT.
- **Payment gateway**: the payment gateway is an external module on which the system relies to finalize payments. It allows users to pay for products with traditional currencies, which are much more used than cryptocurrencies. The Dapp redirects users to the payment gateway

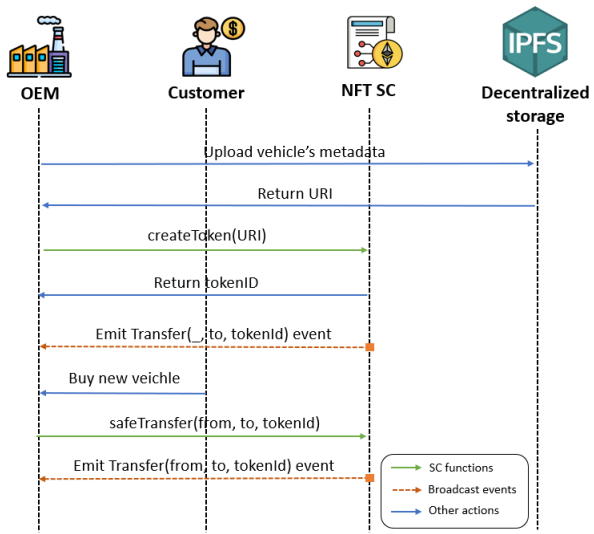

FIGURE 2. Minting process sequence diagram

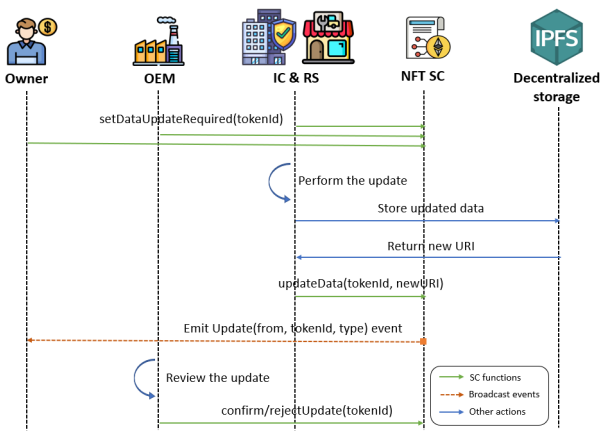

FIGURE 3. Updating process sequence diagram

when an offer is accepted and the winner proceeds to the purchase.

- **Metamask**: MetaMask [41] is an extension for accessing Ethereum-enabled distributed applications. It lets the users create and manage their own identities, so when the Dapp needs to perform a transaction and write data to the blockchain, the user gets a secure interface to review the transaction, before approving or rejecting it.

C. SEQUENCE DIAGRAMS

Sequence diagrams describe the interactions between actors and system components. These interactions require the OEM to first deploy the NFT smart contract and the Marketplace smart contract on the blockchain as a preliminary step. At this purpose, it is worth highlighting that, even though we tested our application with a single OEM, the devised system could be potentially used by several different OEMs, by letting each OEM deploy its smart contracts on the blockchain. Additionally, as previously mentioned, all actors participating in the subsequent interactions must have an account on the platform and an Ethereum address.

The first set of interactions, on which all others depend, is the minting process of a new NFT, shown in Figure 2. The OEM produces a new vehicle and, through the Dapp, uploads the image and the manufacturing metadata to the decentralized storage. Then, the Dapp retrieves the URI, along with the OEM's address, and prepares a transaction to be signed by the OEM. Once signed, this transaction is sent to the NFT smart contract, which mints the new NFT and generates an event to notify the system of a new minting. When a customer buys a new vehicle, the OEM transfers the ownership of the associated NFT from its address to the customer's address.

Figure 3 illustrates the interactions that occur when a vehicle needs an update. First, different actors, such as the vehicle's owner, the repair shop, the insurance company, or even the OEM, can request an update by changing the NFT's state through the Dapp. The *updateRequired* state means that the associated vehicle needs some operation, such as renewal of the insurance certificate, renewal of the inspection report, repair, or dismission. Then, the authorized actor performs the update and uploads the updated metadata to the decentralized storage. The Dapp receives the new URI (which points to the new metadata) and sends it to the NFT smart contract, which updates the NFT, changes its state to *underReview*, and issues an event to inform that the NFT has been updated. The OEM then checks the NFT and approves or rejects the update by changing its state to *Reviewed* or *updateRequired*, respectively. In this implementation, it is assumed that the OEM can guarantee that updates have been performed correctly. In reality, it would be possible to manage different cases in which the "approvers" could be several players or, in some cases, the smart contracts themselves contain the proper logics and data that confirm the inserted information (e.g. in the case of connected vehicles).

Figure 4 depicts the listing and purchasing vehicle processes on the secondary market. Users who wish to list their vehicles in the marketplace can request the market price evaluation to the market price evaluator (MPE) module. The MPE retrieves vehicle metadata from the decentralized storage using the URIs provided by the NFT smart contract and, based on them, computes the market price by exploiting such data. For example, a newer vehicle with fewer repairs will receive a higher price evaluation than an older vehicle with more repairs. Concerning NFTs listing on the Marketplace, the ERC-721 standard mandates that in order to allow an address (other than the owner itself) to transfer an NFT the owner must authorize it. Since the Marketplace smart contract directly transfers NFTs once they are sold, the sellers must authorize the smart contract's address to transfer their NFTs before listing them. Next, users set their price (knowing the market price) and list their NFTs for sale. When the NFT is listed, the system checks if it has an expired insurance certificate

