INTERNATIONAL JOURNAL OF INFORMATION SECURITY SCIENCE
O. Oksuz, Vol.13, No.1, pp.1-22
https://doi.org/10.55859/ijiss.1399189

A Smart Contract Based Secure Ride Sharing System

Ozgur Oksuz[1]

[1]Faculty of Engineering and Natural Sciences, Department of Software Engineering,
Konya Technical University, Konya, Turkey
Corresponding Author: ooksuz@ktun.edu.tr

Research Paper Received: 01.12.2023 Revised: 17.01.2024 Accepted: 26.01.2024

Abstract—A ride-sharing system provides many advantages. It reduces energy consumption. Moreover, it mitigates traffic congestion. Furthermore, it provides benefits to drivers and passengers in that they share travel costs (gas, toll ticket). Ride-sharing has been getting very popular since the COVID-19 pandemic. Since COVID-19 is very contagious and spread by infected people via coughs, sneezes, or talking, people avoid using public transportation to get the disease. People prefer to travel with a few people to protect their health. To prevent people from getting the disease and to have the advantages of a ride-sharing system, we introduce a ride-sharing system in which a driver only travels with a passenger for each event. Traveling with only one person provides advantages to both the driver and passenger. The driver (passenger) not only shares the cost of the trip but also reduces the risk of getting or spreading the disease. This paper proposes a smart-contract-based ride-sharing system that uses a $Vickrey$ (second highest bid) auction mechanism during a pandemic to determine the passenger. The system provides a decentralized, transparent, trackable, verifiable, and secure ride-sharing with the help of consortium blockchain. Using smart contracts in the system allows users' bids to be transparent and verifiable. Furthermore, the proposed system provides a secure and lightweight mechanism to protect users' (drivers and passengers) travel data (locations and cost of travel). Their travel data for each event is going to be unlinkable. Since any information in transaction is not encrypted, transaction data (travel data) can be used for data mining and machine learning to extract useful information such as determining the frequently used destinations.

Keywords—COVID-19, Privacy, Ride-sharing, Smart-contract, Unlinkability, Vickrey

1. Introduction

With the COVID-19 disease transmitted from one person to another via cough, sneezing, and talking, people avoid crowds and keep their distance from others. People prefer to use their private cars rather than use public vehicles to go from one place to another to avoid being sick. However, there are some disadvantages for each person to use their vehicle for traveling: It increases the carbon footprint in the environment. The cost of travel (gas, tolls, tickets) of the person increases. Traffic congestion and traffic jams happen so that the person's duration of travel increases. A ride-sharing system is a solution for all of the above. The ride-sharing system helps decrease total CO2 emissions. It also provides benefits to the users (drivers, passengers). People share travel costs using a ride-sharing system. Moreover, ride-sharing

reduces traffic congestion.

However, ride-sharing has some problems to solve. One of the problems is that a driver or passengers do not want to travel with many people since they worry about getting infected in a pandemic (COVID-19). In other words, they want to feel safe and comfortable when they travel. Another problem of a ride-sharing system is that users can misbehave. They can cheat each other. The driver or the passenger may not show up at the time of travel. If one of these parties gives up traveling, the other party is badly affected (time and budget). This results in time and budget-consuming for the users. There should be a mechanism to protect users' time and their budgets. In the system, there should be a punishment for fraudulent users. Another problem in ride-sharing systems is that a passenger wants to travel with a legitimate driver and should be convinced that the driver's car is in good condition. The driver also needs to be convinced that the passenger is legitimate. The ride-sharing system should protect users' data privacy from any malicious entity. The travel data consists of users' locations (starting point, end point), their identities, and the cost of the travel. If there is no protection for the travel data, an untrusted party easily tracks the users.

Ride sharing data can be useful. Using machine learning and data mining techniques on travel data extract information for some real-world applications. For example, detecting the frequently used destinations. Determining such places can help to build police stations to provide security, healthcare services to deal with treatment of disease and restaurants to serve entities foods and drinks. Another application can be extracting the travel time between two places based on the starting and ending times of the travel from the transaction data. This could help re-adjusting travel costs.

To protect users' privacy, a system can allow entities to encrypt their sensitive information such as identity, start and end locations, and cost of travel. However, using encryption algorithms to hide sensitive information affects the speed and efficiency of data processing and analysis, and limit the visibility and usability of data. However, it does not allow entities to use data for data mining and machine learning since the data is encrypted.

To have a secure system, the existence of a central trusted party in the system can manage all of these properties. However, the trusted party can be hacked by a dishonest party. It results that the system has single point of failure.

To address all the problems above, we propose a blockchain and smart contract-based ride-sharing system. Applying blockchain technology to ride-sharing systems eliminates all these problems above. It is a decentralized system that does not rely on a single party. It has a ledger technology that consists of verifiable, transparent user data and is available to all users in the system.

The contributions of this paper is as follows:

- In the proposed system, drivers (or passengers) want to travel with one person to protect their health. The system uses $Vickrey$ (second price) auction to determine the passenger. The use of $Vickrey$ auction provides entities a transparent and secure auction mechanism. It protects users' data (bid) privacy.
- We use smart contracts to solve conflicts between drivers and passengers. Using smart contracts forces users to obey the rules. If the driver or passenger does not show up at the time of the travel, the party who suffers should get a bonus, and the other party who does not obey the rules will be punished. Moreover, using blockchain structure provides transparency in the authentication of users (drivers, passengers),

travel costs, and integrity of the ride-sharing service. Furthermore, to tackle any dispute between users and protect users' time and budget in the ride-sharing system, each user pays a collateral (deposit) to the smart contract. If any fraudulent driver (passenger) exists, the other party passenger (driver) gets the collateral.

- The $Vickrey$ auction provides benefits to both parties. The driver gets at least half of the travel cost (gas, tool fares). The passenger should not pay all travel costs. He pays at least half of the travel cost.
- The proposed system does not implement heavy cryptographic operations like encryption to protect users' travel data privacy. The system uses a secure randomization technique for their identities to protect users' travel data privacy. This method does not allow an untrusted party to link users' travel data.
- The proposed system allows entities to use transaction data for machine learning and data mining purposes to extract information to build real-world applications.

2. Related Work

There have been some studies that proposed ride-sharing systems using blockchain. In [1] uses proxy re-encryption mechanism to protect users' privacy. Moreover, it uses smart contracts to allow users to interact with each other. Another work in [2] also uses a proxy re-encryption scheme to share data with other users and protect the privacy of the users and uses anonymous signature to provide anonymity. The same work also uses smart contracts to provide interactions between users. In work [3] proposed a ride-sharing system in which drivers are also the miners in the system. This results in the drivers doing the heavy work and means that the drivers need to have strong computation capabilities. Besides this, there is no data protection mechanism for the drivers and passengers. The entities can be easily trackable by any third-party entity. In work [4], authors proposed an efficient ride-hailing system based on the $Ethereum$ blockchain. The system is called $PEBERS$. The authors demonstrated how a decentralized system based on a consortium blockchain can be developed to keep track of ride data. However, they did not focus on users' data privacy (travel data). The users' identities were not changed. Thus, any third party can keep track of any specific user. The authors in [5] use a private blockchain to provide a system that has a secure ride-sharing service. The study in [6] proposed a decentralized ride-sharing system. In their system, passengers (riders) request their ride-sharing information. Then, the drivers publish their offers. Then, there is a matching phase that comes into play. Once the matching step is over, the entities communicate with each other. They assume that both parties (driver and passenger) are honest. So, they did not examine if the parties were dishonest. There is no travel data privacy for the users. A malicious entity can map each transaction to a specific driver or rider. [7] proposed a ride-hailing system to know optimal pricing and hygiene level decisions in an outbreak. The authors focus on the safety of the people, not the privacy. They leave it as a future work in their paper. In [8], the authors proposed a privacy-preserving ride-matching scheme. The system provides matching between multiple riders and drivers. The system uses $Paillier$ encryption [9] to protect user location privacy. However, their work does not use a blockchain structure.

In another work in [10], the authors consider the case in which the platform is risk-averse and serves a market with both safety risk-averse and non-safety risk-averse customers. However, they do not focus on the data privacy of the users. Another work in

Table 1.
Symbols and their descriptions.

Symbol	Description	Symbol	Description
$Ride_{id}$	Identity of the Ride	H, HC, HF, H_l	Hash functions
CA	Central Authority	KG	Key Generation Algorithm
add_p	Address	$Sign$	Signing Algorithm
f	Pseudo Random Function	$Verify$	Verification Algorithm
ts	Time Stamp	$RevTime$	Bid Reveal Time
pk	Public Key	bid	Bid
HB	Hash Function	$StartLoc$	Start Location of User
$dpst$	Deposit	$EndLoc$	End Location of User
$ComTime$	Commitment Time	sk	Signature Secret Key
F	Pseudo Random Generator	$StartTime$	Start Time of Travel
$HigBid$	Highest Bid	$SecHigBid$	Second Highest Bid
$StartPoint$	Start Location of Travel	$EndPoint$	End Location of Travel
$Winadd$	Address of Winner	r	Random number

[11] proposed a blockchain smart contract privacy-preserving ride-sharing system that uses encryption to protect users' location privacy. The users (drivers and passengers) encrypt their location information and put it into the blockchain. Then, a ride-matching algorithm is executed by the smart contract. Recent work in [12] proposed a secure ride sharing based on private smart contracts that users' privacy is obtained by using short group signatures and encryption. Users encrypt their transactions and send them to a local roadside unit (RSU). RSU then decrypts the ciphertexts and matches riders and drivers. Then, the rider and the driver communicates via cell phones to negotiate a specific pick-up location. Using an encryption algorithm results in entities perform too much computation and the transaction data is not suitable for machine learning and data mining purposes. Another recent work in [13] proposed a privacy-preserving ride-matching scheme for ride-sharing services in a hot spot area. In work [13] uses homomorphic encryption ($Paillier$) to preserve location privacy. However, their work is not based on blockchain and does not consider ride-sharing in a pandemic.

