IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 2, 2020, accepted August 11, 2020, date of publication August 17, 2020, date of current version August 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3016945

Smart Parking System With Privacy Preservation and Reputation Management Using Blockchain

MAHMOUD M. BADR[1], WESAM AL AMIRI[2], MOSTAFA M. FOUDA[3,7], (Senior Member, IEEE), MOHAMED M. E. A. MAHMOUD[1], (Senior Member, IEEE), ABDULAH JEZA ALJOHANI[4,6], (Member, IEEE), AND WALEED ALASMARY[5], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, Tennessee Technological University, Cookeville, TN 38505, USA
[2]Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO 65409, USA
[3]Department of Electrical and Computer Engineering, College of Science and Engineering, Idaho State University, Pocatello, ID 83209, USA
[4]Department of Electrical and Computer Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[5]Department of Computer Engineering, Umm Al-Qura University, Makkah 21421, Saudi Arabia
[6]Center of Excellence in Intelligent Engineering Systems (CEIES), King Abdulaziz University, Jeddah 21589, Saudi Arabia
[7]Department of Electrical Engineering, Faculty of Engineering at Shoubra, Benha University, Cairo 13518, Egypt

Corresponding author: Mahmoud M. Badr (mmbadr42@students.tntech.edu)

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant DF-265-135-1441.

ABSTRACT Most of the existing smart parking systems threaten the drivers' privacy by revealing information about their visited locations. Moreover, they are centralized making them vulnerable to a single point of failure and attack, which threatens the availability of the parking service. They also suffer from a lack of transparency, where the centralized service organizer may favor some parking lots by booking their parking slots first. To tackle these concerns, we propose a blockchain-based smart parking system with privacy preservation and reputation management. In our system, a consortium blockchain is created by different parking lots to run the parking system securely and transparently, where the parking offers are recorded on a shared and immutable ledger. We use a commitment technique during the submission of the offers to ensure fair parking rates. Then, we use a private information retrieval technique during the offers retrieval to preserve the drivers' location privacy. Furthermore, to anonymously and efficiently authenticate drivers during the reservation process, we use a short randomizable signature. We also use a time-locked anonymous payment technique to discourage drivers from not committing to their parking reservations and provide a secure and privacy-preserving payment method for parking service. Finally, we integrate a blockchain-based anonymous reputation management scheme into our system, where drivers can anonymously rate the parking service to ensure high quality of service. Our evaluations demonstrate that our smart parking system is secure and capable of preserving drivers' privacy with low communication, computation, and storage overheads.

INDEX TERMS Blockchain, intelligent transportation systems, privacy, private information retrieval, reputation management, security and smart parking.

I. INTRODUCTION

With the fast-growing number of vehicles, finding a vacant parking slot has become a daily problem in our societies, especially in crowded cities. According to the report of South China Morning Post [1], more than 1.3 million drivers struggle every day to find available parking slots in Shanghai city, China. This problem has several negative consequences on individuals, society, and environment [1], [2]. On the individuals, finding an available parking slot takes time. For example, a driver may spend around 8.1 minutes and 20 minutes searching for an available parking slot in Los Angeles (USA) and Cairo (Egypt), respectively [3], [4]. In addition to wasting time, searching for an available parking slot increases fuel consumption. Around 47, 000 gallons of gasoline are consumed in Los Angeles only per year in finding vacant parking

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

slots [5]. For the impact on society, searching for available parking slots may cause traffic congestions. According to [6], it causes 30% of the traffic congestions in crowded cities. For the impact on the environment, according to the report of United States Environmental Protection Agency [7], searching for available parking slots has become a big contributor to the air pollution problem in the United States. In Los Angeles alone, the amount of carbon dioxide produced due to this problem is 728 tons per year [5].

Smart parking systems have been emerging as an effective solution for the fast-growing problem of finding vacant parking slots. In these systems, the parking slots are equipped with Internet of Things (IoT) devices to check the availability of these slots. These devices can communicate directly with a parking server. Hence, it can provide real-time parking availability information using an online platform to drivers, and allow them to make online reservations via smart phones.

Deploying smart parking systems has gained a great interest in the past few years, and many companies have started investing in these systems in different cities around the world [8]–[12]. However, the existing systems have several security and privacy problems.

In terms of security, most of the existing smart parking systems [8]–[12] are centralized, i.e., they rely on a single entity (a server) to publicize the available parking slots and make reservations. This centralized architecture makes the parking system vulnerable to a single point of failure problem, distributed denial of service (DDoS) attacks, and remote hijacking attacks, which threaten the parking service availability. Moreover, in these systems, drivers and parking lots owners should trust the service manager. However, the manager may be dishonest and favor some parking lots by booking their parking slots first, which may cause financial losses to some parking lots owners and deprive the drivers from finding parking slots close to their destinations. This is because these systems suffer from a lack of transparency.

In terms of privacy, the existing smart parking systems [8]–[12] do not consider the privacy of drivers. They require drivers to disclose sensitive information during queries for available parking slots and reservations, such as real identities, destinations, and reservation times to the service manager. By analyzing these information, the manager can infer drivers' visited locations, daily activities and life patterns such as work address, health condition, income level, etc. [13]. These information can be sold to insurance and marketing companies. Moreover, if a user is travelling, criminals can break into his house to steal it. Therefore, it is very essential to hide the drivers' desired destinations from the service manager to preserve their privacy while enabling the manager to return the parking offers in the area of interest. Generally, with the increasing popularity of the location-based services, *location privacy has been recognized as a big concern in the literature in different applications* such as location-based queries, reporting traffic data, and ridesharing organization [14]–[17]. In this regard, some schemes [18]–[21] have been proposed in the literature to address the privacy issues in the existing smart parking systems. However, they are centralized, and thus they do not resolve the related security issues in terms of single point of failure and attack and lack of transparency. Also, they lack enough protection to location privacy due to using location obfuscation technique which leaks information on drivers' locations.

