Towards Blockchain-based Ride-sharing Systems

Edgar Vazquez and Dario Landa-Silva
COL Lab, School of Computer Science, University of Nottingham, Nottingham, U.K.

Keywords: Ride-sharing, Blockchain, Smart Contracts, Smart Transport Systems.

Abstract: Blockchain technology has been used in finance, health care, supply chain, and transport, with the main goals of improving security and eliminating the need for a third party to manage transactions in the system. In blockchain, smart contracts are used to facilitate negotiation between stakeholders. The sharing economy movement has gained popularity in recent years in various sectors including transport. Ride-sharing has become an important component of sustainable transportation by increasing vehicle utilisation and reducing the number of vehicles on the road. Current ride-sharing systems are centralised with an intermediary maintaining users' data and managing transactions between drivers and passengers. This paper proposes the use of blockchain and in particular smart contracts, to develop decentralised ride-sharing systems. The benefits of having a distributed approach to maintaining users' data and managing transactions between users include more automation, more transparency, better data privacy, and possibly more trust between users.

1 INTRODUCTION

In the past few years, ride-sharing (also known as lift-sharing, car-sharing or car-pooling) has spread around the world with many people using this as their means of transportation (Clewlow and Mishra, 2017) mainly in USA. The main problems faced by this new technology include lack of real-time response, data privacy and community involvement. These problems are normally related to the centralised environment where the ride-sharing system is executed. This is because all data and pre-processing happens in a central server regulated by a central authority or intermediary. In some big and largely populated cities, where people spend long time commuting, ride-sharing has been adopted rapidly and used by commuters on a daily basis.

Many commuters use their own cars which brings several problems. These are: 1) time lost in traffic congestion (time/comfort cost), 2) fuel waste caused by driving for longer than necessary (economic cost), 3) higher pollution of the environment caused by gas emissions (environmental damage cost). Most vehicles on the streets are driven with single or low occupancy. This increases the number of vehicles on the road at a pace similar to that of the increase in the number of commuters. Around 3 billion gallons of fuel are used annually due to traffic congestion (Fagnant and Kockelman, 2014). This has an estimated cost of around \$ 160 billion or \$ 960 per commuter (Güneralp et al., 2017). It has been reported that mobility in the USA will increase by a factor of 2.6 by 2050 (Schafer and Victor, 2000).

Ride-sharing solutions can bring benefits like less wasted time, lower transportation costs, less pollution, and less stress. Improvements in ride-sharing systems can help to increase overall human productivity by reducing up to 40% the number of hours spent in traffic (Lanctot, 2017). In big cities like London, commuting time between home and work has increased by 72% over the past decade (Scott Le Vine and Humphrey, 2016). Ride-sharing systems can help to tackle traffic congestion by reducing the distance that vehicles are driven partially empty. It is also known that traffic congestion causes emotional stress (Buckley and O'Regan, 2003).

Centralised ride-sharing systems like *Bla Bla Car* have enjoyed some success for intercity ride-sharing thanks to features like flexibility in ride planning (in advance or same-day). Centralised ride-sharing systems have the downside of privacy risks due to data being shared and replicated often without the knowledge of users. Other issues with centralised ride-sharing systems are lack of transparency (e.g. in the fees charged) and regulatory conditions.

There is an opportunity in developing decentralised ride-sharing solutions. In a decentralised ride-sharing system there is no single source of truth, the

Vazquez, E. and Landa-Silva, D.
Towards Blockchain-based Ride-sharing Systems.
DOI: 10.5220/0010323204460452
In *Proceedings of the 10th International Conference on Operations Research and Enterprise Systems (ICORES 2021)*, pages 446-452
ISBN: 978-989-758-485-5

data is distributed across the network nodes. Instead, there is a network made of a lot of nodes connected to each other in a secure way, sharing and transacting data. Blockchain can help to provide a novel approach, in which an autonomous system can use data to match drivers to passengers and execute the transactions involved in the riding service. The blockchain can help to implement a safe and efficient relationship between passenger and driver while still allowing to use data for predicting common patterns and frequent trips.

Current ride-sharing systems could be vulnerable when transmitting information or processing it in the system. Blockchain could help with both the security of the application and stability of the system. Hence, incorporating a blockchain layer to the current ride-sharing framework could help to maintain anonymity among users and preserve integrity of the data.