In our system, the entities do not use any encryption algorithm to send transactions to RSU. Location privacy is obtained by using an anonymization technique over the public keys and addresses of the users. Moreover, the system provides fast authentication for the users. Thus, the proposed protocol is more efficient than the other studies. The matching algorithm is done by executing $Vickrey$ by the smart contract. Furthermore, the proposed work focuses on ride-sharing in a pandemic.

Another work in [14] proposed a blockchain based package delivery via ride-sharing system. The authors proposed a hash oriented $PBFT$ consensus algorithm to reduce the confirmation delay. Authors also examined the system's resilience against sybil and double spend attacks. However, the work does not have any trust model and focus on users' (requesters) data privacy and anonymity of the users (requesters). A requester generates a

package delivery transaction message that includes the origin of the packages delivery, the destination of the packages delivery. The transaction also includes the signature of the requester. Since the requester's uses the same identity for each request, their study does not satisfy unlinkability property. The authors in [15] proposed an extension of their previous work in [14] to elaborate the distributed dynamic ride-sharing matching algorithm to further complete the delivery system. In study [16], the authors proposed proposes a blockchain-Enabled Shared Mobility ($BESM$) architecture that involves smart city authorities, vehicle owners, hospital management, and residents/tourists to decide on collaboratively utilizing a shareable vehicle in smart cities. In $BESM$, even though a traveler books a seat, the feasibility of accommodating the passenger in the shareable vehicle is cross-checked by other participants in the blockchain network, including smart city officials, vehicle owners, and hospital authorities. $BESM$ collaboratively allocates seats to travelers in a shareable vehicle depending on the air quality and $COVID-19$ cases of traveling destinations. The air quality of cities are predicted by using some machine learning algorithms. The authors uses a permissioned blockchain to provide immutability database and transferring route plans across organizations. However, the study did not give any information about threat model for the system and data privacy for the entities and unlinkability of the transactions. Authors in [17] proposed a ride sharing system that the route of the driver is updated once a new rider is added to the route. In the beginning, the riders and drivers submit their itineraries to the system. A smart contract is utilized to match the driver with the riders having compatible routes. The driver offers a plan of its travel that includes the riders to pick them up. Thus, once the driver picks a new rider, it updates its travel. However, it their work, they did not give any security model for the system. It did not examine users' (driver and rider) data privacy and unlinkability of the transactions. In study [18] proposed a blockchain-based framework for a ride-sharing service. The authors then implemented their prototype of the framework as a decentralized application (DApp). The prototype is based on smart contracts on $Ethereum$ blockchain. The authors also provided some algorithms to match riders with drivers to decrease travel distance. However, in their work, the users' (drivers and riders) data privacy (travel data), unlinkability of users' multiple events and security model of the system were not mentioned.

The given Table 2 shows the comparison of our system and other studies. in the table, $Unlinkability$ property represents if multiple events of the same passenger or driver are not linkable, $Machine\ Learning$ property represents if the transaction data in the corresponding work can be applied to machine learning and data mining. $Privacy$ property shows if the users' data is protected, $Method$ property is for if the study uses a protection mechanism of the users' data. If data of the users are not protected in the system, we label the corresponding study's cell as $No\ Method$. $Blockchain$ shows the used blockchain type in the corresponding study.

3. Definitions

In this section, we give some definitions used in this paper.

3.1. Ride Sharing

A carpooling system focuses on reducing traffic jams and the amount of energy like gas and electricity. It also provides benefits for drivers and passengers. Drivers and passengers share travel costs. The passengers have comfortable and fast travel.

Table 2.
Comparison of the proposed system and other studies (blockchain-based).

	Unlinkability	Machine Learning	Privacy	Method	Blockchain
[1]	No	No	Yes	Encryption	Consortium
[2]	Yes	No	Yes	Encryption	Private
[4]	No	Yes	No	No Method	Private
[6]	No	Yes	No	No Method	Public
[12]	No	No	Yes	Encryption	Private
[14]	No	Yes	No	No Method	Public
[16]	No	Yes	No	No Method	Permissioned
[17]	No	Yes	No	No Method	Public
[18]	No	Yes	No	No Method	Public
Our Work	Yes	Yes	Yes	PR	Consortium

3.2. Blockchain

Blockchain is a decentralized and peer-to-peer data storage that eliminates a central party. In the network, each node stores a copy of a ledger. This ledger consists of all the transactions between the nodes. Once the transactions are put into a block, they are immutable. There are three types of blockchain: Public, Private and Permissioned. In a public blockchain, any participant can join (leave) (from) the network as a node without having any permission from any entity. Bitcoin [19] and $Ethereum$ [20] are examples of public blockchain. Any participant node can issue a transaction and read from the block. The blockchain is open to everyone. All transactions are verified using a consensus mechanism. If a transaction is valid, it is added to the block by the honest nodes. The private chain is used inside the enterprise. This type of chain provides a minimum degree of decentralization. A permissioned or consortium blockchain is in between public and private blockchains. In consortium blockchain, some of the authorized participants can issue transactions. Moreover, the other authorized entities can freely read and view the transactions. If entities are known in advance, a consortium blockchain is used.

In consortium blockchain, a formed block consisting of transactions needs to be checked and verified by other nodes before it is added to the blockchain. This verification mechanism in the system is called consensus. There have been several consensus algorithms proposed and used: Proof of Work (PoW), Byzantine Fault Tolerant (BFT), Practical Byzantine Fault Tolerant ($pBFT$), Voting based $PBFT$ ($vPBFT$), and so on.

3.3. Smart Contracts

A smart contract is an automated and deterministic computer program consisting of functions and parameters. These functions are executed once the predefined requirements are satisfied. When a smart contract is deployed, it is stored in the blockchain. The function outputs the corresponding result. Once a smart contract is executed, it results in a state change or a new transaction in the blockchain.

$Ethereum$ [20] is the most used platform for developing smart contracts.

3.4. Vickrey Auction

$Vickrey$ auction is also known as the second price auction mechanism. $Vickrey$ auction consists of two phases: commit and reveal. In the commit phase, users (passengers) submit their bids in a predefined period. However, the users' bids are protected so that anyone can not see other users' bids. In the reveal phase, each user in a predefined period exposes their bids. The algorithm then outputs the winner with the highest bid and the second highest bid. The second highest bid is the price that the winner pays. Vickey auction is known to determine fair market prices and is used in many research areas like energy trading [21] and wireless networks [22].

3.5. Digital Signature

A digital signature protocol consists of 3 algorithms: Key Generation (KG), $Sign$, and $Verify$. KG algorithm takes a security parameter, outputs a key pair for user d: Signature public key/verification key (pk_d) and Signature secret key (sk_d). $Sign$ algorithm takes a message $Message$ and signature secret key (sk_d), it outputs a signature $Signature = Sig_{sk_d}(Message)$. Verify algorithm takes signature public key pk_d, signature $Signature$, and message $Message$, outputs 1 if $Verify(Message, Signature, pk_d) == 1$. This means that the signature is generated by user d under message $Message$. If $Verify(M, S, pk_d) == 0$, the signature under message $Message$ is not generated by user d.

A secure signature scheme needs to satisfy two properties: authenticity and integrity. Authenticity says that the owner of the signature convinces a verifier that the owner of the signature generates the signature using the message. Integrity provides that signed data cannot be altered by any entity.

There are several digital signature algorithms are used in blockchain: $Schnorr$ signature [23], RSA signature [24], $ECDSA$ [25].

3.6. Hash Function

A hash function (H) is a deterministic algorithm that takes a value (x) of any size and outputs a fixed size value (y). This process is shown as $H(x) = y$. Moreover, this algorithm is efficiently computed. It has the one-way property that it is infeasible to compute x from given y. Another property is that It is infeasible to find two different inputs such that x, z that satisfy $H(x) = H(z) = y$. This property is known as collision resistance.

3.7. Bloom Filter

A $Bloom$ filter is data structure that represents a set of $Data = \{a_1, \ldots, a_\beta\}$ of β elements and its represented by an array of ξ bits initially set to 0. The filter uses ℓ independent hash functions $\{H_1, \ldots, H_\ell\}$, where $H_i : \{0,1\}^* \rightarrow [1, \xi]$ for $i \in [1, \ell]$. For each item a_i in A, where $i \in \{1, \ldots, \beta\}$, the array bits at positions $H_1(a_i), \ldots, H_\ell(a_i)$ are set to 1. If any location can be set to 1 multiple times, only the first is noted. There is, however, some probability of a false positive, in which a_i appears to be in $Data$ but actually is not. False positives occur because each location may have also been set by some element other than a_i.

