Ethereum-Based Decentralized Car Rental System

Cándido Caballero-Gil[0000−0002−6910−6538], Pino Caballero-Gil[0000−0002−0859−5876], Jezabel Molina-Gil, and Néstor García-Moreno

Department of Computer Engineering and Systems
University of La Laguna.
Tenerife. Spain.
{ccabgil, pcaballe, jmmolina, ngarciam}@ull.edu.es

Abstract. Blockchain is a pioneering technology that allows the creation of innovative information exchange ecosystems thanks to unique properties, such as immutability, distribution and transparency. By combining this technology with the emergent internet of things (IoT), many innovative business models can be created. The main objective of this work is to present the design and an initial implementation of a decentralized rental system that takes advantage of smart contracts developed on a public blockchain, combined with the potential of the IoT and its intimate relationship with cyber-physical systems. The logic to be implemented in the blockchain is applied in this paper by using the Ethereum programming language, so that the developed application covers the entire car rental process offered in traditional web applications, but adding more autonomy, functionalities and ease of use for both lessees and lessors. Following Ethereum application development guidelines, all business logic is located in the smart contracts implemented in the network, where they can be used to control the car rental system. While this is a work in progress, the results obtained in the first proof of concept have been very promising.

Keywords: Blockchain · Smart contract · Car rental · Cyber-Physical System.

1 Introduction

In recent years, the emergence of various disruptive technologies is generating great and rapid changes in society. In addition, this transformation has been favoured by the COVID-19 pandemic, which has caused people to try to avoid physical interaction with other people and with physical money. On the other hand, the sharing economy is transforming markets around the world in such a way that the consumer mindset is moving from prioritizing property to prioritizing use of goods. As a result, many rental and sales companies are facing the need for sudden and urgent transformations.

The combination of technologies such as the internet of things (IoT) and cyber-physical systems (CPSs) with blockchain technology offers the possibility of allowing shared use, avoiding interaction with both people and physical money.

IoT technologies and CPS devices have evolved significantly in recent years in many different sectors and, in particular, in the automotive sector. For example, tracking systems are widely known and used nowadays. In contrast, remote vehicle systems are not as well known, but currently, it is possible to remotely lock and unlock the car and perform other more complex and advanced operations on the vehicle. In addition, autonomous cars are becoming an ever closer reality. However, until regulations and technologies are prepared for that, a possible intermediate step can be the rental of connected cars through a decentralized platform that allows to rent and open them remotely.

Until recently, all electronic monetary transactions between two parties required centralized platforms, such as banks or credit card entities, in order to be able to mediate the interests of transmitter and acquirer and enable valid secure payments. Specifically, these platforms store the description of the items for sale or rent together with their price, so that customers must interact with those platforms to purchase or rent any item. A common feature of those platforms is that they all require a trusted third party (TTP) to operate, resulting in many disadvantages for consumers. For example, generally, consumers must register on each platform separately, share their private data with the owners of the platform, pay transaction fees and rely entirely on the security of the TTP. A solution to overcome all these problems is possible through the application of the smart contract concept, which was first introduced in the early 1990s by Szabo [1], as a digital protocol that facilitates an agreement process between different parties without intermediaries, enforcing certain predefined rules that directly incorporate contractual clauses in hardware and software.

The most important novelty that smart contracts include is the fact that they are agreed and executed automatically and directly by the network nodes, since they eliminate the need to depend on a TTP. In particular, every smart contract is a set of executable functions and state variables, which are input variables and output variables. During the execution of any function of a smart contract, the status of the state variables is changed depending on the logic implementation. In every smart contract, contract conditions are defined so that, as soon as the conditions are met, the contract is executed and the output variables are delivered to the entity indicated in the contract conditions. Before the emergence of blockchain technology, there was no platform that could make the idea of smart contract a reality. Blockchain-enabled smart contracts offer the ability to create business logic. The cryptocurrency invented in 2008 called Bitcoin is a specific example of a smart contract, but the smart contract idea requires a tool to enable the implementation of the general logic of programs and decentralized applications that go far beyond the simple transfer of monetary values.

Ethereum is a platform based on blockchain technology and developed in open source, which allows the creation and execution of smart contracts in general. In particular, Ethereum is a set of network, platform and protocol that

shares many of the basic concepts behind the realization of Bitcoin, such as transactions, mining and consensus. In fact, Ethereum started on the same consensus algorithm as Bitcoin, known as proof of work (PoW), based on mining using computational brute force. PoW involves significant energy consumption and therefore is not the best option for the environment. Furthermore, the growing demand for Ethereum both as a cryptocurrency and as a platform for smart contracts is causing a rapid increase in the price of its cryptocurrency, called Ether, so that the price of Gas, which is the unit that measures the amount of computational effort that Ethereum requires to carry out transactions or interactions within the network, is now higher than the fees charged by traditional banks. Consequently, Ethereum is now under the process of being moved to proof of stake (PoS) as the new basis of the distributed consensus algorithm to mine Ethers. With the PoS mechanism, the probability of finding a block of transactions, and receiving the corresponding prize, is directly proportional to the number of coins that have been accumulated up to that moment. In this way, the individuals who help the network to validate transactions and create new blocks, known as stakers, are the most interested in the correct functioning of the network.