In a typical client-server web application, all clients such as mobile phones, tablets or laptops make requests to a central authority (server) where all transactions are served. The server responds to all requests from the clients normally through an API (application programming interface). The API is responsible for extracting data from the central database and applying the business logic to it. Hence, a couple of problems arise as explained next:

- All the information of the users, such as names, personal address and payment data are saved in a database on the server. As a consequence, the data could be changed or removed entirely.
- The source code of the API can be updated at any time by anyone. This could result in bad practices, for example, the company could benefit some drivers more than others.

In this study, we propose a ride-sharing system combined with blockchain as a solution to tackling the excess of vehicles on the road. The methodology proposed uses blockchain technology from a new perspective, that is, a decentralised blockchain-based ride-sharing application. For this, we leverage the power of smart contracts to mitigate the problems of classic client-server architectures. The smart contracts are deployed on the Ethereum Network to guarantee the stability of the network.

We explore ways to mitigate the current problems of Blockchain technology, such as the slow transaction time for writing and reading on the blockchain and the high computational expenses per transaction for writing in Ethereum. This proposal is an effort to develop smart ride-sharing systems and contribute to the realisation of intelligent transportation systems for smart cities.

2 METHODOLOGY

For a ride-sharing system, benefits from using the blockchain can be realised in the transactions between drivers and passengers where usually an intermediary is required in centralised systems. This intermediary mechanism in systems like *Uber* guarantees that the driver is paid and that the passenger gets the ride. However, all rules and conditions for the operation of the system and the transactions between drivers and passengers are stipulated by the intermediary.

The proposed methodology uses *smart contracts* and a cryptography protocol in order to automate the transactions between drivers and passengers in a decentralised manner. The mechanism to eliminate the intermediary in blockchain is a *smart contract*, which is a self-executing contract, implemented in computer code, that includes the terms and conditions of the contract between the parties. Processes that run outside or within the blockchain are called *off blockchain* and *on blockchain* respectively. In the proposed methodology, the *smart contracts* execution are on blockchain while other processes are off blockchain as explained below. The protocol *ZKP (Zero-Knowledge Proof)* is a method where one actor (the prover) can prove to another one (the verifier) the knowledge of a certain value without revealing any information other than the fact that they know the value.

The process in the proposed decentralised ride-sharing system is as follows:

1. The passenger publishes a ride request on the blockchain through a smart contract. The pick-up and drop-off information is saved using *spatial cloaking*, a technique to blur a user's exact location into a spatial region in order to preserve privacy (Chow, 2008).
2. The matching algorithm runs off the blockchain to determine which drivers' destinations match the spatial location cloaked.
3. The matched drivers are invited to post a travel offer. The price is posted to all matched drivers in order to create a fair bid system. The route of each driver is encrypted using the public key of the passenger to ensure that only the passenger can see the location of the driver.
4. To promote trust between both passenger and drivers, the passenger picks the preferred offer, posts a ZKP-based smart contract with a deposit fee and posts a smart contract with a budget to guarantee the trip.
5. The driver posts a smart contract with a deposit fee as a commitment to the offer.

6. The smart contract acts as the verifier of the transaction. The smart contract uses ZKP to determine the outcome of the contract as follows:
   (a) If the passenger does not show up or cancels the ride request, the smart contract will auto-release the passenger's deposit fee to the driver.
   (b) If the driver does not show up at the time scheduled or cancels the ride offer, the smart contract will auto-release the driver's deposit fee to the passenger.
   (c) When the driver successfully validates arrival at the pick-up location and the passenger is there, the smart contract will auto-release the passenger's deposit fee to the driver as initial partial payment and the driver's deposit fee will be returned to the driver.
7. Once the ride starts, a new smart contract automatically transfers partial payments to the driver in lapses between 5 and 10 minutes as the travel progresses.
8. Once the ride is over, the smart contract works as the validator that the trip has been completed successfully and then it pays the service fee to the driver.

Since smart contracts only work with data in the blockchain, trusted entities validate external data before adding it to the blockchain. This validation is done outside the blockchain, hence data provided by the user, such as latitude and longitude, is assumed to be valid for the purpose of this paper (Cascudo and David, 2017).