3.8. Pseudo Random Generator

A pseudo-random generator (PRG) outputs strings that are computationally indistinguishable from random strings. More precisely, we say that

a function $G : \{0,1\}^\kappa \to \{0,1\}^\xi$, where $\xi > \kappa$ is a (t, ϵ)-pseudo-random generator if

1. G is efficiently computable by a deterministic algorithm,
2. for all t time probabilistic algorithm $\mathcal{A}$, $|\Pr[\mathcal{A}(G(s)) = 0 | s \leftarrow \{0,1\}^\kappa] - \Pr[\mathcal{A}(r) = 0 | r \leftarrow \{0,1\}^\xi]| \leq \epsilon$

3.9. Pseudo Random Function

A pseudo-random function (PRF) is computationally indistinguishable from a random function - given pairs $(x_1, f_s(x_1)), \ldots, (x_m, f_s(x_m))$, an adversary cannot predict $f_s(x_{m+1})$ for any x_{m+1}. More precisely, we say that a function $f : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ is a (t, ϵ, q)-pseudo-random function if

1. $f(s, x) = f_s(x)$ can be computed efficiently from input $x \in \{0,1\}^\kappa$ and key $s \in \{0,1\}^\kappa$.
2. for any t time oracle algorithm $\mathcal{A}$ that makes at most q adaptive queries, $|\Pr[\mathcal{A}^{f_s(.)} = 0 | s \leftarrow \{0,1\}^\kappa] - \Pr[\mathcal{A}^g = 0 | g \leftarrow F : \{0,1\}^\kappa \to \{0,1\}^\kappa]| \leq \epsilon$

4. Threat Model

In our system, CA is a trusted entity. It registers users to the system. However, drivers, passengers, and roadside units are not trusted. Passengers can provide a fake bid or bids to manipulate auctions. Moreover, passengers (drivers) send false data or requests to gain an advantage in the system. Furthermore, they can issue fake transactions to provide replay attacks. Replay attacks prevent the system from working and manipulate the ride-sharing system.

Drivers can also send fake data and misbehave. Moreover, they can send fake transactions to do replay attacks.

RSUs can issue fake transactions to manipulate the ride-sharing system. RSUs also initiate replay attacks that sniff some transactions and change some parts of the transaction to degrade the system.

The privacy of offers needs to be taken care of. The system should protect users' offers not to be seen by any other bidders since the transactions and offers are stored in the blockchain. The offers can easily be seen by other bidders. Other bidders can use this information to gain advantage. Travel information of the users should not be leaked to public.

In the system, entities can freely read and view all transactions. Any untrusted party can be able to track drivers and passengers. The untrusted entity tracks a specific user (driver or passenger) to figure out where it is going and how much money it spends/earns for each ride.

A secure ride-sharing system should provide the following guarantees for the entities:

1. The system should protect users' bids from being viewed by other bidders to prevent other bidders from gaining an advantage.
2. The system should be resistant to replay attacks.
3. The system should provide a transparent and secure mechanism for entities that they can not refuse or deny if they have a ride-sharing service. If a driver (passenger) tries to misbehave, there should be a punishment for fraudulent users.
4. The entities' transactions should be unlinkable in the system to protect their privacy.

5. Proposed System

In this section, we present a smart contract-based ride-sharing system that is utilized with $Vickrey$ to determine passengers. The proposed system consists of 4 actors: Certificate Authority (CA), Driver,

Passenger, and Road Side Unit (RSU). CA is responsible for registering entities' public keys to the system. Once the entities are authorized, they can issue transactions. RSUs are only entities that have strong computation capabilities to form blocks. They initiate a consensus protocol to agree on the block of transactions that can be added to the blockchain. Drivers provide ride-sharing services to passengers to share their travel costs. Drivers are the only entities that create smart contracts. Passengers want to have comfortable and fast ride-sharing services. Drivers and passengers use their phone applications to provide travel information like starting point, destination point, and schedule. All users choose their public and secret keys and register their public keys to a certificate authority. This authority is trusted and can be the government (Ministry of Transport). CA also generates system parameters for the signature scheme, hash function, and pseudo-random generator. Drivers create smart contracts and put them into the blockchain with the help of RSUs that initiate consensus. Then, passengers view these contracts. If there is a match between a driver's and passenger's travel information (start and end locations), the passenger submits its bid to the contract.

Then, a $Vickrey$ auction comes into play. $Vickrey$ auction outputs the winner (passenger). The winner pays the second-highest bid. To ensure that the travel indeed happens. The driver and winner send start and end locations (destination) to the smart contract as transactions. The smart contract is executed if the transactions satisfy the conditions. Then, the smart contract finalizes the event by paying the entities.

Since the entities are known in the system in advance, we use a permissioned blockchain.

The workflow of the system as follows:

1. Each entity (passenger, driver, RSU) chooses a public/secret key pair and register their public key to Certificate Authority (CA).
2. A driver creates a smart contract (initialize smart contract parameters and create functions) and send it to a local RSU.
3. The contract is added to blockchain after a consensus protocol.
4. Passengers retrieve smart contract from the blockchain.
5. Passengers submit their bids to an RSU. Transaction is added to the blockchain as that in step-3. $Vickrey$ auction is then executed by the smart contract. Winner and the two highest bids are determined.
6. Winner is advertised.
7. The winner pays the full price as transaction. After a consensus it is added to the blockchain (step-3).
8. The driver and the passenger send their locations (start and end) to the smart contract.
9. The driver gets paid by the smart contract.

Figure 1 shows the architecture of the system and work-flow. Now, we give the details of the our ride sharing system. The system consists of two phases: $Setup$ and $Smart\ Contract$.

5.1. Setup

In this phase, each node (passenger and driver), chooses a uniformly random secret key and public key. A driver chooses sk_d as its secret key and pk_d as its public key. A passenger does the same process to have its sk_p, and pk_p. Then, the driver and the passenger registers their public keys to the certificate authority. Thus, each registered user is authorized for creating smart contracts and putting them into the blockchain, issuing transactions to the blockchain and viewing transactions from blockchain. The addresses/identities of the users are the hash of their public keys, $add =$

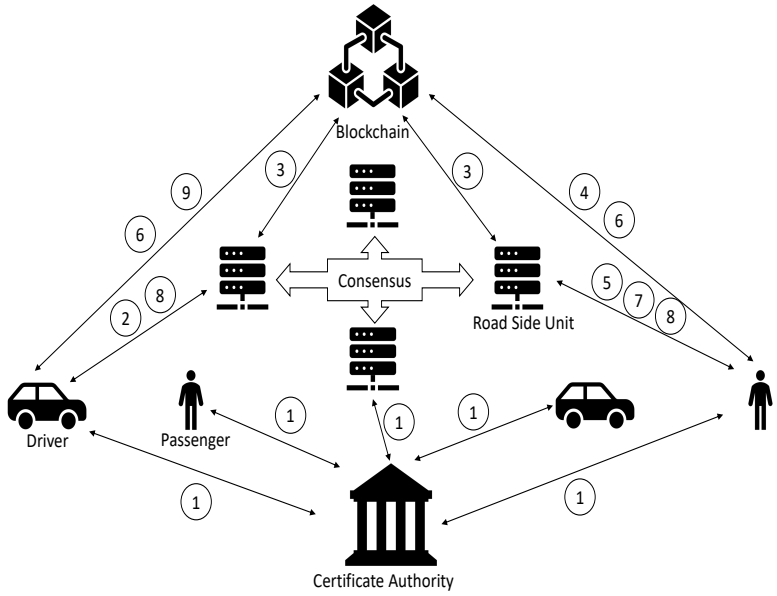

Figure 1. Architecture and workflow of the proposed system.

$HF(pk)$. The users can interact with the blockchain using their username and password combinations. CA then publish the public keys of the entities. CA also sets system's public parameters: specification of signature scheme $(KG, Sign, Verify)$, pseudo random generator F, hash functions for $Bloom$ filter $\{H_i\}_{i=1}^{l}$, $HF : G \rightarrow \{0,1\}^{\lambda}$, $HB : \{0,1\}^{*} \rightarrow \{0,1\}^{\lambda_1}$ and $HC : \{0,1\}^{\kappa} \rightarrow [1, n-1]$, where G is a group element. The symbols and their descriptions are given in Table 1.

5.2. Smart Contract

Smart-contract phase consists of $Create$, $CommitBid$, $RevealBid$, $PayPrice$, $StartTravelPassenger$, $StartTravelDriver$, $EndTravelPassenger$, $EndTravelDriver$, and $Final$ functions. With these functions users are able to interact with the contracts. Moreover, $Vickrey$ auction is implemented with the smart contract. Figure 2 shows the entity relationship diagram.