The Ethereum blockchain is an example of distributed ledger technology (DLT), which is a database of information shared and replicated over a network of computers in different locations, without a central administrator. Therefore, the DLT characteristics of Ethereum allow the development of Decentralized Applications (DApps), which are computer applications that run on a distributed computer system without a central server. In the case of Ethereum, DApps are often called smart contracts.

The use of smart contracts provides a layer of security and transparency that centralized applications cannot offer. In the proposal here described, sensitive information is stored in the blockchain, preventing it from being modified or manipulated. On the one hand, in blockchain-based schemes, cryptographic hash functions are used to generate an unpredictable value of fixed length from each arbitrary input, so that the process is not computationally reversible. In that way, given an output from a cryptographic hash function, it is practically impossible to calculate the corresponding input. Hash functions are often used to verify data integrity, as they allow checking whether they have been modified or not, since the output changes radically with any small change in the input. In blockchain, the hash function is used as a way to verify if the information stored in each block has changed or not. Bitcoin uses the SHA256 hash function, while Ethereum's cryptographic hash function is Keccak-256, which is more resistant to length extension attacks. On the other hand, public key cryptography is used to protect blockchain-based decentralized transactions. In asymmetric cryptography, each user has a pair of keys, one public and the other private, so that in the case of encryption, the public key is used to encrypt and the private key to decrypt. In blockchain, asymmetric cryptography is used to sign transactions so that each user uses their own private key to sign their transactions, while their public key is used by other users to verify the authorship of those transactions. Besides, public

key cryptography is also used to create the public address that users use to send and receive funds. Thus, the public key is used to receive funds, while the private key is used to sign transactions to spend the funds. The public key cryptosystem used in Ethereum and Bitcoin is based on elliptic curves. Specifically, the used digital signature scheme is the Elliptic Curve Digital Signature Algorithm, which is a variant for elliptic curves of the standard for digital signatures, known as Digital Signature Algorithm. In particular, both blockchain technologies use the elliptic curve secp256k1.

This paper introduces a decentralized social web platform for car rental, called AutoRent. It is a DApp based on smart contracts on the Ethereum blockchain, and on IoT and CPS for smart locking/unlocking of rented cars. In particular, the proposal consists of a complete design based on the aforementioned technologies, which facilitates the management of car rentals and the remote opening of vehicles in such a way that physical interaction between people and with money is unnecessary. In the proposal, all the processes, from the offer of the vehicle, to the verification of the identity, the driving credentials, the reservation, the payment, the opening and the return of the vehicle, can be carried out in a secure way, without the need for physical interaction.

The structure of this paper is as follows. In Section 2, some works related to the proposal are mentioned. Section 3 details the proposal, including used DApp technologies, developed smart contracts, applied remote control technologies. In Section 4, several implementation issues are discussed, while Section 5 includes a brief security analysis. Section 6 includes some conclusions and future works. Finally, an appendix closes the paper with several code listings.

2 Related Works

Blockchain is a relatively new area of research, but several reviews on some related concepts have already been published in recent years. Several books on bitcoin had already been published when the first book on blockchain appeared in 2015, written by Swan [2]. Later in 2017, Dannen wrote the first book [3] featuring Ethereum and the object-oriented programming language for writing smart contracts called Solidity. In 2018, Vujicic published a brief introduction to blockchain technology, bitcoin, and Ethereum [4].

Regarding blockchain-based applications for objectives similar to that of this work, several authors have described different proposals. A DApp for the sharing of everyday objects based on a smart contract in the Ethereum blockchain is demonstrated in the short paper by Bogner et al. [5]. The work by Huh et al. [6] proposes a proof of concept using an Ethereum-based blockchain computing platform to control and configure IoT devices. In the survey [7], Reyna et al. analyse how blockchain could potentially improve the IoT.

In relation to car rental, there are currently numerous practical solutions that are not based on blockchain technology. For instance, the company called Social Car [8] is an example of a P2P solution based on private cars. Another example is Cocoche [9], which is a private car rental platform located in South America.

Getaround [10], formerly Drivy, is another private car rental platform, located in Europe.