The implementation architecture for the proposed system is based on the following:

- The blockchain network proposed here must be public and permissive, so that anyone can read and write transactions within the network. It must also allow the exchange of some currency and be able to execute smart contracts. Ethereum is the recommended network since it complies with these features.
- Passengers and drivers use their smartphone to communicate with the blockchain. The model assumes that in addition to GPS and internet connection, the phones support peer-to-peer wireless communication.
- Some location based service provider helps to ensure that the driver has reached the ride starting point.

The four key mechanisms in the above methodology are described next. These are: generating trip data, smart deposit contract, matching process and smart transparent payment contract.

2.1 Generating Trip Data

To protect user's privacy, we implement cloaking, a generalisation technique that obfuscates precise location information by considering spatial regions. (Amiri et al., 2019). Also, the exact departure and destination times of the trip are generalised in five-minute intervals. Basically, cloaking has the effect of obfuscating the user's location information.

For driver d, the ride offer can be represented by (1) where $(L_0^{(d)}, T_0^{(d)})$ represents the location and time of the trip origin and $(L_n^{(d)}, T_n^{(d)})$ represents the location and estimated time of arrival of the trip destination. In between, there is a sequence of intermediate locations and their corresponding times. This sequence of pairs (location,time) correspond to points in the planned route for driving from the origin to the destination.

$$R^{(d)} = \left\{ (L_0^{(d)}, T_0^{(d)}), \ldots\ldots, (L_n^{(d)}, T_n^{(d)}) \right\} \quad (1)$$

The driver d generalises his exact location and time of the trip by (2) where $(C_0^{(d)}, T_0^{(d)})$ represents the generalised information of $(L_0^{(d)}, T_0^{(d)})$. This simply means that $A^{(d)}$ corresponds to the obfuscated version of $R^{(d)}$.

$$A^{(d)} = \left\{ (C_0^{(d)}, T_0^{(d)}), \ldots\ldots, (C_n^{(d)}, T_n^{(d)}) \right\} \quad (2)$$

For passenger p, the ride request can be denoted by (3) where $(L_0^{(p)}, T_0^{(p)})$ represents the location and time of the trip origin and $(L_1^{(p)})$ represents the location of the trip destination.

$$R^{(p)} = \left\{ (L_0^{(p)}, T_0^{(p)}), (L_1^{(p)}) \right\} \quad (3)$$

Similar to the driver, the passenger's information is generalised by (4) so that $A^{(p)}$ represents the obfuscated information.

$$A^{(p)} = \left\{ (C_{orig}^{(p)}, T_{orig}^{(p)}, C_{dest}^{(p)}) \right\} \quad (4)$$

All the above trip data is generated off the blockchain.

2.2 Deposit Smart Contract

Algorithm 1 shows the procedure for the deposit smart contract. The class first initialises the basic variables for the operation of the contract. These include the blockchain addresses for the driver and the passenger. In order to maintain anonymity, the passenger and driver are auto-assigned a pair of private and public temporary keys $(K_{private}, K_{public})$ and an address within blockchain (lines 3-4). The array A_i in line 8 contains the blockchain addresses representing the signatures of elements in the set. Then, when the smart contract is created, the constructor in line 9 assigns the driver's address to the contract and initialises the rest of the variables (the generalised location information $C^{(p)}_{orig}, C^{(p)}_{dest}$ and time $T^{(p)}_{orig}$).

Lines 14-19 implement the DriverDeposit procedure which first validates that the driver arrived on time at the place of origin for the trip and that the trip has not yet expired. Once the validation is complete, the deposit is assigned to the contract. Lines 20-23 implement the FineForDriver procedure for the case of the driver not arriving on time, hence returning the balance to the passenger. Finally, lines 24-27 implement the ProofOfArrival procedure which validates that the driver reached the origin. This is where the ZKP function is executed to validate the arrival time and location. If the validation is successful, the driver receives the fee for the service. Also during the proofOfArrival procedure, the passenger is notified using web sockets, so that the smart contract can evaluate if the trip request matches their path $(A^{(p)} \in A^{(d)}_k)$.

Finally, the driver uses the passenger's public key to encrypt all the information about the trip. This information will be useful for the next phase during the ride.