Create : This is for creating a smart contract. The driver first initializes some parameters for the contract. These are address and public key of the driver (add_d, pk_d) who creates the contract, a unique identity $Ride_{id}$, starting point $(StartPoint)$ where the travel should begin, end point $(EndPoint)$ where the travel should end, starting time $(StartTime)$ when the travel should start, highest bid $(HigBid)$, second highest bid $(SecHigBid)$, deposit $(dpst)$ is collateral to prevent dishonest users from a fraudulent behavior, km is the length of the travel, $BidCommitTime$ is a time period for users to commit their bids, $BidRevealTime$ is a time period for users to reveal their bids, $Winadd_p$ is the address of the winner. Then, the driver sets values to these parameters. Here, $dpst$ has to be paid by the driver and the passenger to eliminate any fraudulent behavior. $dpst$ is calculated by the following formula:

$$dpst = (km.GasPrice + \sum_{i=1}^{k} TollPrice_i))/2$$

, where km is the travel length, $GasPrice$ is the price of the gas (diesel or unloaded), and $TollPrice$ is the price of tolls used during the travel. $dpst$ is

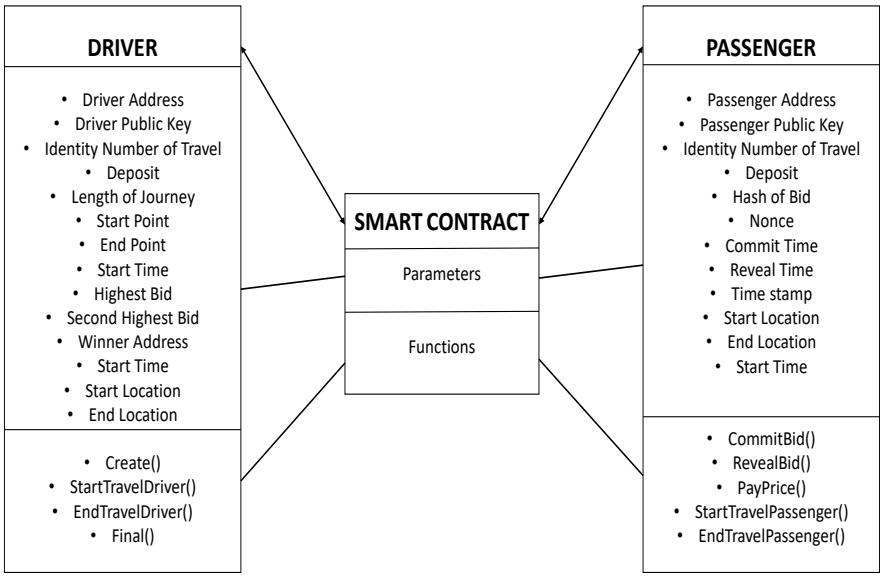

Figure 2. Entity interactions with smart contract.

calculated by the system's application automatically. Once the contract is created, it is deployed to the blockchain. $Create$ function returns address of the smart contract which is going to be used by the parties in the system.

CommitBid : This takes $(Ride_{id}, add_p, pk_p, dpst_p, h_p, ComTime, Sig_{sk_p}(A))$ as input, it outputs $bids[add_p] = (h_p)$. The input transaction is issued by he passenger. In the transaction, $h_p = HB(bid_p, r_p)$, bid_p is the bid of the passenger, r_p is the uniformly chosen random number, and HB is the collision resistant hash function, $ComTime$ is the transaction' time stamp, $A = H(Ride_{id}, add_p, dpst_p, h_p, ComTime)$. The passenger sends this transaction to the smart contract.

RevealBid : This takes $(Ride_{id}, add_p, pk_p, bid_p, r_p, RevTime, Sig_{sk_p}(B))$, where $B = H(Ride_{id}, add_p, bid_p, r_p, RevTime)$, from the passenger, $RevTime$ is the timestamp of the transaction, and outputs the identity of winner and top two highest bids. To determine the winner and the two highest bids, a $Vickrey$ auction is executed by the smart contract.

PayPrice : This function takes $(Ride_{id}, add_p, pk_p, ts, (price - dpst)_p, Sig_{sk_p}(C))$, and outputs 1 if the remaining price is paid by the winner. Otherwise, it outputs 0.

$C = H(Ride_{id}, add_p, ts, (price - dpst)_p)$, where ts is the time stamp of the transaction. If the remaining price is not paid by the winner, $2dpst$ ($dpst_d$+$dpst_p$) is sent to the driver.

StartTravelPassenger : This function takes $(Ride_{id}, add_{p'}, pk_{p'}, StartLoc_{p'}, StartTime_{p'}, Sig_{sk_{p'}}(D))$, where $D = H(Ride_{id}, add_{p'}, StartLoc_{p'}, StartTime_{p'})$ from the passenger. Here, the smart contract verifies if $add_p == Winadd_p$, verifies if the location of the passenger ($StartPoint$), time of the passenger that sends this transaction matches with $StartTime$. In this phase, before the travel is started, the driver and the passenger send their real time locations

and the time that these transactions are issued to the smart contract.

StartTravelDriver : This function takes $(Ride_{id}, add_{d'}, pk_{d'}, StartLoc_{d'}, StartTime_{d'}, Sig_{sk_{d'}}(E))$ from the driver, where $E = H(Ride_{id}, add_{d'}, StartLoc_{d'}, StartTime_{d'})$. Here, the smart contract verifies if the address of the driver equals to the address of the creator of the smart contract ($add_d == add_{d'}$), verifies if the location of the driver is $StartPoint$, checks if the driver's transaction time is the same as $StartTime$. In this phase, before the travel is started, the driver sends their real time locations and the time that these transactions are issued to the smart contract.

EndTravelPassenger : This function takes $Ride_{id}$, $add_{p'}$, $pk_{p'}$, $EndLoc_{p'}, EndTime_{p'}$, $Sig_{sk_{p'}}(H(Ride_{id}, add_{p'}, EndLoc_{p'}, EndTime_{p'}))$ from the passenger, where $EndLoc_{p'}$ is destination location of the passenger. Once the travel is over, the passenger sends his real-time location using the system's location service as a transaction to the smart contract. This function also checks if the passenger is the winner.

EndTravelDriver : This function takes $Ride_{id}$, $add_{d'}$, $pk_{d'}$, $EndLoc_{d'}, EndTime_{d'}$, $Sig_{sk_{d'}}(H(Ride_{id}, add_{d'}, EndLoc_{d'}, EndTime_{d'}))$ from the passenger, where $EndLoc_{d'}$ is destination location of the driver. Once the travel is completed, the driver sends its real-time location using the system's location service as a transaction to the smart contract. This function first checks if the driver is the creator of the smart contract.

Final : This algorithm sets second highest bid as the price that the passenger should pay. Moreover, it returns back the deposits of the other passengers that their bids are not the highest. Once the users' start and end locations are verified, and these are the same, the smart contract sends $((price - dpst)_p + dpst_p + dpst_d)$ to the driver's address. If the driver does not show up ($StartTravelDriver == \perp$) by the starting time of the travel or does not send its location to the contract and the passenger shows up in the starting point and on time, the contract sends $2dpst$ ($dpst_p + dpst_d$) to the passenger's address. Moreover, if the passenger does not show up ($StartTimeDriver == \perp$) or does not send its location by the starting time of the travel and the passenger does not send its location to the contract, but the driver sends its location which is the starting point of the travel, the contract sends $2dpst$ ($dpst_p + dpst_d$) to the driver's address and sends $(price - dpst)_p$ to the passenger's address. If two parties do not show up ($StartTimePassenger == \perp$) and ($StartTimeDriver == \perp$) or ($EndTimePassenger == \perp$) and ($EndTimeDriver == \perp$) the contract sends $dpst_d$ to add_d and send $dpst_p$ to add_p. Table 3 shows the details of smart contract functions and their properties.

$Step - 34$ in Table 3 shows that if the driver and the passenger do send their destination locations, but they do not match with the destination location ($EndPoint$) of the ravel, the contract sends $(dpst_p + dpst_d)$ to the address of the passenger.

A note that in this paper we do not examine transaction fees that entities need to pay once they issue transactions and send them to the smart contract. Since for each transaction the smart contract's state changes, the entities need to pay some transaction fees.

5.2.1 Consensus

In the proposed system, each transaction is sent to a RSU by entities (drivers, passengers). Once the RSU gets a transaction, it checks if the transaction

Table 3.
Smart contract algorithm of ride-sharing.