In the specific case of blockchain-based applications proposed for the rental of goods, some solutions can be found in the bibliography. The paper written by Hassija et al. [11] presents a review of how blockchain and smart contracts can be used to create a platform for car rental services. The paper by Niya et al. [12] introduces the design and implementation of an Android-based Peer-to-Peer (P2P) purchase and rental application, which takes advantage of smart contracts and the Ethereum public blockchain and uses WiFi-Direct for the communication between the parties. Another related work is the one published by Ren et al. [13], which presents a car rental alliance solution based on blockchain technology. Valastin et al. propose in [14] a solution to create and implement peer-to-peer short term car-sharing application based on Ethereum blockchain technology and smart contracts using Solidity. Solidity works with blockchain. Saurabh *et al.* introduces in [15] the concept of a decentralized car-sharing application as an alternative to private car ownership.

Recently, some DApps have emerged in the market to exploit smart contracts to develop secure transaction executions for car rentals. HireGo [16] is the first decentralized car-sharing platform developed on Ethereum operating in the United Kingdom since 2019. Its uses its own token to pay for the use of its service, and Internet of Things and smart cars to automate operations. Darenta [17] applies Blockchain technology to create a peer-to-peer car rental marketplace that connects people who need to rent private car with car owners. Helbiz [18] is a decentralized mobile platform based on the Ethereum blockchain and propelled using its own coin tokens to rent bicycles and electric scooters. DAV [19] is a blockchain-based open source global transportation company that provides a computer network to connect self-driving vehicles to everyone, enabling them to transact with them using DAV tokens. FFQuest [20] is another marketplace that uses the blockchain for car rental. A recent project is WONO [21], which includes a P2P platform for car rental, apartments rental and freelancing, based on smart contracts using PoS to ensure consensus in the network. Mix.rent [22]is another recently developed vehicle-renting services platform that uses P2P general consensus and smart-contract technology for agreement between vehicle owners and renters.

In Section 5, Table 1 shows a comparison between some of the proposals described in the aforementioned papers and companies and the proposal described in this work. From the analysis of the related works mentioned in this section, it can be concluded that the formal description and practical development of new service concepts and business models for blockchain-based shared mobility using DApps on Ethereum is an emerging area that is beginning to be explored and in which there is still much to improve. Specifically, the field of application of electronic commerce for the purchase and rental of goods is one with great potential, although several problems such as efficiency, scaling, interoperability and privacy must be solved prior to its exploitation. The number of companies offering DApps in different sectors has been growing in recent years. In this sec-

tion, some examples of them have been mentioned. The fact that the number of users has increased rapidly demonstrates the good performance of DApps for different functions.

Car rental system	Blockchain implementation	Remote tracking	Remote opening	Implementation
AutoRent	Yes, Ethereum	Yes, GPS	Yes	Yes, blockchain and web platform
Cocoche [9]	No	No	No	Yes, web platform
Darenta.[17]	No	No	No	Yes, web platform
Getaround [10]	No	No	No	Yes, web platform
HireGo [16]	Yes, HireGo coin	No	No	No
PuRSCA. [12]	Yes, Ethereum	No	No	Yes, blockchain and smartphone app
Social Car [8]	No	Yes, GPS	Yes	Yes, web platform

Table 1. Technology comparison of state-of-art works

In this sense, the main objective of this work is the definition of a novel and secure platform that takes advantage of blockchain, IoT technology and CPS devices to improve trust, information exchange and interaction in P2P car sharing ecosystems. Part of this is also the creation of a prototype to demonstrate the technical feasibility of the proposed service concept using state-of-the-art blockchain, IoT and CPS technologies.

3 AutoRent Components and Technologies

The blockchain infrastructure provides the trust, transparency and immutable traceability required for developing the platform for renting cars securely. The first step that must be taken to build a DApp is to set up an environment that allows the proper development of smart contracts, so a suitable choice of technologies and software is essential to correctly deploy a solution.

The proposal has been developed using the JavaScript Truffle framework, which is one of the most popular in DApps development. This framework includes the entire development cycle of an application: preproduction, production and deployment. For the development of smart contracts, the high-level language Solidity oriented to contracts has been used. Its syntax is similar to that of JavaScript and is specifically focused towards the Ethereum virtual machine (EVM). Furthermore, the remote control included in the system has been solved using CAN-CONTROL [23] devices, which provide the possibility to lock/unlock a car remotely and control its windows by special SMS/GPRS commands. Below, more elaborated details about the used technologies are included.

3.1 Used DApp Technologies

Typically, in non-decentralized application development environments, the first step is to write code and then compile and execute the code on a computer. However, the development of a DApp is different because it brings decentralization to code execution, allowing any machine on the network to execute the code. To develop a DApp without paying costs for the implementation or execution, the best solution is to use a blockchain simulator, which is a lightweight program that contains implementations of the Ethereum blockchain to be ran locally with minor modifications, such as control over mining speeds. In this way it is possible to mine blocks instantly and run DApps very quickly.