2.3 Matching Process

In order to find a match between the passenger and drivers, three algorithms from the literature have been compared. These are: (a) a Distance Greedy Heuristic approach, (b) a Time Greedy Heuristic approach and (c) a Data Structure approach.

The three algorithms have been configured with the following restrictions: (1) vehicle capacity, (2) maximum passenger wait time, and (3) maximum detour for the ride request. The objective function (5) seeks to minimise passenger timeout (Jaeyoung Jung and Park, 2012). Here, T_p is a fine for a dropped request, P^r is the number of rejected requests, $TT(P^c_i)$ passenger travel time and $WT(P^c_i)$ is the passenger waiting time.

$$C = T_p \cdot P^r + \sum_{i \in I} TT(P^c_i) + \sum_{i \in I} WT(P^c_i) \qquad (5)$$

The *Distance Greedy Heuristic* simply finds the available vehicle that is closest to the trip origin (Tao and Chen, 2007). The *Time Greedy Heuristic* finds the vehicle that minimises waiting and travel times (Santos and Xavier, 2013). The *Data Structure* approach is more elaborate as it generates nodes for the ride offers and then finds the shortest route for the trips (Xingshen Song and Jiang, 2019). The algorithm also uses the inverted index strategy to organise and access the data efficiently.

Algorithm 1: Pseudocode for Deposit Smart Contract.

```
1:  contract Deposit
2:      uint public Balance                     ▷ The Balance for deposits
3:      int64 address passenger
4:      int64 address driver
5:      uint public driverD                     ▷ The driver deposit
6:      uint public passengerD
7:      address [] location
8:      address [] $A_i$
9:      procedure CONSTRUCTOR($driver, location, A_i$)
10:         $this.driver \leftarrow driver$
11:         $this.location \leftarrow location$
12:         $this.A_i \leftarrow A_i$
13:         $this.driverD \leftarrow driverD$
14:     procedure DRIVERDEPOSIT($driverD$)
15:         if $block.timestamp \geq expiration$ return;
16:         if $msg.sender \neq driverAddress$ return;
17:         if $msg.value \neq driverD$ return;
18:         if $now \geq acceptanceDeadline$ return;
19:         $driverD \leftarrow driverD$
20:     procedure FINEFORDRIVER($driverD$)
21:         if $block.timestamp \leq expiration$ return;
22:         if $msg.sender \neq passenger$ return;
23:         $transfer(balance, passenger)$      ▷ Send back the money
24:     procedure PROOFOFARRIVAL($\sigma$)      ▷ Where $\sigma$ is lat, lng
25:         if $msg.sender \neq driverAddress$ return;
26:         if $ZeroKnowledge.Verifier(\sigma)$ then
27:             $transfer(balance, driver)$     ▷ Pay fare to driver
28:
```

2.4 Payment Smart Contract

The smart contracts can also be used to implement a more transparent and dynamic payment system. It has been shown than in some conventional ride-sharing systems dishonest drivers can report longer distances than the actual travelled distance (Zhao et al., 2018). There can also be issues if the full fee is paid at the start or the end of the trip (Singh et al., 2020).

The proposed payment smart contract ensures that the travelled distance is sent periodically by the driver through the passenger's private key. Once the passenger confirms the travelled distance, the process

ProofOfDistance in algorithm 2 releases the fee to the driver. Hence, the driver is paid only for the elapsed journey at any time and the corresponding partial information is saved in the blockchain. The variables and functions within the payment smart contract are very similar to those of the deposit smart contract.

Algorithm 2: Pseudocode for Payment Smart Contract.

```
1:  contract Payment
2:      address payable passenger
3:      address payable driver
4:      uint public distance
5:      procedure CONSTRUCTOR($driver, distance$)
6:          $this.driver \leftarrow driver$
7:          $this.distance \leftarrow distance$
8:      procedure PROOFOFDISTANCE($distance$)
9:          if $msg.sender \neq riderA$ return;
10:         while $totalDistance \leq distance$ do    ▷ Pay only for distance traveled
11:             $transfer(balance * distance, driver)$
12:             $totalDistance \leftarrow this.totalDistance - distance$
13:     procedure GETFUNDS    ▷ Return the money to the passenger in case the waiting time runs out
14:         if $current.block.timestamp \leq expirationTime$ return;
15:         if $current.block.owner \neq message.sender$ return;
16:         $transfer(balance, passenger)$    ▷ Send back the money to passenger
17:
```

3 EVALUATION EXPERIMENTS AND RESULTS

Preliminary experiments were conducted in order to evaluate the performance of the proposed decentralised ride-sharing approach based on blockchain smart contracts. We implemented the smart contracts in the Ethereum network and then executed some experiments to evaluate computational effort and response time.