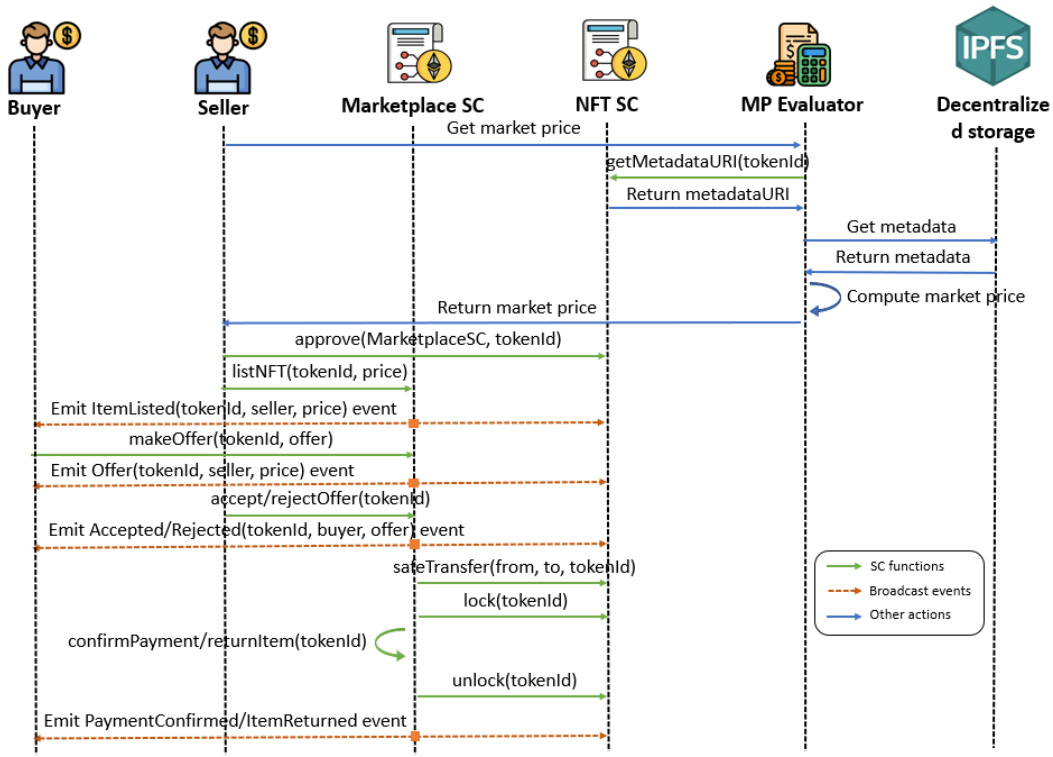

FIGURE 4. Listing/purchasing process sequence diagram

or inspection report and eventually updates the relative states of the NFT to *updateRequired*. Buyers may browse available NFTs for sale and submit an offer for any vehicle they wish to purchase. The system will automatically reject bids lower than a threshold t, where t is the minimum value among the market price, the price specified by the seller, or a previous offer. Sellers can choose to accept or reject the remaining offers. When an offer is accepted, the system notifies the winner, who will complete the purchase with the payment, and emits an event that notifies actors of the sale. At this time, the NFT state is *Locked* and the Marketplace smart contract holds the ownership of the NFT until the payment deadline. If the winner proceeds with the payment (the payment is executed through the external gateway), the system unlocks the NFT and transfers the NFT's ownership to the winner; if not, it returns the ownership to the seller. It is worth remarking that the verification of the physical delivery of the vehicle is out of the scope of this work.

IV. IMPLEMENTATION DETAILS

In this Section, we provide an in-depth explanation of the system, in particular, of the structure of the proposed NFT, the functions and the algorithms behind the diagrams described in Section III, and the core pages of the front-end component. Our smart contracts are developed and tested in Solidity with the online Remix IDE [42]. Figure 5 illustrates the NFT smart contract and the Marketplace smart contract, along with all the functions and attributes we have implemented. As previously mentioned, the NFT smart contract is devoted to storing information on the vehicle's state and history, whereas the Marketplace smart contract records the data related to the offers made by the buyers. In particular, the NFT smart contract stores one or more Ethereum addresses allowed to call the update functions and a list of *Vehicle* structs (one for each NFT minted) that includes additional data to the standard ERC-721 data. It is worth highlighting that Figure 5 does not depict the functions inherited from the AccessControl smart contract [43] and the ERC721URIStorage smart contract [44] (which itself inherits from the ERC721 smart contract [19]) because we have imported them from the OpenZeppelin library. In our implementation, the Marketplace smart contract interacts with only one NFT smart contract, even though in the future it could be linked to several NFT smart contracts (more details on this aspect are provided in Section VI). It provides functions to list and buy NFTs, make offers, and accept or reject them. For each NFT listed, the Marketplace smart contract saves an *Item* struct that includes the seller, the price, the current offer, the buyer, the payment deadline, and the state.

The front-end part of the system has been developed in JavaScript using React [45] library for the user interface, Web3 [46] library to interact with smart contracts and thus also with the blockchain, and JS-IPFS [47] library to run and connect with an IPFS node. Figure 6 depicts the main pages of the user interface, such as the account pages (one for each user type), the evaluator page, the list of vehicles for sale, and

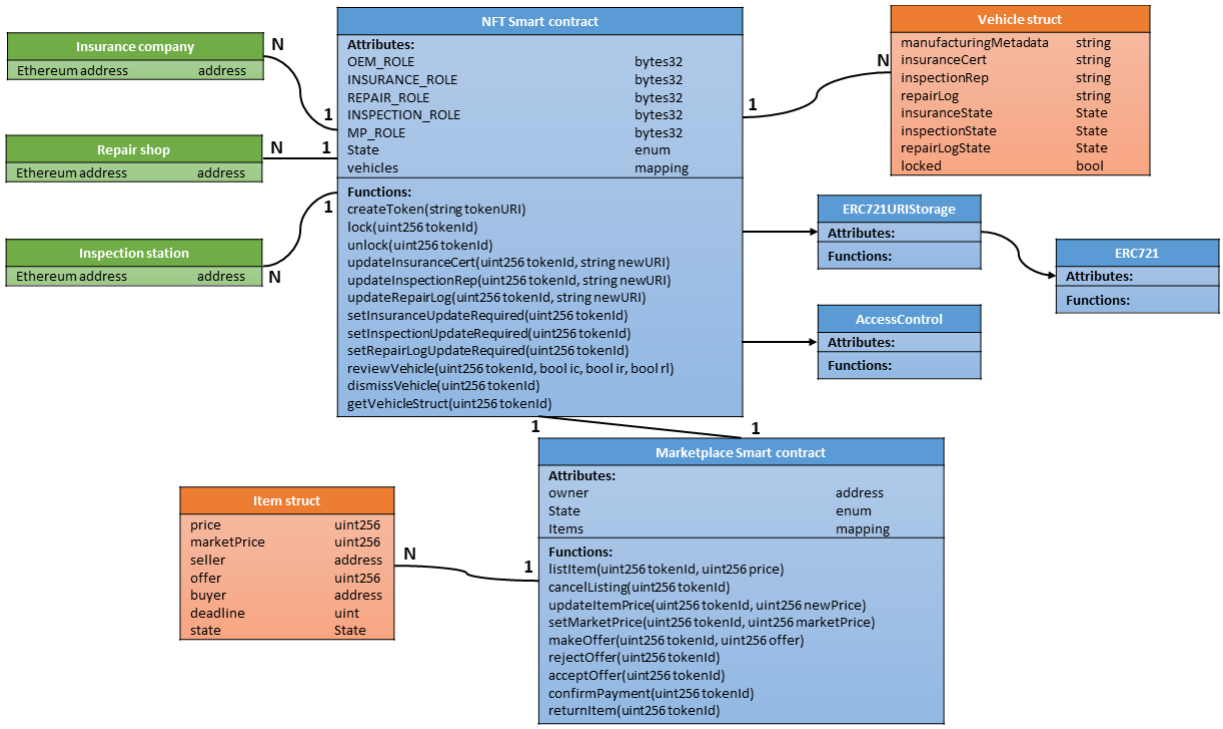

FIGURE 5. Smart contracts diagram

the vehicle page.

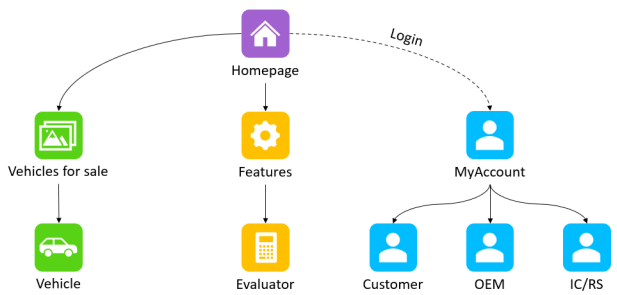

FIGURE 6. Site map

A. NFT STRUCTURE

In order to track vehicle's data, our solution proposes an enriched version of the standard NFT. In addition to the base components, such as an incremental unique identification number, the owner's address, and a URI that links the NFT to its metadata, we add a struct that tracks three additional metadata types. Specifically, we have the following types:

- **Manufacturing metadata**: manufacturing metadata are immutable and include a graphic representation (also saved on decentralized storage), the vehicle identification number, the production date, type, make, model, and so on. When the OEM mints the NFT, it also provides manufacturing metadata.
- **Insurance Certificate**: the Insurance Certificate is a document issued by an insurance company and contains details about the active policy and the insured vehicle. The insurance company updates the certificate once the vehicle's owner purchases/renews the insurance policy.
- **Vehicle Inspection Report**: the Vehicle Inspection Report (VIR) is a report issued by an inspector or inspection station, i.e., authorized repair shops, that indicates whether the vehicle has passed the required safety and emission tests. The inspection station periodically conducts the required tests and updates the VIR accordingly.
- **Repair log**: the repair log is a record that documents all the repairs performed on the vehicle. Each entry in the log includes the date of the repair, a detailed description of the work carried out, and the repair shop or technician who performed the repair. Each time a new repair is made, the log is updated.