Some of the technologies used in this work to allow this development cycle to be carried out using a blockchain simulator are the following:

- Truffle [24]: Framework that encompasses the entire Ethereum ecosystem, including decentralized application development, application testing and deployment on the Ethereum blockchain. It allows the development, compilation, test and deployment of smart contracts in the blockchain. It also allows the maintenance of private networks or public test networks (Rinkeby, Kovan, Ropsten). In addition, this framework contains an integrated command line interface for application development and configurable scripts to automatically launch contracts to the blockchain.
- Web3 [25]: Collection of libraries which allow interaction with a local or remote Ethereum node using a HTTP or IPC connection.
- Solidity [26]: It is an object-oriented, high-level language for implementing smart contracts. Solidity was influenced by C++, Python and JavaScript and is statically typed, supports inheritance, libraries and complex user-defined types among other features.
- Ganache [27]: Environment for the deployment of private or local blockchains for testing and preproduction. It comes integrated with Truffle. This environment allows to inspect the entire record of the transactions of the blocks and the whole chain of private blockchain.
- Embark [28]: Framework to develop and implement decentralized applications without a server. Embark is currently integrated with Ethereum Virtual Machine (EVM), InterPlanetary File System (IPFS) and decentralized communication platforms.
- MetaMask [30]: Browser extension that acts as a bridge between the browser and the blockchain. It allows to visit and interact with DApps without running a full Ethereum node.

3.2 Smart Contracts

Truffle includes a compiler for smart contracts developed in Solidity, with the tools necessary for the deployment on the Ethereum network (migrations). In particular, it has a command-line interface to facilitate this task, allowing to compile, deploy (migrate), tests, etc.

Driving License Contract One of the main problems of developing a decentralized car rental application is verifying the authenticity of the driver's license in order to prevent anyone from being able to provide a fake permit when renting a car. The ideal way to remedy this problem would be for the government to have an official platform (which maybe based on a blockchain) where it is possible to check if a person has a specific driving license. However, in most countries there is no official platform to confirm the authenticity of driving licenses.

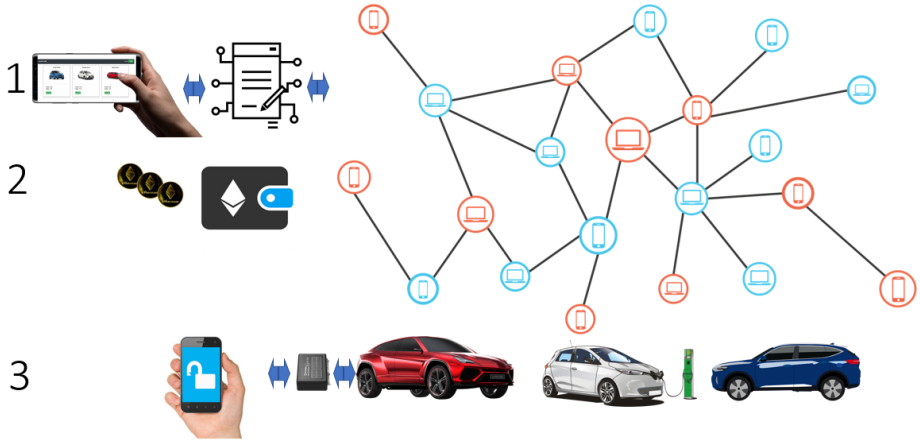

Fig. 1. AutoRent smart contract steps

As an alternative, in this work a private blockchain has been created, in which the platform administrator in charge of the car rental company can introduce the driving licenses that have been previously validated, thus giving the corresponding lessees access to rent cars. This contract is simply composed of a method to introduce previously validated driving licenses, and another method listing the driving licenses that have been introduced.

Another possibility, compatible with licenses in digital format, would be the use of public key certificates. In this proposal, the certifying authority would be the National Traffic Administration or equivalent in each country. Hence, the license validation would be corroborated by the possession of a valid public key certificate, which could be easily verified against the responsible authority that issued the certificate.

AutoRent Contract In order to lease a car belonging to a rental fleet, a smart contract is necessary that verifies two different aspects: that the desired vehicle exists on the private blockchain of the company and that it has not yet been rented for the selected period. If everything is correct, this contract establishes a mapping between vehicle and customer data, declaring it as a tenant (see Figure 1). First, the lessee checks the available cars and selects the one he/she

wants to rent. At this moment, a smart contract is signed, the transaction in cryptocurrency is done, and the key for opening the car is sent to the lessee's smartphone. This key is valid for the rented period, so the lessee can then lock and unlock the car with the Bluetooth connection of his/her smartphone.