3.1 Computational Effort

Running blockchain calculations such as Proof of Work (PoW) or in this case zero-knowledge proof (ZKP) requires a lot of processing power due to the complex mathematical and cryptographic calculations.

In Ethereum, the term *gas* is used to quantify the fixed cost of performing each transaction in the blockchain. This mechanism is not different from the used by cloud computing companies such as Google Cloud Platform (GCP) and Amazon Web Services (AWS). For example, for an *add* operation Ethereum charges 3 gas and for a *multiplication* it charges 5 gas (Wood et al., 2014).

Two costs are involved in measuring the overall cost for the on blockchain operations in the proposed system. *Transaction cost* is the gas cost of sending information to the blockchain. It includes the base cost, the cost of contract deployment and the cost of every byte. *Execution cost* is the gas spent on running the code on the blockchain.

Currently in traditional ride-sharing systems, the driver must pay a commission of approximately 25% of the trip fee depending on the platform used. For the proposal to be viable, we assume that the price for each trip in dollars must be below the 25% commission. At present, 1 ether = 1,000,000,000 gwei (10^9) or 1 gwei = 0.000000001 ether. The Ether price is $206 as of May 24th, 2020 (Gas,). Then, 1 million gas is approximately 2 dollars. For testing purposes, the smart contracts have been pre-compiled and executed at run time, so that contracts use less gas.

The purpose in this subsection is to measure the cost of executing the proposed smart contracts within the Ethereum network. This will allow to assess the economic viability of running a decentralised ride-sharing within the blockchain as proposed here.

We generated 10 trips requests, all with the same origin and destination. For each trip there is always only one driver available within the travel area. The first 5 trips were analysed from the driver's perspective and the other 5 trips from the passenger's perspective. The cost for each smart contract process on the blockchain was measured. This was done by setting breakpoints before and after the 4 main driver operations on the blockchain.

Figure 1 and Figure 2 show the average cost in terms of gas for the smart contract procedures of the driver and passenger respectively. Transferring the deposit to the driver and publishing a request are the most expensive processes with almost 900K gas consumption. The average total of computing a trip is 5,890,000 gas, which converted to dollars it would be approximately 12 USD.

3.2 Response Time

The response time is influenced by the speed at which the matching is done to identify suitable drivers for a ride request. In order to evaluate the matching algorithms, 5 different sample data sets were generated. A data sets includes a ride request and a number of

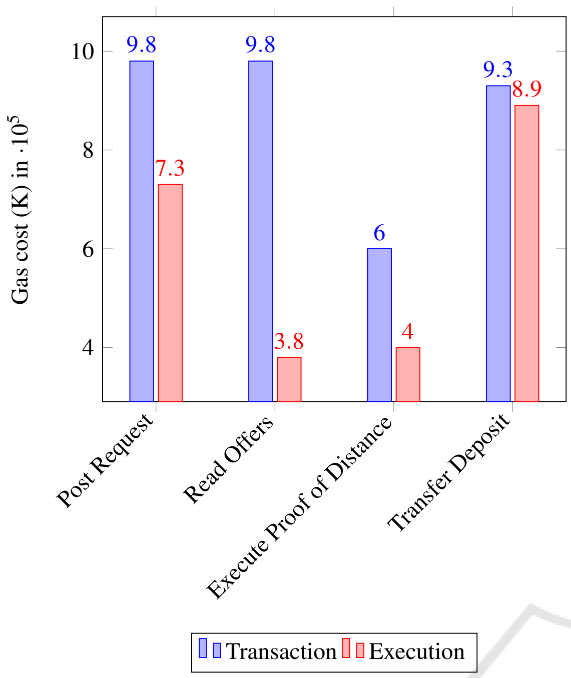

Figure 1: Average cost of executing the driver smart contract procedures in Ethereum.