Our NFT structure also includes three state variables, one for each updatable data item, which indicates when the vehicle needs an action. There are four different states for each variable, i.e., *Reviewed*, *UpdateRequired*, *UnderReview*, and *Dismissed*. When the OEM mints the NFT, all the three variables are set as *Reviewed*. If some action is required, the respective variable changes its value to *UpdateRequired*. After the update, the variable changes its value to *UnderReview* because it needs to be reviewed by the OEM. Finally, the OEM sets the variable to *Reviewed* or *UpdateRequired* if it confirms or denies the update, respectively. All the three

variables take on the value *Dismissed* when the vehicle stops circulating. The *locked* field of the struct is a boolean value that prevents the NFT transfer when it is set to *True*.

B. ALGORITHMS

In the following, the algorithms devised for the functioning of the system are described and discussed. Algorithm 1 shows the steps behind minting a new NFT. The OEM produces a new vehicle and uploads its manufacturing metadata to IPFS. Then, it calls the *createToken* function (callable only by the NFT smart contract's owner) that takes the IPFS URI as input. Internally, *createToken* generates a new *tokenId* and calls the inherited functions *_safeMint* and *_setTokenURI*, which assign id, owner, and URI to the new NFT. Next, the function associates with the NFT a *Vehicle* struct and initializes it with default values. Finally, a *Transfer* event is emitted to notify the system of the new token.

Algorithm 1: Minting process algorithm

Input: URI, OEM
URI: IPFS URI for manufacturing metadata;
$caller$: who calls the function;
OEM: OEM's Ethereum address;
call: $createToken(URI)$
if $caller\ is\ OEM$ **then**
  $tokenId$ = $tokenIdCounter$++;
  call: $\_safeMint(caller, tokenId)$
  map: $tokenId \rightarrow caller$
  call: $\_setTokenURI(tokenId, URI)$
  map: $tokenId \rightarrow URI$
  map: $tokenId \rightarrow Vehicle$
  emit: $Transfer(\_, caller, tokenId)$
else
  throw $Error$;
end
Result: $tokenId$

call indicates that a function is executed, **map** stores the data in the form of key-value pairs where the key is the left member and value the right one, **emit** indicates the issuing of an event

Algorithm 2 describes how vehicles' NFTs are updated. First, when the OEM deploys the NFT smart contract, it must assign access roles to the addresses that will perform the updates. Such roles prevent the execution of functions from unauthorized addresses. As shown in Figure 5, there are five roles, i.e., *OEM_ROLE*, *INSURANCE_ROLE*, *REPAIR_ROLE*, *INSPECTION_ROLE*, and *MP_ROLE*. The OEM assigns roles by calling the *grantRole* function, inherited from AccessControl smart contract. Either the OEM, the vehicle's owner, or one of the authorized actors can signal the need for an update by calling *setInsuranceUpdateRequired*, *setInspectionUpdateRequired*, or *setReparilLogUpdateRequired* functions, which change the NFT's state to *UpdateRequired*. Insurance companies, repair shops, or the OEM perform the updates required and upload the updated data to IPFS. Then, they call the update functions by passing

Algorithm 2: Updating process algorithm

Input: $newURI, IC, RS, caller, OEM, O$
$newURI$: new IPFS URI for updated metadata;
IC: insurance company's address;
RS: repair shop's address;
$caller$: who calls the function;
OEM: OEM's address;
O: NFT owner's address;
call: $setDataUpdateRequired(tokenId)$
if $caller\ is\ O$ || IC || RS || OEM **then**
  $Vehicle.state$ = $UpdateRequired$;
else
  throw $error$;
end
call: $updateData(tokenId, newURI)$
if $caller\ is\ IC$ || RS || OEM && $Vehicle.state$ == $UpdateRequired$ **then**
  $Vehicle.dataURI$ = $newURI$;
  $Vehicle.state$ = $UnderReview$;
  emit: $Update(tokenId, caller, type)$
else
  throw $error$;
end
call: $reviewVehicle(bool, tokenId)$
if $caller\ is\ OEM$ **then**
  if $bool\ is\ False$ && $Vehicle.state$ == $UnderReview$ **then**
    $Vehicle.state$ = $UpdateRequired$;
  else
    $Vehicle.state$ = $Reviewed$;
  end
else
  throw $error$;
end

call indicates that a function is executed, **map** stores the data in the form of key-value pairs where the key is the left member and value the right one, **emit** indicates the issuing of an event

the new IPFS URI and the *tokenId*. Such functions are *updateInsuranceCert*, *updateInspectionRep*, and *updateRepairLog*, which save the new IPFS URI, change the NFT's state to *UnderReview* and emit an *Update* event to notify the system of the update. The *Update* event includes the NFT's *tokenId*, the actor who performed the update, and the type. Finally, the OEM verifies that the updates under review have been performed correctly and calls the function *reviewVehicle* by passing three boolean values for the three types of updates. *False* value sets the relative state to *UpdateRequired* if it is *UnderReview*, whereas *True* value sets the state to *Reviewed*.

Algorithms 3 and 4 describe the listing and purchasing processes in detail. Users can sell their vehicles by listing them in the Marketplace smart contract. For this purpose, the *listItem* function has been created, which adds a new *Item* to the list of vehicles on sale. An *Item* includes the NFT's id to which it is associated, the seller's address, the price

Algorithm 3: Listing process algorithm

Input: $caller$, $seller$, O, $notListed$, SCO
$caller$: who calls the function;
$seller$: seller's address;
O: NFT owner's address;
$notListed$: NFT not listed yet;
SCO: who deployed the smart contract;
call: $listItem(tokenId, price)$
if $caller$ *is* O *&&* $tokenId$ *is* $notListed$ **then**
  map: $tokenId \rightarrow Item(seller, tokenId, price, Open)$
  emit: $ItemListed(tokenId, seller, price)$
else
  throw $error$;
end

call indicates that a function is executed, **map** stores the data in the form of key-value pairs where the key is the left member and value the right one, **emit** indicates the issuing of an event

set by the seller, the market price obtained from the MPE module, its selling status (open to bids or not), the (potential) buyer's address, the (potential highest) buyer's offer, and the deadline by which the buyer must complete the payment. The *listItem* function first requires the seller to enable the Marketplace smart contract to transfer its NFT once it is sold, and checks whether the caller is the owner of the NFT it wants to list. Then, it initializes a new *Item* with the seller's address and desired price and sets the state to *Open*. Finally, the function emits a *ItemListed* event. Buyers that wish to buy a vehicle make their offers by calling the *makeOffer* function, which takes as input the NFT's id and the offer's price. This function checks if the offered price is higher than a threshold *t*, which is the minimum value among the market price, the price specified by the seller, or an existing offer (if present) and sets the buyer and offer fields of the *Item*. Sellers can reject or accept the bids by calling *rejectOffer* or *acceptOffer*, respectively. The *rejectOffer* function deletes the offer and emits a *OfferRejected* event. The *acceptOffer* function emits a *OfferAccepted* event, changes the state from *Open* to *Close*, transfers the ownership of the NFT from the seller to Marketplace smart contract, sets a seven days deadline for the payment, and locks the NFT transferability as far as the buyer completes the payment. If the buyer pays within the deadline, the *confirmPayment* function unlocks the NFT, transfers the NFT's ownership from the Marketplace smart contract to the buyer, and removes the item from the list of items of the marketplace. Otherwise, *returnItem* transfers the NFT back to the seller and resets the item struct as open to bids. The events *PaymentConfirmed* and *ItemReturned* notify the system.