If the lessee deposit is insufficient or non-existent the smart-contract is not signed and the user does not receive the key. If the lessee has bad behaviour an extra expense will be charged, even the car can be located and blocked. At any time, the lessee can add days to the contract in order to extend the rent, and the new cost is applied. In order to return the car, the lessee must return the car in the conditions indicated in the smart-contract.

Once the vehicle is returned, the status of the vehicle changes to available, and the existing customer-vehicle relationship is deleted so that from then another person can rent the car. The car rental benefits are sent to the digital wallet of the contract owner who launched it to the Ethereum network. If funds were left over, these would be sent to the lessee's Ethereum account.

3.3 Remote Control

AutoRent proposal includes a smart rental car door lock solution that integrates IoT, CPS and smart contracts to manage access to rented cars in a decentralized way. The solution eliminates the need for lessors and lessees to meet in person, as the blockchain network validates the information and enables a unique rental car smart door lock setting for each new lessee. In addition, the solution prevents anyone else besides the lessee from accessing the car while it is rented.

First, some remote control system requirements were analysed, both from the perspective of the lessee and the lessor, to capture the expected functionalities of the chosen system. On the one hand, the lessor must be able to provide the lessee with access to the rented car without having to meet in person. On the other hand, it is also necessary that no one other than the lessor can provide access to the rented car, that no one but the lessee has access to the car during the rental period, and that the previous lessees do not have access to the rented car after the end of their rental. Additionally, it is convenient that access to the car can be immediately revoked for any lessee who breaches the conditions of the contract, and that no one can track when the car lock is open or closed.

The technology chosen for the remote control in the AutoRent proposal was CAN-CONTROL, which allows to connect directly to the vehicle CAN lines. Besides, the implemented solution uses Infura API as a bridge to connect the IoT infrastructure to the Ethereum blockchain network. In this way, the proposal includes the automatic reception by the lessee of the necessary special SMS/GPRS commands to lock/unlock the car according to the smart contract.

4 AutoRent Implementation

Smart Contract Implementation To deploy the contracts in the blockchain, the specification of the network is required. To do this, it is necessary to create

a configuration file, 'truffle.js' which is taken into account when compiling and launching the smart contract.

The private network and port 7545 (private blockchain) are specified. Next, migrations for the contract are launched. A migration is the operation of deploying the contract in the blockchain (see Listing 1.4), specified by means of a JavaScript file using the native Truffle artifact library.

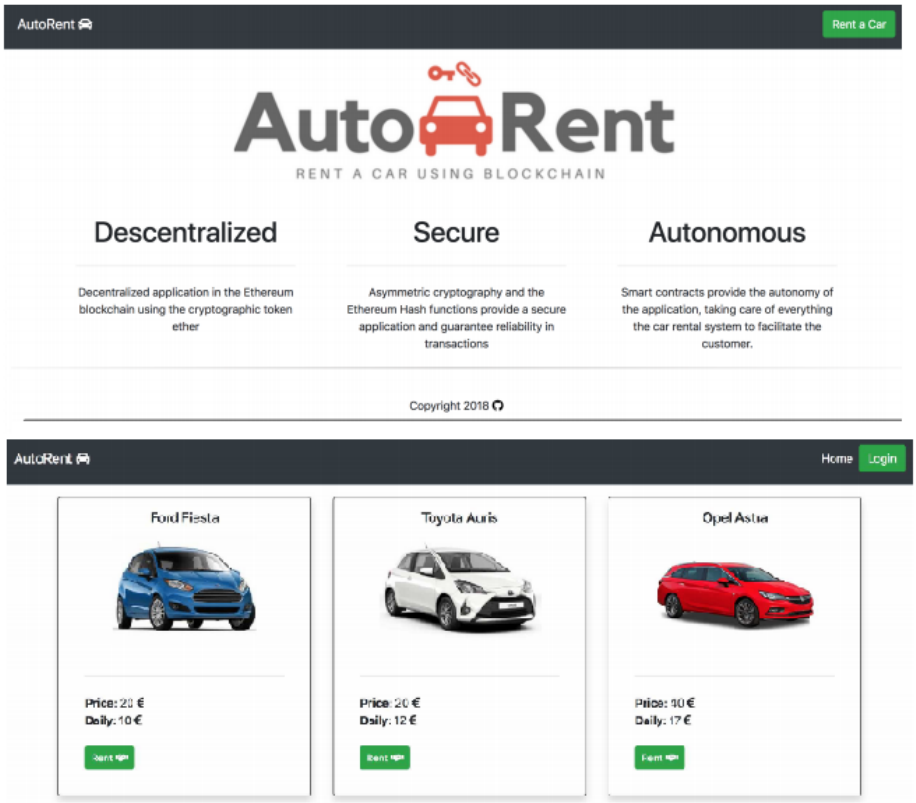

Fig. 2. AutoRent web interface

Since MetaMask is a browser extension and is the one that interacts with the Ethereum nodes, the interaction is with the contract from the browser side. To do this, the .json files generated from the contract compilation phase must be loaded in the lessee side.