Initialize
$List = []$, $HigBid = 0$, $SecHigBid = 0$, $km = 0$, $StartPoint = 0$, $EndPoint = 0$,
$BidCommitTime = 0$, $BidRevealTime = 0$, $dpst = 0$, $Winadd_p = 0$, $StartTime = 0$.
Create$(Ride_{id}, add_d, pk_d, dpst_d, km_{id}, StartTime_{id}, StartPoint_{id}, EndPoint_{id}, BidTimeRange_{id})$.
1. checks if $dpst(km.GasPrice + \sum_{i=1}^{k} TollPrice_i))/2) == dpst_d$
2. checks if $add_d \in DriverSet$ and add_d is built from pk_d
3. if all checks are verified return $ContractAddress$
CommitBid$(Ride_{id}, add_p, pk_p, dpst_p, h_p, ComTime, Sig_{sk_p}(H(Ride_{id}, add_p, dpst_p, h_p, ComTime)))$.
4. checks if $((dpst_p == dpst) \wedge (ComTime \leq BidCommitTime))$
5. checks if $add_p \in PassengerSet$ and checks the signature if it is valid
6. checks if $add_p \notin List$ and checks if add_p is built from pk_p
7. If all checks are true set $List[add_p] = (h_p)$
RevealBid$(Ride_{id}, add_p, pk_p, bid_p, r_p, RevTime, Sig_{sk_p}(H(Ride_{id}, add_p, bid_p, r_p, RevTime)))$.
8. checks if $((h_p == HB(bid_p, r_p)) \wedge (RevTime \leq BidRevealTime))$
9. checks if $bid_p > dpst$ and check the signature if it is valid
10. checks if $ts \in BidTimeRange$ and checks if add_p is built from pk_p
11. if all checks are correct then Vickrey auction starts
12. $(Winadd_p, HigBid, SecHigBid) \leftarrow Vickrey(add_p, bid_p)$.
PayPrice$(Ride_{id}, add_{p'}, pk_{p'}, (price - dpst)_{p'}), ts, Sig_{sk_{p'}}(H(Ride_{id}, add_{p'}, (price - dpst)_{p'}, ts))$
13. checks if $add_{p'} == Winadd_p$ and checks if $add_{p'}$ is built from $pk_{p'}$
14. checks if $(price - dpst) == (price - dpst)_{p'})$ and checks the signature if it is valid
15. if step-14 is not verified, sends $dpst_d + dpst_{p'}$ to add_d.
16. if all checks are verified, it outputs 1.
StartTravelPassenger$(Ride_{id}, add_{p'}, pk_{p'}, StartLoc_{p'}, StartTime_{p'}, Sig_{sk_{p'}}(H(Ride_{id}, add_{p'}, StartLoc_{p'}, StartTime_{p'}))$.
17. checks if $Winadd_p == add_{p'}$, checks if $add_{p'}$ is built from $pk_{p'}$, checks the signature if it is valid
18. checks if $StartPoint == StartLoc_{p'} \wedge StartTime == StartTime_{p'}$
19. if all checks are verified, it outputs 1.
StartTravelDriver$(Ride_{id}, add_{d'}, pk_{d'}, StartLoc_{d'}, StartTime_{d'}, Sig_{sk_{d'}}(H(Ride_{id}, add_{d'}, StartLoc_{d'}, StartTime_{d'}))$.
20. checks if $add_d == add_{d'}$, checks if $add_{d'}$ is built from $pk_{d'}$, checks the signature if it is valid
21. checks if $StartPoint == StartLoc_{d'} \wedge StartTime == StartTime_{d'}$
22. if all checks are verified, it outputs 1.
EndTravelPassenger$(Ride_{id}, add_{p'}, pk_{p'}, EndLoc_{p'}, EndTime_{p'}, Sig_{sk_{p'}}(H(Ride_{id}, add_{p'}, EndLoc_{p'}, EndTime_{p'}))$.
23. checks if $add_p == add_{p'}$ and check if $add_{p'}$ is built from $pk_{p'}$.
24. checks if $EndPoint == EndtLoc_{p'}$ and checks the signature if it is valid.
25. if all checks are verified, it outputs 1.
EndTravelDriver$(Ride_{id}, add_{d'}, pk_{d'}, EndLoc_{d'}, EndTime_{d'}, Sig_{sk_{d'}}(H(Ride_{id}, add_{d'}, EndLoc_{d'}, EndTime_{d'}))$.
26. checks if $add_d == add_{d'}$ and checks if $add_{d'}$ is built from $pk_{d'}$.
27. checks if $EndPoint == EndtLoc_{d'}$ and checks if the signature is valid.
28. if all checks are verified, it outputs 1.
Final()
29. set $price$ = $SecHigBid$
30. sends $(add_j, dpst_j)$ for every j such that $(add_j \in List \wedge add_j \neq Winadd_p)$,
31. (if step-19 is satisfied $\wedge$ step-21 is not satisfied) $\vee$ (if step-19 is satisfied $\wedge$ $StartTravelDriver == \perp$),
sends $dpst_d + dpst_p$ to add_p
32. (if step-22 is satisfied $\wedge$ step-18 is not satisfied) $\vee$ (if step-22 is satisfied $\wedge$ $StartTravelPassenger == \perp$),
sends $dpst_d + dpst_p$ to add_d and send $price - dpst)_p$ to add_p
33. (if $StartTravelPassenger == \perp \wedge StartTravelDriver == \perp$),
sends $dpst_d$ to add_d and send $dpst_p$ to add_p
34. if the steps-19, 22 are satisfied but $EndPoint \neq EndLoc_p$ ($EndPoint \neq EndLoc_d$),
sends $dpst_d + dpst_p$ to add_p.
35. if the steps-19, 22, 25, 28 are satisfied,
sends $((price - dpst)_p + dpst_p + dpst_d)$ to add_d.

is valid and puts it into a pool. Depending on the consensus ($PBFT$), a master node is chosen to form a block. If enough transactions are presented in the pool, the master node (leader) builds a block and sends it to the other nodes. Other nodes check the transactions and return a message if the transactions are valid. If the master node gets enough number positive confirmations, the master node puts the block into the blockchain.

5.3. Un-Linkable Carpooling System

In this section, we provide a secure ride-sharing system.

The proposed system in Section 5 does not consider drivers' and passengers' data privacy. Since each party uses the same address for each event, any dishonest entity can easily track the party. An untrusted (malicious) party can easily track users' (passengers, drivers) locations. Moreover, an untrusted party can easily view how much money a specific driver (passenger) earns (spend). The users use different secret and public keys (addresses) to protect their privacy. Thus, they use different addresses to achieve un-linkability. The user generates a fresh random secret and public keys for each event. Moreover, it generates an address by hashing the public key. However, it is troublesome for the users to store these keys. Moreover, these public keys should be registered by a trusted authority for authorization. This results in many interactions between each user and the trusted authority (CA). If there are m public keys, there are m interactions. So, the complexity is linear in the number of keys. To tackle these problems, we use Pseudo Random Generator (PRG). To generate the Elliptic Curve Digital Signature Algorithm ($ECDSA$) secret keys and corresponding public keys, each user chooses a random seed k. Then, it uses a PRG function F and computes $F(k) = k_1||k_2||\dots||k_m$, where $k_i = f_k(i)$ is ith secret key, where f_k is a pseudo random function. Its corresponding public key is $k_i.G$, where G is the group generator on a curve. The user then shares this secret seed (k) with the Certificate Authority (CA) via a secure channel. Thus, the communication complexity will be $O(1)$ instead of $O(m)$ between CA and driver (passenger). CA then generates these keys using F and registers the corresponding public keys to the system. CA then needs to publish these authorized keys in the system. If CA uses this method, each RSU needs to store mn keys, where n is the number of users in the system. Each RSU searches a specific public key (address) to check if it is authorized in $O(mn)$ time. To have an efficient search, We use the technique in [26] to have fast authorization. It only takes $O(1)$ time instead of $O(mn)$ to verify whether a user is authorized to issue a transaction, create a smart contract, and bid for ride-sharing.

The unlinkable construction requires to modify $Setup$ and $Consensus$ phases in Section 5. Moreover, in the contract, each user (passenger and driver) uses a different public key and address for each ride.

NewSetup : In this phase, each node (passenger and driver) chooses a uniformly random secret key and public key for a signature algorithm such as $ECDSA$ [25]. A driver picks sk_d as its secret key and pk_d as its public key (verification key). The hash of its public key is the address of this user. A passenger does the same process to have its sk_p and pk_p. Moreover, each user chooses a secret random seed k_d for a driver (k_p is for the passenger) and sends it to CA using a secure channel.

Then, CA chooses l independent hash functions: $H_1, \dots, H_l$, where $H_i : G \rightarrow [1, \xi]$ and does the followings:

1. for each user (passenger and driver), it computes $F(k_d)$ for driver ($F(k_p)$ for passenger)

to derive all secret keys. Then, it computes all public keys $k_{d,1}.G$, ..., $k_{d,m}.G$.
2. it uses each public key $k_{d,i}.G$, where $i \in \{1, \ldots, m\}$ to compute $H_1(k_{d,i}.G)$, ..., $H_l(k_{d,i}.G)$.
3. it sets a 1 in each location of the hash result in the array.
4. it repeats steps-1,2,3 for each driver and passenger.
5. it then sends the $Bloom$ filter to the each RSU.

NewConsensus : Each RSU does a couple of checks in the new consensus. Each user uses a different public key and address for each event. In other words, each RSU rejects the new transaction if the new transaction consists of the previously used public key to eliminate replay attacks. A replay attack happens when an unauthorized user uses a previous transaction issued by an authorized user as their transaction to fool the RSU. If the new public key has not been used previously, it evaluates it on ℓ hash functions to check its authenticity by verifying if all positions are 1s in the $Bloom$ filter. Then, it controls whether the new address is formed from the public key by computing its hash. Then, it checks the validity of the signature. This check is to prevent replay attacks.

All checks and the consensus process are as follows:

- RSU checks if the public key used in the current transaction was previously used in the blockchain.
  - if it was used, RSU discards the transaction.
- if the public key was not used before it evaluates the public key in ℓ hash functions. Then, it verifies if all positions are 1s in the $Bloom$ filter.
  - it not, the RSU discards the transaction.
- if all positions consist of 1s in the $Bloom$ filter, RSU checks the address if it is obtained from the public key.
  - if the address is not formed by the public key, the RSU discards the transaction.
- if the address is formed from the public key, it checks if the signature is valid.
  - if the signature is not valid, the RSU discards the transaction.
- if the signature is valid, the RSU then puts the transaction to the pool.

The remaining process is followed as that in Section 5.2.1.

6. Complexity Analysis

In this section we analyze the the proposed protocol's complexity. Our analysis is based on m number of events (smart contracts) that a passenger/driver would like to participate.