C. FRONT-END DETAILS

As shown in the sitemap in Figure 6, currently, our application is composed of the homepage through which users can navigate to the registration/login page, the evaluator page,

Algorithm 4: Purchasing process algorithm

Input: $buyer$, $seller$, O, $caller$
$buyer$: buyer's address;
$seller$: seller's address;
O: NFT owner's address;
$caller$: who calls the function;
call: $makeOffer(tokendId)$
if $offer > price$ || $marketPrice$ || $Item.offer$ *&&* $Item.state$ *not* $Closed$ **then**
  $Item.offer = offer$;
  $Item.buyer = buyer$;
  emit: $Offer(tokenId, buyer, offer)$
else
  call: $rejectOffer(tokenId)$
end
if $rejectOffer(tokenId)$ **then**
  delete: $item.offer$
  delete: $item.buyer$
  emit: $OfferRejected(tokenId, buyer, offer)$
end
else if $acceptOffer(tokenId)$ **then**
  $Item.state = Close$;
  $Item.deadline = now + 7days$;
  call: $safeTransfer(tokenId)$ $seller \rightarrow buyer$
  call: $lock(tokenId)$
  emit: $OfferAccepted(tokenId, buyer, offer)$
end
if $payment\_date < Item.deadline$ **then**
  call: $confirmPayment(tokenId)$
  emit: $PaymentConfirmed(tokenId, buyer, seller)$
else
  call: $returnItem(tokenId)$
  emit: $ItemReturned(tokenId, seller, buyer)$
end

call indicates that a function is executed, **map** stores the data in the form of key-value pairs where the key is the left member and value the right one, **emit** indicates the issuing of an event

and the vehicles for sale page. First, the application provides the possibility to register for four types of users, i.e., OEMs, customers, insurance companies, and repair shops. For each of them, there is a different account page described below:

- **Customers**: the customer page allows users to monitor and manage owned vehicles, active listings, and offers. It includes five tabs: owned vehicles, active listings, under update, offers made, and offers received.
- **OEMs**: the OEMs' page allows the OEM users to generate new NFTs when new vehicles are produced, monitor all the produced vehicles, and review updates. It includes three tabs: produced vehicles, review updates, and minting NFTs.
- **Insurance companies**: the page allows authorized insurance companies to check their insurance certificates, notify vehicle owners of expired certificates, and update them upon renewal. It includes three tabs: insured vehicles, expired certificates, and upload certificates.

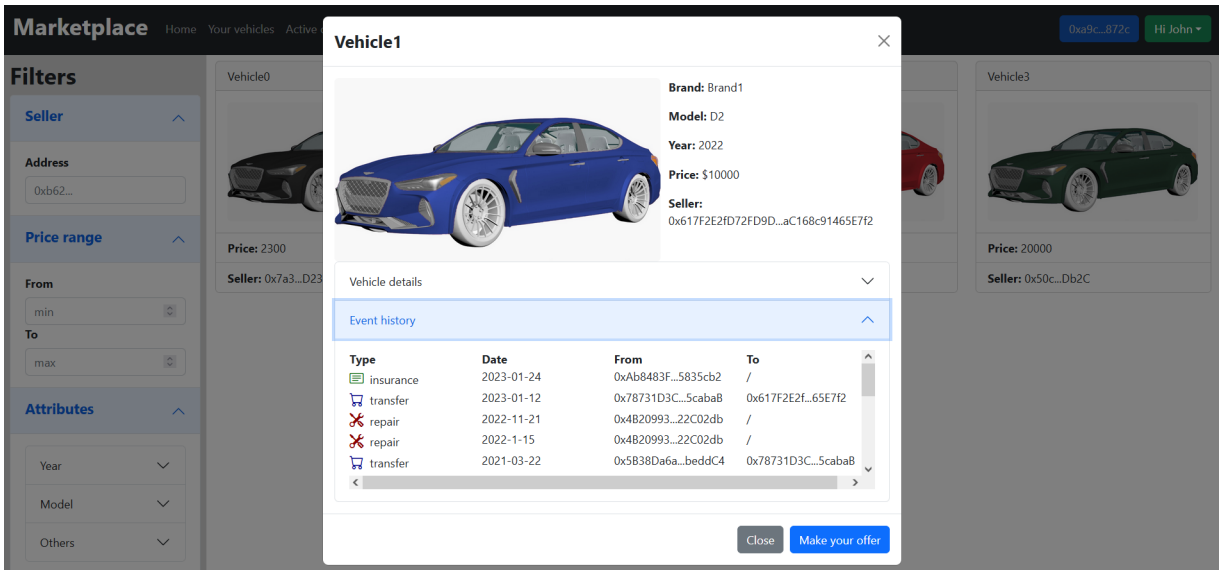

FIGURE 7. Vehicle details page

- **Repair shops**: the page allows authorized repair shops to register maintenance works and inspection reports for vehicles they worked on. It includes four tabs: under maintenance, expired inspections, upload inspection reports, and update repair log.

The evaluator page connects the application with the external evaluator module. It includes a form through which users can query the evaluator to obtain the market price of their vehicles. The evaluator returns a report containing how it calculates the market price and what elements it considers. It is worth mentioning that a detailed description of how the evaluator works is out of the scope of this paper.

Finally, the vehicles for sale page is where customers can scroll all the listed vehicles and place bids for the ones they want to purchase. This page, shown in Figure 7, includes the possibility to filter elements according to price range, sellers' addresses, and vehicle attributes (model, production year, etc.). Each item on this page shows the graphical representation of the vehicle, the price, and the address of the seller. By clicking on a vehicle, a window appears showing the users the vehicle's main information, a list of all vehicle details, and the vehicle's event history. Each item in the event history list is clickable and shows the user the details of the event. At the bottom of the window, the "make your bid" button allows users to make their bids.

V. TESTING

This Section describes the procedure followed to test the two smart contracts. All the functions contained in the smart contracts were tested to verify whether the implemented rules and functionalities were violated or not. In addition, we implemented several modifiers to prevent unauthorized executions, and we included in the code the triggering of events to track and notify the system about occurred transactions. Table 3 shows the Ethereum addresses of the actors and the deployed smart contracts, that we used during testing. We ran all the tests using Remix IDE and showed the results of the main functions, such as minting a new NFT, updating an NFT, and selling/purchasing an NFT in the rest of the Section.