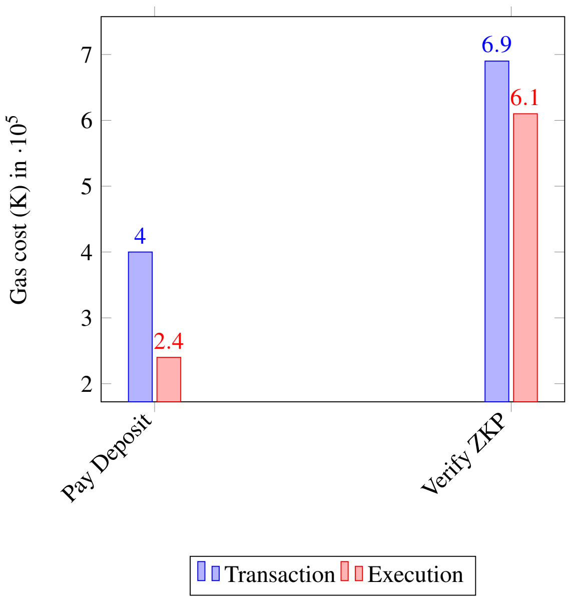

Figure 2: Average cost of executing the passenger smart contract procedures in Ethereum.

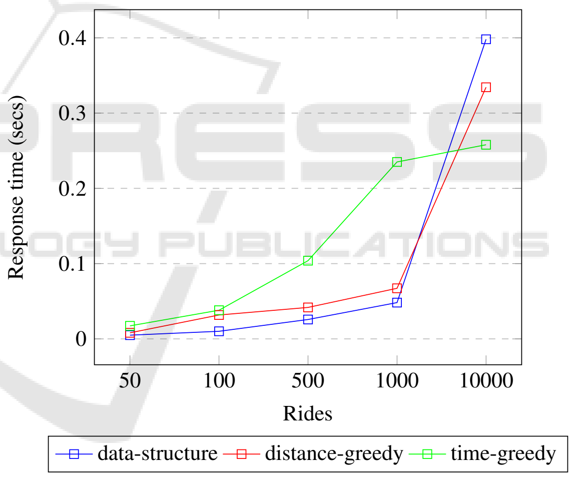

Figure 3: Response time performance of the matching algorithms running off blockchain.

ride offers from drivers, this number can be 50, 100, 500, 1000 or 10,000. This means that for a passenger posting a ride request, the number of drivers in the system can be as small as 50 or as large as 10,000, hence influencing the speed at which a match can be found. For each of the 5 data sets, the three matching algorithms outlined in subsection 2.3 were executed off the blockchain. This means that here we only compare which matching algorithm performs better on finding a match for the given ride request.

Figure 3 shows that response times increase with the number of ride offers or drivers, but even for the 10,000 case the time does not exceed 0.4 seconds. For cases of smaller number of ride offers (50 and 100) the three algorithms perform similarly. Moreover, distance-greedy and data-structure exhibit similar performance across all cases, while for the larger 10,000 case, time-greedy algorithm is faster.

We decided to use time-greedy in the complete process and now focus on evaluating the response time for the on blockchain and off blockchain operations. Table 1 shows the response times for each of the steps in the proposed approach and using time-greedy for the matching. After adding the off blockchain and on blockchain processes we can see that the complete process takes from around 4 seconds to little over 7 seconds depending on the number of drivers or ride offers. It is important to note that the proposal here is to use Ethereum since the mining process takes seconds instead of minutes as happens in Bitcoin blockchain for example.

4 CONCLUDING REMARKS

Ride-sharing can help to make mobility smarter and transport more efficient for communities around the world. The increasing number of smartphones, the continuous growth of internet access and the high vehicular density in cities have made solutions such as ride-sharing more appealing. Existing ride-sharing systems use a centralised approach where a third party

Table 1: Response time for each transaction step processing a ride request while using a time-greedy heuristic for the ride matching.