Among the relevant functions of the smart contract in charge of the entire rental process of a car, the following stand out. On the one hand, the rentCar function receives all the parameters of the customer and the car to be rented. This function (see Listing 1.5) verifies that the car is not rented and that the license it has received by parameter exists in the other contract. On the other

hand, the returnCar function verifies that the customer has no pending charges. If the lessee has no pending charges, the deposit that has been left over to the lessee is returned. When the car is returned, the lessor receives its rental benefits, the SMS/GPRS lock/unlock commands sent to the customer are no longer valid, and the car is listed in the web app as not rented again.

Front-End Implementation The lessee can rent an available car, add a deposit and return a rented car. To rent a car, he/she simply has to choose an available car in the Web (see Fig. 2). Automatically the confirm transaction tab will open (see Fig. 3). If successful, the application is updated and marks the newly rented car as unavailable. The contract is responsible for all these operations, as well as for storing the lessee's deposit and charging the costs of surcharges in case he/she has not added a required deposit or has spent some day without paying the corresponding rent. The contract is also responsible for returning the excess deposit to the lessee, and giving the benefits to the lessor.

5 Security Analysis

Unlike the proposals mentioned in Section 2, the scheme described in this work includes a car rental system that is fully distributed and offers remote tracking and car lock/unlock. Besides, this work also describes a preliminary implementation of the system developed following the standards of a DApp to include a novel automation process with smart contracts based on blockchain technology that provides a secure distributed platform. This implementation is open source and no company controls the tokens.

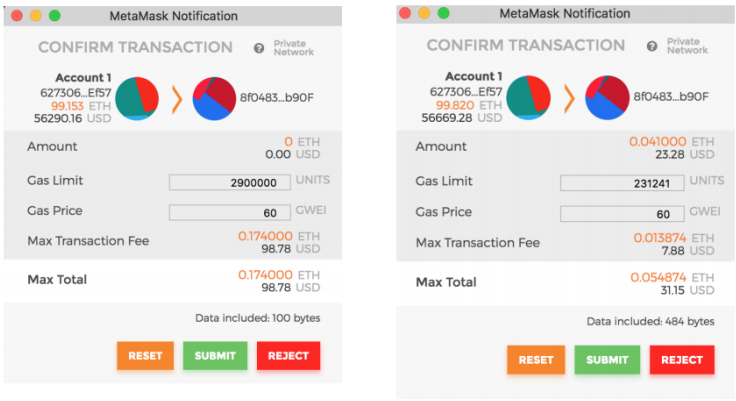

Fig. 3. AutoRent MetaMask transaction example

Table 1 shows a comparison between our proposal and other existing solutions for car rental from different companies and authors. In particular, it shows

some aspects related to the security of the technology used in each proposal and indicates whether or not these solutions implement the three modules proposed in this work. That is, an analysis has been carried out as to whether the studied solutions include a blockchain system, remote location and remote opening. In addition, it has also been analysed whether they are real implementations or only theoretical proposals.

Unlike traditional web applications, in AutoRent, customers do not need to register in any database, they do not have to store any password, and there is no control over their sessions. Their information is directly managed by the smart contract, which is in charge of using the client's digital wallet and storing the data in it, so this information is not vulnerable to security attacks. Since the transactions are handled by MetaMask through an Ethereum wallet, so that they are not centralized in any virtual point of sale, the application does not manage any bank information. In addition, blockchain wallets use asymmetric cryptography for transactions in order to prevent currency theft attacks.

The protection of blockchain integrity through hash functions makes it virtually impossible to maliciously alter any data from a DApp hosted on it. The decentralization feature of blockchain technology makes any distributed denial-of-service (DDoS) attack computationally very difficult, since if a DDoS is launched by only one or a few nodes, it would not change anything. It would have to be launched by half plus one nodes to be successful. Therefore, DApps based on distributed networks like this one are less exposed to these attacks.

In addition to the inherent security provided by blockchain technology, remote tracking allows knowing the location of the vehicle and the use that the driver gives it. Besides, the remote lock/unlock system allows the car to be locked in the event that a lessor does not return the car when the lease expires.