TABLE 3. Ethereum addresses

Name	Ethereum address
OEM	0x5B38Da6a701c568...FcB875f56beddC4
Insurance company	0xAb8483F64d9C6d1...677dD3315835cb2
Repair shop	0x4B20993Bc481177...eCaE8A9e22C02db
Customer1	0x78731D3Ca6b7E34...a7cC18A495cabaB
Customer2	0x617F2E2fD72FD9D...aC168c91465E7f2
Customer3	0x17F6AD8Ef982297...C1DbfFE4348c372
NFT SC	0xd9145CCE52D386f...eB44e9943F39138
Marketplace SC	0xd8b934580FCE35a...aDeE468a2833fA8

A. MINTING NEW NFT

When the OEM produces a new vehicle, it creates a new NFT through the NFT smart contract. First, the OEM uploads the manufacturing metadata to IPFS, then calls the *createToken* function by passing the IPFS hash (i.e., its URI) as a parameter. This function generates a new NFT, transfers its ownership to the OEM, and emits a new *Transfer* event after its execution that notifies the system of a new NFT. Figure 8 shows the output log of the *createToken* function. The log includes the input parameter, the *from* field which describes from which smart contract the log is, and the Transfer event with its *args*. The *onlyRole(OEM_ROLE)* modifier inherited from AccessControl.sol restricts the ability to execute the function to the OEM only.

```
{
        "string uri": "https://ipfs.io/ipfs/QmbWqxBEKC3P8tqsKc97xmWNzrzDtRLMiMPL8wBuTGsMnR"
}

{}

[
        {
                "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                "topic": "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
                "event": "Transfer",
                "args": {
                        "0": "0x0000000000000000000000000000000000000000",
                        "1": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                        "2": "1",
                        "from": "0x0000000000000000000000000000000000000000",
                        "to": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                        "tokenId": "1"
                }
        }
]
```


FIGURE 8. createToken() transaction output


```
{
        "uint256 tokenId": "1",
        "string newURI": "https://ipfs.io/ipfs/Qmcy5SZyHLcFkEPCZGoTtPy5VVBLZfa311YkosEyAXtRYP"
}

{}

[
        {
                "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                "topic": "0xbddc2eaa1db7705e2379212c48c5c9b8d8c742b5703f0f48eae19c3cc6e1ad54",
                "event": "Update",
                "args": {
                        "0": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "1": "1",
                        "2": {
                                "_isIndexed": true,
                                "hash":
"0x890ca8c41a30a2d82d65e4176b1943ee2f62c6febec30641cb09a69d23bca668"
                        },
                        "3": "https://ipfs.io/ipfs/Qmcy5SZyHLcFkEPCZGoTtPy5VVBLZfa311YkosEyAXtRYP"
                }
        }
]
```


FIGURE 9. updateInsuranceCert() transaction output

B. UPDATE NFT

Updates consist of renewing the insurance certificate, the inspection report, or adding a new repair work to the repair log. Once an update of the vehicle occurs, its NFT must be updated accordingly through the NFT smart contract functions. There exist three update functions, one for each type of updatable data. For instance, when the insurance certificate has expired, the insurance company uploads the renewed version to IPFS and calls the *updateInsuranceCert* function by passing the IPFS hash and the NFT's id as parameters. The execution of the function is restricted to the *INSURANCE_ROLE* by the modifier *onlyRole()*. After the execution, the function emits the *Update* event, which includes who performed the update, the type of the update, and the IPFS hash of the updated data. Figure 9 shows the output of the *updateInsuranceCert*'s execution.

C. SELLING/PURCHASING NFT

Users that want to sell or purchase vehicles exploit the Marketplace smart contract functions. To sell a vehicle, the seller must list its NFT to the marketplace by calling the *listItem* function, which requires as parameters the id of the NFT and the price at which the seller wants to sell it. Before listing a new item, the *listItem* function performs some


```
{
        "uint256 tokenId": "1",
        "uint256 price": "10000"
}

{}

[
        {
                "from": "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8",
                "topic": "0x94e7b934c857a9e3202e8ed9c1f3f96e396b7d2b5885930d2001abcb51ff58fa",
                "event": "ItemListed",
                "args": {
                        "0": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                        "1": "1",
                        "2": "10000",
                        "seller": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                        "tokenId": "1",
                        "price": "10000"
                }
        }
]
```


FIGURE 10. listItem() transaction output


```
[vm] from: 0x617...5E7f2 to: Marketplace.listItem(uint256,uint256) 0xd8b...33fa8
value: 0 wei data: 0x883...02710 logs: 0 hash: 0x178...7976f
transact to Marketplace.listItem errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "marketplace not approved for this token".
Debug the transaction to get more information.
```


FIGURE 11. listItem() transaction failed output

checks with modifiers and *require* statements. In particular, it checks that whoever called the function is the owner of the NFT using the *isOwner* modifier, that the NFT has not been already listed using *notListed* modifier, that the price passed as parameter is greater than 0, and that the owner authorized the Marketplace smart contract to transfer his NFT. Once executed, the function emits a *ListItem* event to notify the system of the new item. Figure 10 shows the output log of a correct execution, whereas Figure 11 depicts the output log when the function fails due to the owner not authorizing the smart contract to transfer his NFT. A customer receives ownership of the NFT when he completes the payment of the offer accepted by the seller. The seller accepts by calling the *acceptOffer* function with the NFT's id as a parameter. This function checks that whoever has called it is the owner of the NFT and that the NFT is present on the Marketplace smart contract. Then, it sets a deadline by which the buyer must complete the payment, transfers the ownership to the Marketplace smart contract, and locks the NFT transferability by calling the *lock* function from the NFT smart contract. Finally, the function emits a *OfferAccepted* event that notifies of the ownership change. Figure 12 shows the output log when the function is executed. To complete the purchasing process, *confirmPayment* or *returnItem* functions must be called in case of successful payment or deadline expired, respectively.

VI. EVALUATION AND DISCUSSION

In this Section, we perform a security analysis by focusing on the security properties and known vulnerabilities. We compute an estimation of gas costs, we carry out a performance analysis for the main processes, and we compare our

```
{
        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
        "topic": "0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
        "event": "Transfer",
        "args": {
                "0": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                "1": "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8",
                "2": "1",
                "from": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                "to": "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8",
                "tokenId": "1"
        }
},
{
        "from": "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8",
        "topic": "0xa6da490df6e8f0e592e795b4d96c5810d8ddfda51c21577adbdd8e1a0fa47a67",
        "event": "OfferAccepted",
        "args": {
                "0": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                "1": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                "2": "1",
                "3": "9500",
                "caller": "0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB",
                "buyer": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2",
                "tokenId": "1",
                "offer": "9500"
        }
}
```


FIGURE 12. acceptOffer() transaction output

```
Compiled with solc
Number of lines: 2046 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 15 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 12
Number of informational issues: 62
Number of low issues: 4
Number of medium issues: 9
Number of high issues: 0

ERCs: ERC721, ERC165

+-----------------+-------------+---------------+------------+--------------+---------------+
|       Name      | # functions |      ERCS     | ERC20 info | Complex code |   Features    |
+-----------------+-------------+---------------+------------+--------------+---------------+
| IERC721Receiver |      1      |               |            |      No      |               |
|     Address     |      13     |               |            |      No      |   Send ETH    |
|                 |             |               |            |              | Delegatecall  |
|                 |             |               |            |              |   Assembly    |
|     Counters    |      4      |               |            |      No      |               |
|     Strings     |      5      |               |            |      No      |   Assembly    |
|       Math      |      14     |               |            |      Yes     |   Assembly    |
|    VehicleNFT   |      81     | ERC165,ERC721 |            |      No      |   Assembly    |
+-----------------+-------------+---------------+------------+--------------+---------------+
vehicle_nft.sol analyzed (15 contracts)
```

FIGURE 13. Slither output for vehicle_nft.sol

results with the state-of-the-art solutions. Also, we discuss our system limitations and future implementations.