For a passenger, each passenger executes each smart contract two times: $CommitBid, RevealBid$. However, if the passenger is the winner, then it executes five times: $CommitBid$, $RevealBid$, $PayPrice$, $StartTravelPassenger$, $EndTravelPassenger$.

For a driver, the driver executes each smart contract four times: $Create$, $StartTravelDriver$, $EndTravelDriver$, and $Final$. The $Final$ function does not take any inputs.

To analyze the complexity of our system, we assume that a user (passenger, driver) gets involved in m events (ride-sharing). Table 4 and 5 show the required storage, communication, and computational complexities for each user. Driver and passenger passes some parameters based on the function to the smart contract to execute. We assume that $ECDSA$ secret key (sk) is represented as 256 bits, public key (pk) 512 bits, signature 512 bits, identity of a

ride ($Ride_{id}$) 128 bits, deposit ($dpst$) 128 bits, each timestamp ($ComTime$, $RevTime$, ts, $StartTime$, $EndTime$) 128 bits, address (add) 160 bits (output of HF), each location ($StartLoc$, $EndLoc$) 128 bits, bid 128 bits, random value r 128 bits, output of HB hash function is 128 bits.

If a passenger wins all m events, it requires 8096 $\cdot\ O(m)$ bits to interact with the smart contracts. For a driver, it requires 3136 $\cdot\ O(m)$ bits to interact with the smart contracts.

In Table 5, CB stands for $CommitBid$, RB stands for $RevealBid$, PP stands for $PayPrice$, STP stands for $StartTravelPassenger$, STD stands for $StartTravelDriver$, ETP stands for $EndTravelPassenger$, ETD stands for $EndTravelDriver$, F stands for $Final$ functions, and $P \leftrightarrow D$ stands for the communication complexity between passenger and driver, and $CA \leftrightarrow D(P)$ stands for the communication complexity between CA and driver (passenger). In Table 4, subscript $F : \{0,1\}^{\kappa} \rightarrow \{0,1\}^{2\kappa}$, $HF : G \rightarrow \{0,1\}^{160}$, exp is denoted as the exponentiation operation in G, $H : \{0,1\}^{*} \rightarrow \{0,1\}^{256}$, $Signature$ is the signature, SMC is denoted as smart contract creation, $HB : \{0,1\}^{256} \rightarrow \{0,1\}^{128}$.

7. Security Analysis

In this section, we examine the proposed scheme's security analysis. The proposed system satisfies all requirements in Section 4.

1. This step is satisfied that bidders commit their bids and then reveal them. In the commit phase, nobody sees others' bids. Thus, nobody gains an advantage over others. However, to determine the winner, everyone reveals their bids.
2. The replay attacks are prevented in the system since each RSU verifies the public keys and addresses of the entities, time stamps of the transactions, and signatures. However, some of the RSUs can be fraudulent that they can issue fake transactions. Thus, the fake transaction can be accepted by them. In the definition of consensus, these RSUs are the minority in the system. The majority can not accept any fake transaction. Thus, this condition is also satisfied. In $PBFT$ consensus, at most $1/3$ of the total RSUs can be malicious. Others should be honest.
3. A passenger can get the ride-sharing service. However, it can deny that it did not get the ride-sharing service to get the driver's collateral. A driver might also refuse that the passenger is fraudulent and did not show up at the start time and starting point to gain the passenger's collateral. To figure out if anyone is lying, the proposed system (smart contract) checks the transactions issued by both parties to execute functions: $StartTravelDriver$, $StartTravelPassenger$, $EndTravelDriver$, and $EndTravelPassenger$. The smart contract verifies the locations and time stamps of the transactions of the entities. Thus, this condition is satisfied.
4. Unlinkability of the entities' transactions are obtained by using different public key and addresses of the entities for different ride-sharing events. Any dishonest entity in the system can not track any specific driver (passenger).
   Unlinkability property is satisfied by using well-known cryptographic primitive which is pseudo random generator (PRG). If F is a PRG with secret seed k then $F(k) = f_k(1)||\ldots||f_k(n)$, where f is a pseudo random function. If the users' secret keys are generated by using a PRF f than the adversary can not distinguish the outputs of the PRF function from a random

Table 4.
Storage and computational complexity analysis of the proposed system.

	CC	CB	RB	PP	STP	STD	ETP	ETD	F	P $\leftrightarrow$ D	CA $\leftrightarrow$ D (P)
Passenger	-	$xO(m)$	$xO(m)$	$yO(m)$	$yO(m)$	-	$yO(m)$	-	-	-	O(1)
Driver	$\|CC\|O(m)$	-	-	-	-	$yO(m)$	-	$yO(m)$	$128O(m)$	-	O(1)

Table 5.
Communication complexity analysis of the proposed system ($x = 1696, y = 1568$).

	Storage	Computation
Passenger	$O(1)$	$O(m\kappa)_F + O(m)_{HF} + O(m)_{HB} + O(m)_{Exp} + O(m)_H + O(m)_{Sign} + O(m)_{Subs}$
Driver	$O(1)$	$O(m)_{SMC} + O(m\kappa)_F + O(m)_{HF} + O(m)_{Exp} + O(m)_{Signature}$

function with non-negligible probability.

Theorem 1 *If $F(k) = k_1||k_2||\dots||k_n$ is a pseudo random generator, where $k_i = f_k(i)$ and f_k is a pseudo random function $f : \{0,1\}^\kappa \times \{0,1\}^\kappa \rightarrow \{0,1\}^\kappa$ is a pseudo random function, then the adversary $\mathcal{A}$ can not link the passengers' multiple events with non-negligible probability.*

Constructing a PRF from a PRG as above by using well-known cryptographic primitive $GGM - PRF$ [27]. In [27], a length doubling pseudo random generator $F : \{0,1\}^\kappa \rightarrow \{0,1\}^{2\kappa}$ is sequentially applied on a random string in the root node of the tree to build a binary tree. The leaf nodes in the tree are represented as the evaluations of the PRF function. The root node value is a uniformly random string chosen from $\{0,1\}^\kappa$. The process is illustrated in Figure 3.

Before we prove the theorem, we define a game between an attacker $\mathcal{A}$ (event attacker), a challenger $\mathcal{B}$ (PRF attacker) and a PRG attacker $\mathcal{C}$. We define each smart contract as an event. Thus, a passenger or a driver can involve multiple events so it uses multiple smart contracts. For an event/smart contract e_i, the entity (a passenger or a driver) uses the same identity/address when it executes the smart contract's functions. For event e_{i+1}, each entity uses different identity/address for the transactions used in event e_i. Each event consists of transactions that include following values:

- ride identity $Ride_{id}$,
- address of passenger add_p,
- address of driver add_d,
- public key of passenger pk_p,
- public key of driver pk_d,
- hash of a bid of the passenger h_p,
- deposit value $dpst$,
- bid commit Time $ComTime$,
- bid reveal Time $RevTime$,
- bid of passenger bid_p,
- random value of passenger r_p,
- time stamp ts,
- starting location of the travel $StartLoc$,
- destination location of the travel $EndLoc$,

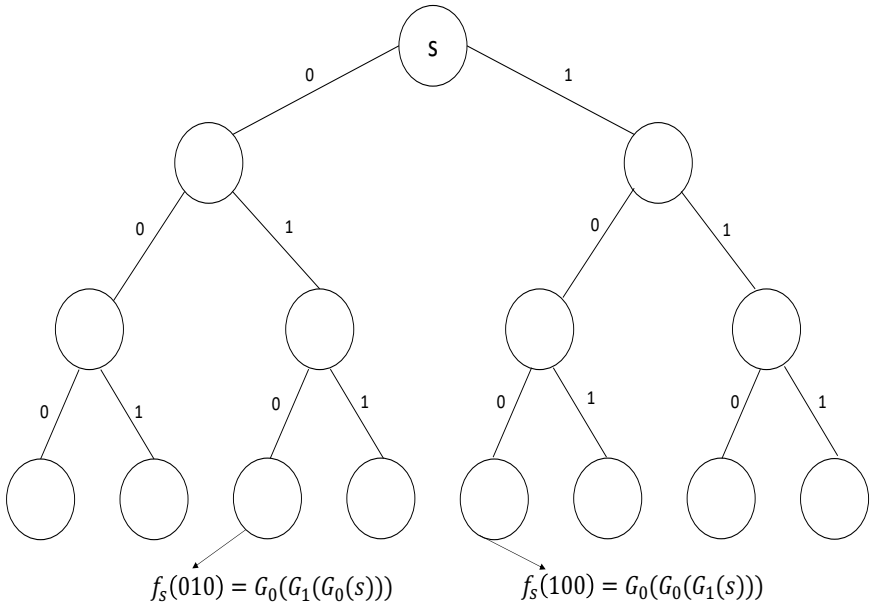

Figure 3. 3-Level GGM tree.

- starting time of the travel $StartTime$,
- signature of passenger,
- signature of driver.