6 Conclusions and Future Works

This work describes the design of a decentralized car rental system based on smart contracts in the Ethereum public blockchain. In order to check the performance of the proposed system, this work also includes the implementation of the proposed rental car platform called AutoRent. However, since Ethereum blockchain charges fees for conducting some actions, the proposed solution was investigated only as a proof of concept. The developed smart contract gives the application autonomy and intelligence, as it is responsible for the rental and return of vehicles, and for the storage and management of payments, automatically distributing and charging each user what is indicated in the contract. Thanks to the combination with IoT and CPS, the proposal avoids the need for lessee and lessor to meet personally to exchange car keys or money. In addition, the proposed system preserves security and privacy for all parties involved and obliges them to comply with the conditions of the contract.

The proposed solution has some limitations and room for future improvement. For example, volatile relationships between the cost of gas in Ethers and the value of Ethers force a constant revaluation of operating cost at different

times. In addition, since Ethereum is a public blockchain, it is necessary to study how much information is available about the smart contract and its operations. On the other hand, the current line of work will allow progress in the development of the proposed platform, providing AutoRent with additional functionalities. In the current implementation, the developed application uses Ethers as cryptographic tokens and the PoW protocol for transactions. However, the plan is to use Ethereum 2.0 and the PoS protocol in the near future to get around some of the limitations of the proposal. That improvement will help significantly reduce the cost associated with transaction fees and increase the number of possible transactions per second.

Acknowledgment

This work was supported by the Spanish Ministry of Science, Innovation and Universities, the State Research Agency and the European Regional Development Fund under the Project RTI2018-097263-B-I00.

References

1. Szabo, N.: Formalizing and Securing Relationships on Public Networks. First Monday, 2(9). https://doi.org/10.5210/fm.v2i9.548 (1997)
2. Swan, M.: Blockchain: Blueprint for a new economy. O'Reilly Media, Inc. (2015)
3. Dannen, C.: Introducing Ethereum and Solidity (Vol. 1). Berkeley: Apress. (2017)
4. Vujicic, D., Jagodic, D., Randic, S.: Blockchain technology, bitcoin, and Ethereum: A brief overview. 2018 17th international symposium infoteh-jahorina, 1–6. (2018)
5. Bogner, A., Chanson, M., Meeuw, A.: A decentralised sharing app running a smart contract on the ethereum blockchain. Proceedings of the 6th International Conference on the Internet of Things, 177–178. (2016)
6. Huh, S., Cho, S., Kim, S.: Managing IoT devices using blockchain platform. 19th international conference on advanced communication technology, 464–467. (2017)
7. Reyna, A., Martín, C., Chen, J., Soler, E., Díaz, M.: On blockchain and its integration with IoT. Challenges and opportunities. Future generation computer systems, 88, 173-190. (2018)
8. Social Car, https://www.socialcar.com. Last accessed 27 June 2021
9. Cocoche, https://www.cocoche.com.ar. Last accessed 27 June 2021
10. GetAround, https://es.getaround.com. Last accessed 27 June 2021
11. Hassija, V., Zaid, M., Singh, G., Srivastava, A., Saxena, V.: Cryptober: A blockchain-based secure and cost-optimal car rental platform. IEEE International Conference on Contemporary Computing, pp. 1-6. (2019)
12. Niya, S. R., Schüpfer, F., Bocek, T., Stiller, B.: A Peer-to-Peer Purchase and Rental Smart Contract-based Application. it–Information Technology, 59, 9. (2017)
13. Ren, P., Xu, J., Wang, Y., Ma, X.: Research and Implementation of Car Rental Alliance Based on Blockchain and Internet of Vehicles. Journal of Applied Sciences, (6), 10. (2019)
14. Valastin, V., Kost'ál, K., Bencel, R., Kotuliak, I.: Blockchain based car-sharing platform. IEEE International Symposium ELMAR, pp. 5-8. (2019)

15. N. Saurabh, C. Rubia, A. Palanisamy, S. Koulouzis, M. Sefidanoski, A. Chakravorty, Z. Zhao, A. Karadimceand, R. Prodan. The ARTICONF approach to decentralized car-sharing. Blockchain: Research and Applications, vol. 100013. Elsevier, 2021.
16. HireGo, HireGo - Decentralised Shared Mobility Platform. Technical report. (2018)
17. Darenta Homepage, https://www.darenta.ru/en. Last accessed 6 March 2021
18. Helbiz Homepage, https://helbiz.com. Last accessed 6 March 2021
19. Copel, N., Ater, T.: DAV White Paper. Technical report. (2017)
20. FFQuest Homepage, https://ffquest.com. Last accessed 6 March 2021
21. WONO, Homepage, https://wono.io. Last accessed 6 March 2021
22. Mix.rent, https://mix.rent. Last accessed 27 June 2021
23. CAN-CONTROL, https://teltonika-gps.com/es/product/can-control. Last accessed 6 March 2021
24. Truffle Homepage, https://www.trufflesuite.com. Last accessed 6 March 2021
25. Web3, https://github.com/ethereum/web3.js. Last accessed 6 March 2021
26. Solidity, https://solidity-es.readthedocs.io/es/latest. Last accessed 6 March 2021
27. Ganache, https://www.trufflesuite.com/ganache. Last accessed 6 March 2021
28. Embark, https://framework.embarklabs.io. Last accessed 6 March 2021
29. InterPlanetary File System https://ipfs.io. Last accessed 27 June 2021
30. MetaMask, https://metamask.io. Last accessed 6 March 2021
31. Dirección General de Tráfico Ya se puede llevar el permiso de conducir en el móvil. https://www.dgt.es. Last accessed 27 June 2021