A. SECURITY ANALYSIS

The proposed solution aims to provide a secure and reliable secondary market for second-hand vehicles based on decentralized technologies such as blockchain, smart contracts, NFTs, and decentralized storage. Those technologies grant availability, non-repudiation, data integrity, traceability, and authorization. These properties are described below:

- **Availability**: data availability is essential since users must always have access to data (current version as well as previous versions) about their vehicles or the vehicles they wish to purchase. This security property is guaranteed by decentralization. Blockchains and decentralized repositories store the data on each network node so that this information is available as long as at least one of these nodes is alive. This makes the data resistant to Denial-of-Service (DoS) attacks or other common attacks typical of centralized servers.
- **Non-repudiation**: each operation in the system is a transaction stored on the blockchain along with other information, such as who executed the transaction, the timestamp at which the transaction was executed, and the list of events related to that transaction. Transactions are stored on the blockchain in an immutable and public manner, so they cannot be repudiated. For example, after performing a repair work, a repair shop cannot deny that it was the performer.
- **Data integrity**: our solution stores small data on the blockchain and big data, such as vehicle metadata, in IPFS storage. The data stored on the blockchain is immutable, so it cannot be manipulated. As for IPFS storage, it creates URIs through hash functions, which means that even small changes, in a document, completely change its URI.
- **Traceability**: an NFT is associated with each vehicle produced by the OEM. NFTs greatly improve the traceability of transactions such as ownership transfer and metadata update, which are key operations in the secondary vehicle market. NFTs are generated following the ERC-721 standard, which guarantees their uniqueness through a numeric id. The id makes it easy to obtain the entire history of the vehicle to which it is attached.
- **Authorization**: our smart contracts implement function access control by defining specific roles. These roles are assigned by the OEM to actors verified by him (thus, assumed to be trustworthy) and ensure that each function is executed only and exclusively by the authorized actors. This mechanism blocks the malicious/erroneous execution of the smart contracts' code.

Smart contracts, like any other data on the blockchain, cannot be changed after the deployment. Therefore, they need to be thoroughly tested to remove any possible bugs. Our smart contracts have been tested on known vulnerabilities, such as reentrancy errors, integer overflow, transaction-order dependency, gas exhaustion problems, and so on via Slither [48], which is a Solidity static analysis framework written in Python 3 that runs a suite of vulnerability detectors. Figure 13 shows the result returned by Slither after analyzing the NFT smart contract. The result describes the number of functions tested and issues found. As it is possible to see, there are no high issues, whereas the tool detected nine medium issues discovered in the SafeMath.sol smart contract, and other minor issues, most of which were found in the inherited smart contracts. Figure 14 shows the result returned by Slither after analyzing the Marketplace smart contract. In this case, there are only eight minor issues related to the different Solidity versions of the inherited smart contracts.

B. GAS COST ESTIMATION

Our system was implemented based on the Ethereum blockchain, so we have reported an estimate of how much it would cost to run the functions related to the main processes. On Ethereum, executing a transaction or smart contract function has a cost measured in *gas*, a unit of measure that varies

```
Compiled with solc
Number of lines: 410 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 4 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 8
Number of low issues: 0
Number of medium issues: 0
Number of high issues: 0

ERCs: ERC721, ERC165

+-------------+-------------+--------------+-------------+--------------+--------------------+
|     Name    | # functions |     ERCS     | ERC20 info | Complex code |      Features      |
+-------------+-------------+--------------+-------------+--------------+--------------------+
| Marketplace |      13     |              |             |      No      | Tokens interaction |
|   IERC721   |      10     | ERC165,ERC721 |             |      No      |                    |
+-------------+-------------+--------------+-------------+--------------+--------------------+
marketplace.sol analyzed (4 contracts)
```

FIGURE 14. Slither output for marketplace.sol

depending on the complexity of execution: the greater the complexity, the greater the *gas* required. Each unit of *gas* has a cost in Ether (Ethereum's currency) called the *gas price*. The *gas price* is very volatile and depends on the number of transactions requested by users in a given period. In addition, users can add an amount to the base *gas price* to increase the execution priority of their transactions. The formula used to calculate the cost in Ether of our smart contracts functions is *total = (gas_price + priority_fee) * gas_units*. Table 4 shows the main functions of our smart contracts divided into processes and for each of them the cost in dollars. At the present time (January 24th, 2023) 1 Ether is equivalent to $1,616.23, whereas the *gas price* has a value of 16 Gwei (sub-multiple of Ether - 1 ETH = 1000000000 Gwei). In general, the cost of the functions is not excessive (the highest one is around $5, for minting a NFT) so the implementation is feasible. However, as we have already mentioned, this is a highly variable cost that could become very high within a short time. It must be noted, though, that when compared with the cost of a used car, even in the case of higher Ether costs, our system would still be worth of attention.

With respect to the state-of-the-art blockchain-based solutions reported in Table 1, our solution is a more cost-effective alternative. In fact, solutions that store all data directly on the blockchain were found to have significantly higher costs, particularly for larger amounts of data. In contrast, our solution utilizes IPFS to store only four hashes of fixed-size 32bytes, which are independent of the data size. This approach significantly reduces the costs associated with storage. Moreover, none of the compared solutions use NFTs. Those approaches are less efficient, as they necessitate the implementation of more complex architectures that are not optimized for tracking and exchanging data. Consequently, these alternative solutions could have higher execution costs due to their complexity. With respect to the state-of-the-art most famous commercial platforms, our solution also provides a cost-effective alternative. As a matter of fact, one of the main drawbacks of existing solutions for selling or buying used cars is that they impose significant fees on users. For example, most online platforms that allow users to list their vehicles for sale charge a fixed amount per listing, which can range from $5 (CarGurus[4]) to $130 (Hemmings[5]) depending on the site and the services they offer. Similarly, users who want to access vehicle history reports on potential purchases must pay a fee per report, which can range from $10 (VinAudit[6]) to $25 (CarFax[7]) depending on the provider and level of detail. In contrast, our solution requires users to pay only the transaction fee to list their vehicles, keeping the cost to less than $3. To complete the sale, the seller pays no more than $10, whereas the buyer spends less than $2 to make an offer. In addition, our solution does not require users to pay to view vehicles' history and information.

From a business cost perspective, we compared the costs reported by Autotrader.uk[8], the most popular used car marketplace platform in the UK, with the costs of the proposed solution. According to Autotrader 2022 annual report[9], in 2022 Autotrader had a monthly average of 430,000 live car stock; hence, we assumed an annual sales volume of 5,160,000 rounded to 4,500,000, as some ads may have been canceled without completing the sale. This estimation is supported also by the total number of second-hand cars sold in UK, in 2022, which is around 6,900,000[10]. As the 75% of the time spent on second-hand vehicles websites is spent on Autotrader, we assumed that a number between 65% (i.e., 4,485,000) and 75% (i.e., 5,175,000) of UK second-hand cars is sold on Autotrader (as some users would not rely on the internet to sell used cars). Looking at Autotrader's cost table, it was reported that the company spent £132m on administrative costs in 2022, divided into £69.8m as "staff costs", £20.5m as "marketing", £34.5m as "other costs" (data services, property costs, and other overheads) and £7.2m as "depreciation and amortization". Assuming the same costs for "staff costs" (it could be less for decentralized solutions), "marketing", and "depreciation and amortization", we considered only 1/3 of "other costs" (i.e., those related to data services) for the comparison. Thus, for each vehicle listed on the platform, we computed a cost in data services, for Autotrader, of around £2.55, which is almost the same as the cost of operating the proposed solution (£1.15 *confirmPayment* + £1.41 *returnItem* = £2.56) to complete the sale.