We prove this theorem using its contrapositive. Assuming that an attacker $\mathcal{A}$ attacks our unlinkable protocol with a non-negligible probability, $\mathcal{B}$ uses $\mathcal{A}$ as a subroutine to distinguish a pseudo random function from a random function with non-negligible probability. Then, a PRG adversary $\mathcal{C}$ uses $\mathcal{B}$ to distinguish pseudo random strings from random strings. The second part of the proof follows from GGM based PRF proof in 3. $\mathcal{C}$ is given a set of strings that their length 2κ chooses an index i then answers polynomial number of queries of $\mathcal{B}$ based on the index. The proof follows using hybrid games: there are κ level games. We assume that $\mathcal{A}$ breaks the unlinkability property in our construction, $\mathcal{B}$ distinguish pseudo random function f_s from a random function with non-negligible property. Then, $\mathcal{C}$ uses $\mathcal{B}$ to distinguish two hybrid games H_0 and $H_{\kappa-1}$ with non-negligible probability, where H_0 is the real protocol using PRG and PRF values (on leaf nodes) for the identities/addresses of the entities while $H_{\kappa-1}$ is the random strings and random function values are used for the identities/addresses of the entities.

Setup: $\mathcal{B}$ sets system public parameters and sends them to $\mathcal{A}$. $\mathcal{A}$ creates a smart contract and puts it into blockchain. Moreover, $\mathcal{A}$ outputs event e_{i_0,j_0} (entity (passenger) i_0's j_0th event) and e_{i_1,j_1} (entity i_1's j_1th event) to be challenged upon.

Query: $\mathcal{A}$ adaptively makes q-event queries. For event e_{i_α,j_β} ($(0 \neq \beta)$ and $(1 \neq \beta)$), where $1 \leq \beta \leq \frac{q}{2}$ and $\alpha \in \{0,1\}$, $\mathcal{C}$ gives the corresponding leaf value on leaf j_β of passenger i_α to $\mathcal{B}$, then $\mathcal{B}$ forms transactions based on the received value and sends them to $\mathcal{A}$.

Challenge: $\mathcal{B}$ flips a coin b forms the transactions based on e_{i_0,j_0} if $b = 0$, otherwise it forms the transactions based on e_{i_1,j_1} received value from $\mathcal{C}$ and sends them to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs bit b'.

If $b = b'$, the adversary wins the game. The

advantage of the adversary is $Adv_{\mathcal{A}}^{UnLink} = \frac{1}{2} + \epsilon$, where ϵ is a negligible value.

A note that we assume that the adversary has some knowledge about the passengers and drivers. That means the adversary knows the entities' travel information. Thus, we introduce some rules to restrict the adversary in our ride-sharing system. Let passenger i_0's travel locations' (departure and destination) set be $TL_{i_0} = \{(StartPoint_{e_{i_0,j_\beta}}, EndPoint_{e_{i_0,j_\beta}})\}_{\beta=1}^{n}$, passenger i_1's travel locations' set be $TL_{i_1} = \{(StartPoint_{e_{i_1,j_\beta}}, EndPoint_{e_{i_1,j_\beta}})\}_{\beta=1}^{n}$, and challenge event's (smart contract) travel location $TL_c = StartPoint_c, EndPoint_c$. Then, $(TL_{i_0} \cup TL_{i_0}) \cap TL_c = \emptyset$, or $TL_c \in (TL_{i_0} \cap TL_{i_0})$. In other words, we do not let the adversary to issue the challenge event's location as TL_c if $TL_{i_0} \cap TL_c = \emptyset$ but TL_c if $TL_{i_1} \cap TL_c \neq \emptyset$ or vice versa. Otherwise, the adversary can win the game with high probability since a passenger can use the same departure and destination locations multiple times.

Proof: **Setup**: $\mathcal{B}$ sets system public parameters and sends them to $\mathcal{A}$. These parameters consists of the specification of PRG F, a hash function $H : \{0,1\}^{\kappa} \rightarrow [1, n-1]$, signature parameters ($ECDSA$), l hash functions for setting $Bloom$ filter, a hash function HB for bid commitment, a hash function HF, $\mathcal{A}$ creates a smart contract and puts it into the blockchain. Moreover, $\mathcal{A}$ outputs event e_{i_0,j_0} (entity i_0's j_0th event) and e_{i_1,j_1} (entity i_1's j_1th event) to be challenged upon.

Query: $\mathcal{A}$ adaptively makes q-event queries. For event e_{i_α,j_β} ($(0 \neq \beta)$ and $(1 \neq \beta)$), where $1 \leq \beta \leq q/2$, $\mathcal{B}$ asks $\mathcal{C}$ to get user i_α's j_βth leaf node value on the corresponding tree. Once $\mathcal{B}$ receives the value $a_{\alpha,\beta}$ which is the value ($f_{k_\alpha}(\beta) \in \{0,1\}^{\kappa}$) of a pseudo random function evaluated on β or a random function's output $\{0,1\}^{\kappa}$. Then, $\mathcal{B}$ forms the transaction based on the function: $CommitBid$, $RevealBid$, $PayPrice$, $StartTravelPassenger$, and $EndTravelPassenger$. Then, $\mathcal{B}$ sends them to $\mathcal{A}$. To form user α's βth address and public key (for signature), it converts $\{0,1\}^{\kappa}$ to range $[1, n-1]$, where n is the order of the group. For this conversion, $\mathcal{B}$ uses H. Then it computes $pk_{\alpha,\beta} = H(a_{\alpha,\beta})G$ (verification key). The address of the passenger is the hash of the public key, $HF(pk_{\alpha,\beta})$.

Challenge: $\mathcal{B}$ flips a coin b and asks $\mathcal{C}$ to receive the value of a_{i_b,j_b}. Once it receives the value from $\mathcal{C}$ forms the transactions as that in query phase. Then, $\mathcal{B}$ sends them to $\mathcal{A}$.

Guess: $\mathcal{A}$ outputs bit b'.

If the received value is the output of PRF, $Adv_{\mathcal{A},\kappa}^{UnLink} = \Pr[\mathcal{B}^{PRF(\cdot)}(1^{\kappa}) = 0]$, where $PRF \leftarrow \{0,1\}^{\kappa}$. If the received value is the output of uniformly random function (RF), $RF \leftarrow \mathcal{F}$, $Adv_{\mathcal{A},\kappa}^{UnLink'} = \Pr[\mathcal{B}^{RF(\cdot)}(1^{\kappa}) = 0]$, where $RF : \{0,1\}^{\kappa} \rightarrow \{0,1\}^{\kappa}$.

Thus, we can write $|Adv_{\mathcal{A},\kappa}^{UnLink} - Adv_{\mathcal{A},\kappa}^{UnLink'}| = |\Pr[\mathcal{B}^{f_{PRF}(\cdot)}(1^{\kappa}) = 0] - \Pr[\mathcal{B}^{RF(\cdot)}(1^{\kappa}) = 0]| \geq \epsilon$. It means that if $\mathcal{A}$ breaks our scheme with non-negligible probability ϵ, $\mathcal{B}$ distinguishes PRF functions from random functions. we also know the result from [27] that the relation between winning probabilities between $Adv_{\mathcal{C},\kappa}$ and $Adv_{\mathcal{B},\kappa}$ is $Adv_{\mathcal{C},\kappa} = \frac{\epsilon}{\kappa q} Adv_{\mathcal{B},\kappa}$, that PRG-attacker $\mathcal{C}$ distinguishes random strings from pseudo random strings generated by PRG F is also non-negligible. Thus, it results in a contradiction that violates the security of F.

The probability of $Adv_{\mathcal{A},\kappa}^{UnLink'}$ needs to be calculated when $\mathcal{B}$ uses random strings to answer queries from $\mathcal{A}$. There are q queries. If $\mathcal{B}$ uses the same strings to answer $\mathcal{A}$'s query, it is possible that $\mathcal{A}$ figures out whether e_{i_0,j_0} or e_{i_1,j_1}. The probability of picking the same string is $\frac{q}{2^{\kappa}}$. Thus, $Adv_{\mathcal{A},\kappa}^{UnLink'} = \frac{1}{2} + \frac{q}{2^{\kappa}}$. When we combine this result

with $|\Pr[Adv^{UnLink}_{\mathcal{A},\kappa} = 0] - \Pr[Adv'_{\mathcal{A},\kappa} = 0]| \leq \epsilon$, we have $Adv_{\mathcal{A},\kappa} = \frac{1}{2} + \frac{q}{2^\kappa} + \epsilon = \frac{1}{2} + \epsilon'$, where ϵ' is a negligible function.

□

Data privacy of a passenger is satisfied by using users' pseudo identities not real identities. Since there is no real identity in the system, our protocol satisfies data privacy. Data privacy and unlinkability property of a driver can be shown in a similar way as above. In the game, the adversary gives two different identities and their events to the challenger to be challenged upon to create one of the smart contracts. Moreover, it asks q-adaptive smart contract queries. In the challenge phase, the challenger gives one of two challenge smart contract and the corresponding driver's smart contract function executions (StartTravelDriver, EndTravelDriver, Final). Then, the adversary makes its guess.

8. Discussion

A user reputation mechanism can be added to our first scheme (users' addresses do not change for each event) but not the second scheme.

In general, a user increases his reputation once he completes a ride-sharing event successfully. The user gets credits for each event. The user's reputation/credits are linked to his address. The user should have a constant identity (address) that it credits to accumulate. The credits are stored in the blockchain so that anyone can view them. In our first scheme, each user can have a reputation score since each user has a single identity (address) for every ride-sharing event. However, it is difficult to add a reputation mechanism to second scheme. In the second scheme (more secure), the entities do not use a fixed address to get credits. Besides, the entities use a different address for each ride-sharing event. A possible solution is to have a trusted third party (CA) that can map entities' credits to entities' addresses. Then, the trusted party maps entities' credits to their current addresses and makes them public. However, mapping a user's credits to his new address results in a privacy breach. If a dishonest entity knows a user's total credits, the entity can easily de-anonymize the user. The entity simply aggregates a bunch of credits that belong to different addresses in the blockchain. Then, it checks if the aggregated value equals accumulated credit. So, an entity tries to map multiple addresses (credits) to a specific user by having their reputation score.