Appendix A

```
const Car = new Schema({
    model: String,
    price: Number,
    pricePerDay: Number,
    img: String,
    account: String
})
```

Listing 1.1. Car model with MongoDB

```
module.exports = {
    networks: {
        development: {
            host: '127.0.0.1',
            port: 7545,
            network_id: '*'
        }
    }
}
```

Listing 1.2.Configuration of the Truff le framework

```
var Temp = artifacts.require('rentacar')

module.exports=  function (deployer) {deployer.deploy(Temp)}
```

Listing 1.3.Truffle artefact example

```
axios.get('drivingLicence.json').then(response => {
    myContract2.contracts.Contract = TruffleContract(response.
        data)
    myContract2.contracts.Contract.setProvider(myContract.
        web3Provider)
})
```

Listing 1.4.Smart contract deployment

```
function rentCar (string _user, string _address, string
    _nPermit, uint _phone, string _drivingLicence permit =
    _drivingLicense(0x345ca3e014aaf5dca488057592ee47305d9b3e10
    );
require (permit.getLicense(_nPermit));
require(!cars[_id].rented);
cars[_id] = Car({
    priceCar: _priceCar,
    priceDaily: _priceDaily,
    rented: true
});
clients[msg.sender] = Client({
    user: _user.
    address: _address,
    nPermit: _nPermit,
    phone: _phone,
    deposit: 0,
    charges:0
});
)
function returnCar (string _id){
    require(clients[msg.sender].charges == 0);
    msg.sender.send(clients[msg.sender].deposit -
        ownerBenefits);
    owner.send(clients[msg.sender].deposit);
    cars[_id].rented = false;
    clients[msg.sender].deposit = 0;
    owner.send(this.balance);
    ownerBenefits = 0;
}
```

Listing 1.5. RentCar and ReturnCar functions

```
backCar: async function (identificator){
    id_ = identificator
    account = web3.eth.defaultAccount
    self = this
    let instance = await myContract.contracts.Contract.
        deployed()
    await instance.returnCar(id_, {from: account, gas
        :2900000})
    axios.post('/car/'+id_+'', {account: ''}).then(function (
        response) {})
    location.reload()
}

getDebt: async function() {
    self = this
    account = web3.eth.defaultAccount
    let instance = await myContract.contracts, Contract.
        deployed()
    let surcharges = await instance.getSurcharge.call({from:
        account})
    let value = web3.fromWei(surcharges.toString(), "ether")
    await $.get('/convertReverse/'+value+'',function(data) {
        self.$store.commit("changeDebt", parseInt(data.price))
        location.reload()
    })
}
```

Listing 1.6.Transaction and Call with Truffle framework

```
var store = new Vuex.Store({
    state: {
        debt: ''
    },
    getters: {
        getDebt: state =>{
            return state.debt;
        }
    },
    mutations: {
        changeDebt: (state,debt) => {
            state.debt = debt
        },
    }
})
```

Listing 1.7.Vuex states example

```
module.exports = {
    networks: {
        development: {
            host: '127.0.0.1',
            port: 7545,
            network_id: '*'
        }
    }
}
```

Listing 1.8.Configuration of the Cron package

```
async function convert (value) {
    let price = await cc.price('EUR', ['ETH'])
    let resultado = Object.values(price) * value
    return resultado
}
async function convertReverse (value) {
    let price = await cc.price('ETH', ['EUR'])
    let resultado = Object.values(price) * value
    return resultado
}
```

Listing 1.9.Cryptocurrency exchange with CryptoCompare

```
it("Bail can be in correctly", async() => {
    let instance = await RentACar.deployed()
    let transaction = await instance.depositBail({value: web3.
        toWei(1,"ether"), from: account})
    let expected = 1
    assert.equal(expected, transaction.receipt.status)
})

it("Bail can be in correctly", async() => {
    let instance = await RentACar.deployed()
    let call = await instance.getBail.call({from: account})
    let expected = 1000000000000000000 // 1 Ether in Wei
    assert.equal(expected, call.toString)
})
```

Listing 1.10.Code of the unit test of the bail