C. PERFORMANCE ANALYSIS

The performance analysis of our solution is dependent on the blockchain we used for implementation. Specifically, the Ethereum 2.0 blockchain (proof-of-stake) has a block generation time of 12 seconds, and each block can hold up to 15 million gas units, which can be scaled up to a maximum of 30 million gas units. Given these parameters, a "priority fee" can be added to the transaction cost to ensure fast transaction registration. Higher values of "priority fee" and low network

[4]https://www.cargurus.com
[5]https://www.hemmings.com/
[6]https://www.vinaudit.com
[7]https://www.carfax.eu
[8]https://www.autotrader.co.uk/
[9]https://plc.autotrader.co.uk/media/2460/at-ar22.pdf
[10]https://www.smmt.co.uk/category/vehicle-data/used-car-sales-data/

TABLE 4. Gas cost estimation and throughput (January 24th, 2023. Gas price is 16 Gwei = 0,000026 USD)

Process	Function name	Cost (gas)	Cost (USD)	Throughput
minting	createToken	200412	4,97	12,47
updating	setUpdateRequired	29236	0,76	85,51
	updateData	125308	3,11	19,95
	reviewVehicle	35958	0,93	69,53
selling	approve	49333	1,28	50,68
	listItem	90251	2,35	27,70
purchasing	makeOffer	75001	1,95	33,33
	rejectOffer	31045	0,81	80,53
	acceptOffer	136943	3,52	18,26
	confirmPayment	55853	1,44	44,76
	returnItem	69004	1,77	36,23

congestion result in faster execution times. Conversely, if the network is heavily congested and no priority fee is added, execution times will be higher than average. To evaluate the performance of our solution, we calculated the throughput of each function and present the results in Table 4, where the throughput represents the number of functions executed per second, taking into account the 12-second block registration time, the maximum block size of 30 million gas units, and low network congestion.

When comparing our solution to the ones in Table 1, we found that solutions using other blockchain technologies have different levels of congestion, different latency (block time), and therefore different throughput. For this reason, it was not possible to make a proper comparison. As for solutions implemented on Ethereum, our solution turns out to be more efficient because it uses NFTs and their standard, which provides an optimized architecture for traceability and token exchange, thus reducing the complexity of the functions, which means higher throughput. Traditional solutions, which do not need to rely on consensus algorithms that perform auditability over the execution of functions, are executed almost instantly (unless the network is overloaded).

D. COMPLEXITY ANALYSIS

The architecture of our solution is mainly based on two smart contracts, one for the creation and management of NFTs and the other for the implementation of the marketplace. It also includes a front-end platform that allows users to interface with different functionalities. The use of NFTs and the ERC-721 standard keeps the smart contract code relatively simple and the complexity of the functions moderate. Currently, users need an Ethereum wallet to interact with the framework, which makes it not accessible to everyone. We decided to avoid developing a "custodial" solution (which could have been simpler to be used by non-skilled users) to foster decentralization as much as possible. Comparing our solution with the solutions in Table 1, none of the analyzed blockchain solutions use NFTs, which implies the development of more complex architectures and functions for vehicle management. This results in higher costs and lower throughput. In addition, not all the solutions have implemented a front-end interface, which is necessary to simplify the user interaction. As for traditional solutions, while users do not require an Ethereum wallet, they cannot take advantage of the benefits offered by blockchains. As a result, users must rely on third parties to properly complete processes, such as buying and selling vehicles or checking the status of a vehicle before purchasing it.

E. LIMITATIONS AND FUTURE IMPLEMENTATIONS

As much as our solution has been tested and proved to be feasible, it is necessary to highlight some aspects that could be improved to increase its efficiency and scalability. First, our system was implemented on Ethereum, a public blockchain accessible by anyone. Public blockchains boast the advantages of complete decentralization, but at the same time, they could introduce privacy issues even though Ethereum addresses are not directly traceable to users. For this reason, in future implementations, it would be good to conduct a thorough analysis on what is the best type of blockchain for this use case, i.e., public or hybrid. In addition, notwithstanding the cost analysis showing affordable costs, the Ethereum blockchain has high costs compared to other blockchains. Thus, other blockchains could be considered to reduce the costs.

Regarding scalability, our solution was implemented considering a single OEM with only one smart contract for NFTs management. In future implementations, the system could be extended to multiple OEMs, each owning a smart contract for NFTs management, all connected to a single Marketplace smart contract. In this way, users could have the ability to buy and sell vehicles of different brands within the same platform. In addition, each OEM could customize its own smart contract within the limits of the features required by the system.

Another limitation of the system concerns the verification of updates. For simplicity, we have assumed in our solution that the OEM is able to verify that the updates claimed by the authorized actors have actually been made correctly. In reality, the OEM may not be able to provide this assurance or still an overload may be created related to the number of vehicles to be reviewed. Therefore, in future implementations, a multi-entity based system could be developed that can confirm the correctness of updates. These entities could also be the smart contracts themselves, by implementing an automatic control logic based on oracles.

VII. CONCLUSION

This work presented an NFT-based solution for tracking and buying/selling vehicles in the second-hand market. The proposed solution is based on a fully decentralized system that leverages the Ethereum blockchain, NFTs and IPFS. Specifically, were designed and implemented two smart contracts to manage the NFTs and the Marketplace respectively, as well as a front-end that allows users to interact with them. We have demonstrated that the proposed decentralized application provides greater security and reliability than

solutions currently used in a field where these properties are considered fundamental. We also performed a cost and performance analysis to assess the feasibility of the system and a vulnerability analysis to minimize exposure to cyber-attacks. In addition, we compared our solution to other existing solutions, leveraging blockchain to show the advantages introduced by relying on NFTs. Finally, we discussed the limitations currently present in our system. As future works, we plan to test our solution on different blockchains to evaluate costs and performances. Furthermore, we intend to increase the scalability by integrating the capability of several OEMs to join the system.

REFERENCES

[1] P. Fraga-Lamas and T. M. Fernández-Caramés, "A review on blockchain technologies for an advanced and cyber-resilient automotive industry," *IEEE access*, vol. 7, pp. 17 578–17 598, 2019.

[2] K. Singh, "Blockchain technology: A potential game changer for automotive industry," *International Journal of Advanced Research in Management and Social Sciences*, vol. 9, no. 3, pp. 49–55, 2020.

[3] Y. Yu, C. Yao, Y. Zhang, and R. Jiang, "Second-hand car trading framework based on blockchain in cloud service environment," in *2021 2nd Asia Conference on Computers and Communications (ACCC)*. IEEE, 2021, pp. 115–121.

[4] S. El-Switi and M. Qatawneh, "Application of blockchain technology in used vehicle market: A review," in *2021 International Conference on Information Technology (ICIT)*. IEEE, 2021, pp. 49–54.

[5] Č. Duboka, Ž. Filipović, M. Gordić, and M. Došlić, "Second hand vehicle maintenance frauds," in *Paper NMV0912 presented at the XXII JUMV International Automotive Conference*, 2009, pp. 0912–1.

[6] European Parliament. (2018) European Parliament vote is a significant step towards restoring consumer trust in the used car market and helping citizens save billions. [Online]. Available: https://citainsp.org/wp-content/uploads/2018/05/20180523-joint-press-release-Ertug-report-final-corr.pdf

[7] A. Nigam, S. Sangal, A. Behl, N. Jayawardena, A. Shankar, V. Pereira, Y. Temouri, and J. Zhang, "Blockchain as a resource for building trust in pre-owned goods' marketing: a case of automobile industry in an emerging economy," *Journal of Strategic Marketing*, pp. 1–19, 2022.

[8] M. Swan, *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[10] N. Szabo. (1997) The idea of smart contracts.

[11] G. Wang and M. Nixon, "Sok: tokenization on blockchain," in *Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*, 2021, pp. 1–9.

[12] S. A. Gebreab, H. R. Hasan, K. Salah, and R. Jayaraman, "Nft-based traceability and ownership management of medical devices," *IEEE Access*, vol. 10, pp. 126 394–126 411, 2022.

[13] A. Musamih, I. Yaqoob, K. Salah, R. Jayaraman, M. Omar, and S. Ellahham, "Using nfts for product management, digital certification, trading, and delivery in the healthcare supply chain," *IEEE Transactions on Engineering Management*, 2022.