A note that if there is only one passenger that it offers bid b for the event, the passenger pays $dpst$ since the passenger should offer $b > dpst$. Thus, $dpst$ will be the second price to pay to the driver.

One of the main reason why we use $Vickrey$ auction is to provide benefits to both parties (passenger and driver). Drivers are paid at least $dpst$ which is half of the travel expenses (toll costs, gas cost). Passengers pay the second highest price as the travel fee. $Vickrey$ auction provides entities a transparent and secure auction mechanism. $Vickrey$ auction provides bid privacy to the entities. The auction consists of two phases: Commit and Reveal. In the commit phase, the passengers offer bids while their bids are not seen by other bidders. In the reveal phase, the passengers should reveal their bids in the clear. The winner is decided once the revealed bid in the reveal phase matches with the committed bid in the commit phase.

In the literature, many studies considered multiple passengers to travel together. In this case, each user occupies an available seat until there is no seat. In the proposed protocol, we focus on one passenger to travel due to some pandemics that protect entities from infections. To decide the passenger among multiple passengers, we use $Vickrey$ auction. Our focus is to have data privacy and functionality

(machine learning, data mining, and protection of health) at the same time.

In the proposed system, the liability of driver is examined by the trusted party (certificate authority). In this paper, we do not focus on potential considerations like in the event of an accident or dispute. We leave it as future work.

We say our system has lightweight operations since the operations heavily consist of hashing for anonymization technique to preserve users' data privacy. Other studies heavily use encryption algorithms [1], [2], [8], [11], [12], [13] to preserve users' data privacy and others do not use any method to preserve users' data privacy [4], [6], [14], [16], [17], [18].

Scalability is also a main problem in blockchain and related to the complexity of the consensus mechanism. Once transactions and blocks are issued by users, they are broadcast to entire network. This results in high throughput in the network. The throughput of the blockchain is defined as the number of transactions stored in a blockchain per second. In our protocol, we might face a scalability problem once a dishonest passenger executes the smart contract functions sequentially many times to form transactions. This results in a scalability problem in the network. If the dishonest passenger uses the same address to form a transaction, the smart contract can be programmed in a way to catch this fraudulence. However, if the passenger uses different pseudonyms (addresses), then all its transactions will be valid since the passenger is anonymous in the system. This fraudulence can not be caught by the smart contract. In this case, the trusted party (CA) should come to play to check all transactions to figure out the dishonest passenger. Once CA figures it out, the passenger should be omitted from the system. Then, CA should update the $Bloom$ filter without using the dishonest passenger's public keys and sends it to RSUs.

9. Conclusion and Future Work

This paper proposes a solution for a secure, safe, and transparent ride-sharing system in a pandemic such as $COVID-19$ using blockchain and smart contracts. To be protected from viral infections in pandemics, people keep physical distance and avoid crowds and close contact. In this paper, instead of fully booking their car seats with passengers, the driver takes one person to share the ride to avoid crowds and getting any infection. A smart contract is proposed in the system to execute a $Vickrey$ auction to determine the passenger. The $Vickrey$ auction allows users to use sealed bids. Thus, bids are protected from being viewed by others. The smart contract keeps all transactions in a secure and transparent ride-sharing system. Moreover, any third party can not link the passenger and drivers' travel data (locations and cost of travel). The user chooses a different pseudo anonymous address and public key for each riding event to unlink their transactions. Since the travel data is not encrypted, the data can be applied to machine learning and data mining to get real-world applications.

As a future work, we would like to implement our work to evaluate its performance. We will examine users' communication and computational costs once they execute the smart contracts. Another future work will be our system's liability once in the event of an accident or dispute.

Declarations

Funding Statement : No funds, grants, or other support was received.

Conflict of Interest Disclosure : The authors declare that they have no conflict of interest.

References

[1] D. Wang and X. Zhang, "Secure ride-sharing services based on a consortium blockchain," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2976–2991, 2021.

[2] D. Zonda and M. Meddeb, "Proxy re-encryption for privacy enhancement in blockchain: Carpooling use case," in *IEEE International Conference on Blockchain (Blockchain)*, 2020, pp. 482–489.

[3] K. Kato, Y. Yan, and H. Toyoizumi, "Blockchain application for rideshare service," in *8th International Conference on Logistics, Informatics and Service Sciences (LISS)*, 2018, pp. 1–5.

[4] S. Kudva, R. Norderhauga, S. Badsha, S. Sengupta, and A. S. M. Kayes, "Pebers: Practical ethereum blockchain based efficient ride hailing service," in *EEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*. IEEE, 2020, pp. 422–428.

[5] M. S. Hossan, M. L. Khatun, S. Rahman, S. Reno, and M. Ahmed, "Securing ride-sharing service using ipfs and hyperledger based on private blockchain," in *24th International Conference on Computer and Information Technology (ICCIT)*, 2021, pp. 1–6.

[6] N. Mahmoud, A. Aly, and H. Abdelkader, "Enhancing blockchain-based ride-sharing services using ipfs," *Intelligent Systems with Applications*, vol. 16, p. 200135, 2022.

[7] T. M. Choi and X. Shi, "On-demand ride-hailing service platforms with hired drivers during coronavirus (covid-19) outbreak: can blockchain help?" *IEEE Transactions on Engineering Management*, vol. 71, pp. 737–752, 2024.

[8] J. Huang, Y. Luo, M. Xu, B. Hu, and J. Long, "pshare: Privacy-preserving ride-sharing system with minimum-detouring route," *Applied Sciences*, vol. 12, no. 2, pp. 1–18, 2022.

[9] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer: Berlin/Heidelberg, Germany, 1999, pp. 223–238.

[10] T. M. Choi and J. B. Sheu, "Risk-averse ride-hailing platform operations with safety risk-averse consumers under pandemics: Roles of blockchain technology and government sponsors," *IEEE Transactions on Engineering Management*, 2023.

[11] X. Shen, Z. Wang, B. Wang, L. Wang, and Q. Pei, "A privacy-preserving ride-matching scheme without a trusted third-party server," *IEEE Systems Journal*, vol. 17, no. 4, pp. 6413–6424, 2023.

[12] M. Li, Y. Chen, C. Lal, M. Conti, F. Martinelli, and M. Alazab, "Nereus: Anonymous and secure ride-hailing service based on private smart contracts," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 2849–2866, 2023.

[13] H. W. Q. Li and C. Dong, "A privacy-preserving ride matching scheme for ride sharing services in a hot spot area," *Electronics*, vol. 12, no. 4:915, 2023.

[14] X. Zhang, J. Liu, Y. Li, Q. Cui, X. Tao, and R. P. Liu, "Blockchain based secure package delivery via ridesharing," in *11th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2019, pp. 1–6.

[15] X. Zhang, J. Liu, Y. Li, Q. Cui, X. Tao, R. P. Liu, and W. Li, "Vehicle-oriented ridesharing package delivery in blockchain system," *Digital Communications and Networks*, 2022.

[16] S. Benedict, "Shared mobility intelligence using permissioned blockchains for smart cities," *New Gener. Comput.*, vol. 40, no. 4, pp. 1009–1027, January 2022.

[17] S. Chopra, B. Palanisamy, and S. Sural, "Credit-based peer-to-peer ride sharing using smart contracts," in *2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, pp. 1–3.

[18] S. Renu and B. G. Banik, "Implementation of a secure ride-sharing dapp using smart contracts on ethereum blockchain," *International Journal of Safety and Security Engineering*, vol. 11, pp. 167–173, 2021.

[19] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2008.

[20] V. Buterin, "Ethereum," Tech. Rep., 2014.

[21] A. Hahn, R. Singh, C. C. Liu, and S. Chen, "Smart contract-based campus demonstration of decentralized transactive energy auctions," in *IEEE Power and Energy Society Innovative Smart Grid Technologies Conference (ISGT*. IEEE, 2017, pp. 1–5.

[22] T. Chen, A. S. Khan, G. Zheng, and S. Lambotharan, "Blockchain secured auction-based user offloading in heterogeneous wireless networks," *IEEE Wireless Communications Letters*, vol. 9, no. 8, pp. 1141–1145, Aug 2020.

[23] C. Schnorr, "Efficient signature generation by smart cards," *J. Cryptology*, vol. 4, pp. 161–174, January 1991.

[24] K. M. Moriarty, B. S. Kaliski, J. Jonsson, and A. Rusch, "Pkcs #1: Rsa cryptography specifications version 2.2," RFC, Tech. Rep., 2016.

[25] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, pp. 36–63, 2001.

[26] Z. Guan, G. Si, X. Zhang, L. Wu, N. Guizania, X. Du, and Y. Ma, "Privacy-preserving and efficient aggregation based on blockchain for power grid communications in smart communities," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 82–88, July 2018.

[27] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *Journal of the ACM (JACM)*, vol. 33, no. 4, pp. 792–807, 1986.