Transaction Steps	50 Drivers	100 Drivers	500 Drivers	1000 Drivers	10000 Drivers
(1) Generate ride data	0.88128	0.622171	0.58353	0.865349	1.785475
(2) Post ride request	1.366758	1.539878	1.685635	1.419069	1.967857
(3) Post ride offer	1.408167	2.577480	1.733291	1.907547	1.227874
(4) Verify ZKP	0.081271	1.029151	1.964220	2.659049	1.888619
(5) Find ride match	0.008072	0.031726	0.100041	0.267208	0.294342
Total time (secs)	3.745548	5.800406	6.066717	7.118222	7.164167

manages data and transactions between drivers and passengers. This paper proposes the idea of implementing ride-sharing systems in a decentralised manner by using blockchain smart contracts.

The decentralised ride-sharing approach outlined here was implemented in Ethereum and experiments were conducted to evaluate the feasibility of the proposed solution. Smart contracts are implemented to execute deposits by drivers and passengers as well as payment as the ride service goes on. It is argued that these mechanisms help to improve transparency of transactions, privacy of users data and accountability in the system. A deposit smart contract guarantees against drivers or passengers not keeping their side of the ride agreement. The payment smart contract implements partial payment of the fee as the trip progresses and ensures the full payment once the ride service is completed. All this implemented in a secure manner and without the need of an intermediary or 3rd party as in traditional centralised ride-sharing systems.

REFERENCES

ETH gas station. https://ethgasstation.info/. Accessed: 2020-05-25.

Amiri, W. A., Baza, M., Banawan, K., Mahmoud, M., Alasmary, W., and Akkaya, K. (2019). Privacy-preserving smart parking system using blockchain and private information retrieval.

Buckley, F. and O'Regan, B. (2003). The psychological effects of commuting in dublin. *Buckley, Finian and O'Regan, Brendan (2004) The psychological effects of commuting in Dublin. LInK Working Paper Series. (Paper No. 07-04). The Learning, Innovation and Knowledge Research Centre, Dublin City University, Ireland.*

Cascudo, I. and David, B. (2017). Scrape: Scalable randomness attested by public entities.

Chow, C.-Y. (2008). *Cloaking Algorithms for Location Privacy*, pages 93–97. Springer US, Boston, MA.

Clewlow, R. R. and Mishra, G. S. (2017). Disruptive transportation: The adoption, utilization, and impacts of ride-hailing in the United States. resreport UCD-ITS-RR-17-07, Institute of Transportation Studies, University of California, Davis.

Fagnant, D. J. and Kockelman, K. M. (2014). The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios. *Transportation Research Part C: Emerging Technologies*, 40:1–13.

Güneralp, B., Zhou, Y., Ürge Vorsatz, D., Gupta, M., Yu, S., Patel, L., Fragkias, M., Li, X., and Seto, K. (2017). Global scenarios of urban density and its impacts on building energy use through 2050. *Proceedings of the National Academy of Sciences*, 114:8945–8950.

Jaeyoung Jung, R. J. and Park, J.-Y. (2012). Design and modeling of real-time shared-taxi dispatch algorithms.

Lanctot, R. (2017). Accelerating the future: The economic impact of the emerging passenger economy. Technical report, Strategy Analytics.

Santos, D. and Xavier, E. (2013). Dynamic taxi and ridesharing - a framework and heuristics for the optimization problem. pages 2885–2891.

Schafer, A. and Victor, D. G. (2000). The future mobility of the world population. *Transportation Research Part A: Policy and Practice*, 34(3):171–205.

Scott Le Vine, J. P. and Humphrey, A. (2016). Commuting trends in england 1988 - 2015.

Singh, G., Dwesar, R., and Kumar, S. (2020). Uber's bumpy ride in china. *The CASE Journal*, ahead-of-print.

Tao, C. and Chen, C. (2007). Heuristic algorithms for the dynamic taxipooling problem based on intelligent transportation system technologies. In *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)*, volume 3, pages 590–595.

Wood, G. et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32.

Xingshen Song, Yuexiang Yang, Y. J. and Jiang, K. (2019). Optimizing partitioning strategies for faster inverted index compression.

Zhao, S., Luo, X., Ma, X., Bai, B., Zhao, Y., Zou, W., Yang, Z., Au, M. H., and Qiu, X. (2018). Exploiting proximity-based mobile apps for large-scale location privacy probing.