[14] N. Alnuaimi, A. Almemari, M. Madine, K. Salah, H. Al Breiki, and R. Jayaraman, "Nft certificates and proof of delivery for fine jewelry and gemstones," *IEEE Access*, vol. 10, pp. 101 263–101 275, 2022.

[15] F. K. Elmay, K. Salah, R. Jayaraman, and I. A. Omar, "Using nfts and blockchain for traceability and auctioning of shipping containers and cargo in maritime industry," *IEEE Access*, vol. 10, pp. 124 507–124 522, 2022.

[16] A. Battah, M. Madine, I. Yaqoob, K. Salah, H. R. Hasan, and R. Jayaraman, "Blockchain and nfts for trusted ownership, trading, and access of ai models," *IEEE Access*, vol. 10, pp. 112 230–112 249, 2022.

[17] U. Majeed, L. U. Khan, S. S. Hassan, Z. Han, and C. S. Hong, "Fl-incentivizer: Fl-nft and fl-tokens for federated learning model trading and training," *IEEE Access*, pp. 1–1, 2023.

[18] M. Madine, K. Salah, R. Jayaraman, A. Battah, H. Hasan, and I. Yaqoob, "Blockchain and nfts for time-bound access and monetization of private data," *IEEE Access*, vol. 10, pp. 94 186–94 202, 2022.

[19] (2023) Openzeppelin docs erc721 smart contract. [Online]. Available: https://docs.openzeppelin.com/contracts/4.x/api/token/erc721

[20] Groupe Renault. (2017) Groupe Renault teams with Microsoft and VISEO to create the first-ever digital car maintenance book prototype.

[21] VINChain. (2017) Decentralized Vehicle History - Car Accident History Check by VIN. [Online]. Available: https://vinchain.io/

[22] CarVertical. (2017) Car-Vertical – VIN Decoder & Car History Report. [Online]. Available: https://www.carvertical.com/

[23] P. K. Singh, R. Singh, and S. Nandi, "V-care: A blockchain based framework for secure vehicle health record system," *arXiv preprint arXiv:2007.13647*, 2020.

[24] M. Chanson, A. Bogner, F. Wortmann, and E. Fleisch, "Blockchain as a privacy enabler: An odometer fraud prevention system," in *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, 2017, pp. 13–16.

[25] M. Vasile and B. Groza, "Demetra-decentralized metering with user anonymity and layered privacy on blockchain," in *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2019, pp. 560–565.

[26] G. Saldamli, K. Karunakaran, V. K. Vijaykumar, W. Pan, S. Puttarevaiah, and L. Ertaul, "Securing car data and analytics using blockchain," in *2020 Seventh International Conference on Software Defined Systems (SDS)*. IEEE, 2020, pp. 153–159.

[27] K. L. Brousmiche, T. Heno, C. Poulain, A. Dalmieres, and E. B. Hamida, "Digitizing, securing and sharing vehicles life-cycle over a consortium blockchain: Lessons learned," in *2018 9th IFIP international conference on new technologies, mobility and security (NTMS)*. IEEE, 2018, pp. 1–5.

[28] T. Reimers, F. Leber, and U. Lechner, "Integration of blockchain and internet of things in a car supply chain," in *2019 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPCON)*. IEEE, 2019, pp. 146–151.

[29] W. Zhang, L. Guo, and Y. Li, "Used car traceability system based on blockchain," in *2021 International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*. IEEE, 2021, pp. 235–240.

[30] J. Zhang, H. Zhao, Y. Yang, and J. Yan, "Towards transparency and trustworthy: A used-car deposit platform based on blockchain," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2019, pp. 46–50.

[31] S. G. Yoo and B. Ahn, "A study for efficiency improvement of used car trading based on a public blockchain," *The Journal of Supercomputing*, vol. 77, no. 9, pp. 10 621–10 635, 2021.

[32] (2023) Interplanetary file system. [Online]. Available: https://ipfs.tech/

[33] M. N. Kumar, G. Akshatha, M. D. Bangre, M. Dhanush, and C. Abhishek, "Decentralized used cars bidding application using ethereum," in *2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)*. IEEE, 2021, pp. 1–7.

[34] G. Subramanian and A. S. Thampy, "Implementation of hybrid blockchain in a pre-owned electric vehicle supply chain," *IEEE Access*, vol. 9, pp. 82 435–82 454, 2021.

[35] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[36] X. Yang, Y. Zhang, S. Wang, B. Yu, F. Li, Y. Li, and W. Yan, "Ledgerdb: a centralized ledger database for universal audit and verification," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 3138–3151, 2020.

[37] "Permissionless innovation," *White Paper*, 2022.

[38] L. Baird, M. Harmon, and P. Madsen, "Hedera: A public hashgraph network & governing council," *White Paper*, vol. 1, pp. 9–10, 2019.

[39] A. Brock, D. Atkinson, E. Friedman, E. Harris-Braun, E. Mcguire, J. M. Russell, N. Perrin, N. Luck, and W. Harris-Braun, "Holo green paper," *Green Paper*, 2018.

[40] "Radix defi white paper," *White Paper*, 2021.

[41] (2023) Metamask. [Online]. Available: https://metamask.io/

[42] (2023) Remix. [Online]. Available: https://remix.ethereum.org

[43] (2023) Openzeppelin docs accesscontrol smart contract. [Online]. Available: https://docs.openzeppelin.com/contracts/4.x/api/access

[44] (2023) Openzeppelin docs erc721uristorage smart contract. [Online]. Available: https://docs.openzeppelin.com/contracts/4.x/api/token/erc721\#ERC721URIStorage

[45] (2023) React. [Online]. Available: https://reactjs.org/

[46] (2023) Web3. [Online]. Available: https://web3js.readthedocs.io/en/v1.8.1/
[47] (2023) Sjs-ipfs. [Online]. Available: https://github.com/ipfs/js-ipfs
[48] Crytic. (2023) Slither. [Online]. Available: https://github.com/crytic/slither

ALBERTO BUTERA received the B.S. and M.S. degrees in computer engineering from Politecnico di Torino, Italy, in 2018 and 2022 respectively. He is currently an industrial Ph.D. student in computer engineering at Reply and the Department of Control and Computer Engineering of Politecnico di Torino, Italy. His research interests include decentralization, blockchain technology, smart contracts, and Web3 development.

VALENTINA GATTESCHI received the B.S. and M.S. degrees in management engineering from Politecnico di Torino, Italy, in 2005 and 2008 respectively, and the Ph.D. degree in computer engineering from Politecnico di Torino, Italy, in 2013. She currently is an Associate Professor at the Department of Control and Computer Engineering of Politecnico di Torino, Italy. Her research interests include intelligent systems, semantic computing, and blockchain technology.

FILIPPO GABRIELE PRATTICÒ received his P.h.D. degree in computer engineering from Politecnico di Torino, Italy, in 2022. He is currently a post-doctoral fellow at the Department of Control and Computer Engineering of Politecnico di Torino. His research interests include eXtended Reality, human-computer, human-robot interaction, vehicular technology and simulation. He is a member of the IEEE and IEEE Consumer Technology Society.

DANIELA NOVARO is Associate Partner at Reply. She has 20 years of professional experience initially in AI for Telco companies and then in consultancy, in Reply. Her skills in solution design, platform delivery and project and people management increased over time working on in different markets and industries, from Telco to Automotive and Financial Services.

DEBORAH VIANELLO is a IT Consultant at Reply. She graduated in 2012 in Computer Science at Università Statale di Milano, Milan, Italy, and then started her working career in consultancy. Since 2018 she is in Blockchain Reply, the company inside Reply which is focused on studying, designing and delivering blockchain-based solutions, in which she contributed to the design and development of different blockchain solutions in different industries.