Motivated by the research gaps in the literature in terms of security and privacy as explained above, in this article, we propose a blockchain-based smart parking system with privacy preservation and reputation management. In recent years, a promising blockchain technology with advantages of decentralization, security and transparency has been used in different applications, such as intelligent transportation systems [22], [23], eHealth systems [24], smart grids [25], human resource record management [26], wireless sensor networks [27], etc. Creating a blockchain network from the parking lots to manage the parking service could tackle the previously mentioned security issues as follows. First, the blockchain can decentralize the parking system making it robust against single point of failure and DDoS attacks that threaten the parking service availability. Second, blockchain can make our system transparent and guarantee the integrity of the system, because the parking offers of all parking lots can be recorded in a shared and immutable ledger, where the content of the ledger is agreed on by the blockchain validators (e.g., parking lots) through a predefined consensus algorithm and cannot be changed once recorded in the ledger. However, using the blockchain still cannot tackle the location privacy issues, if the drivers expose their desired destinations to the blockchain. Moreover, the blockchain validators may not serve drivers who query parking slots outside their area. To provide high protection to location privacy, we use a private information retrieval (PIR) technique to enable drivers to privately retrieve parking offers in the area of interest from the blockchain without revealing any information about their destinations. The PIR is also necessary to encourage each parking lot to serve all drivers even if they search for parking slots outside their area. We also leverage blockchain and smart contract technology to enable drivers to pay for the parking service in an anonymous and decentralized manner. Moreover, we use blockchain to integrate an anonymous and decentralized reputation management scheme into our parking system. The scheme boosts mutual trust among drivers and parking lots and guarantees transparency and reliability, where the management of the reputation scores is verifiable by all drivers and parking lots.

To the best of our knowledge, this is the first work that uses the blockchain to decentralize smart parking systems for public parking lots, while preserving the driver's location privacy. Our main contributions in this article can be summarized as follows:

- We propose a decentralized and privacy-preserving parking system. To preserve drivers' locations, unlike the existing location obfuscation techniques that reveal coarse location information, the PIR technique has been adopted so that drivers can privately retrieve parking offers from the blockchain without leaking any information on the locations of the parking lots of interest. We also use an efficient anonymous authentication scheme based on short randomizable signature to allow drivers to make parking reservations anonymously without revealing their real identities. We adopt anonymity-yet-accountability concept, where a trusted authority can identify and revoke drivers in case of misbehavior.

- Instead of using centralized payment methods (e.g., debit or credit cards) that can breach drivers privacy, we use a time-locked and anonymous payment by leveraging smart contract and blockchain technology and integrate it into the parking system. This payment is done in a trust-less manner between parking lots and drivers.
- We integrate an anonymous reputation management scheme based on the blockchain technology into our parking system so that drivers can rate the parking lots' services. Thus, each parking lot has a reputation score that reflects the quality of the service provided in the past. This score enables drivers to select good parking lots.
- We analyze the security and privacy of the proposed system and evaluate the overhead. Our analysis and evaluations confirm that our system is secure, privacy-preserving and has low communication and computation overheads.

The remainder of this article is organized as follows. Section II presents preliminaries and necessary background. The system model and design objectives are explained in section III. The proposed scheme is presented in details in section IV. The security and privacy analysis, and performance evaluation are provided in section V. Section VI discusses the related work. Finally, conclusions and future work are given in section VII.

II. PRELIMINARIES

In this section, we present the necessary background on blockchain and smart contracts, PIR, and cryptographic primitives that are used in our system.

A. BLOCKCHAIN AND SMART CONTRACTS

Blockchain is a verifiable, immutable, and distributed ledger that allows mistrusting entities to transact with each other and agree on one shared ledger without relying on a central party [28]. Blockchain provides transparency, where transactions data are recorded on a shared ledger that is organized as a chain of blocks and managed by a network of computers, called validators, running a peer-to-peer (P2P) protocol. Each block includes a set of messages (transactions) committed by the network peers and is validated by the whole network through a pre-defined consensus algorithm [29]. The underlying consensus and incentive mechanisms of the blockchain technology also contribute to boosting the mutual trust among participants in the blockchain network. Generally, there are two broad categories of the blockchain [30], a permissionless blockchain, where any one can join the blockchain network and all the users have the same read/write access rights, and a permissioned blockchain, where the read/write access rights are permitted to authorized users. In this article, we use a consortium blockchain [30], which is a type of the permissioned blockchians, where only a group of authorized (certified) nodes have the write permission on the shared ledger, and the other users can only query the blockchain to read data from the ledger. This is suitable for our parking system because only a group of users (the parking lots in our system) have parking offers and need to write them on the shared ledger so they act as the blockchain validators, while the other users (the drivers in our system) need to query the blockchain to retrieve the parking offers.

Regarding the consensus algorithms used in the blockchain to enable validators to agree on the same ledger and verify that each block has been added to the ledger correctly, they are divided into two categories, including proof-based and voting-based algorithms [31]. In this article, we use the Raft consensus algorithm [32], which belongs to the second category. The Raft is an efficient algorithm that provides faster consensus for the blockchain validators than proof-based consensus algorithms, such as proof of work or proof of stake. In Raft, there is a voting every period of time to select a leader. The role of the leader is to append a new block to the existing chain of blocks. A validator in Raft can have one of three different roles: follower, candidate, or leader. Initially, all the validators start as followers. Then, any follower who intends to become a leader should transfer to the candidate state for a period of time (election period) in order to collect enough votes to become a leader. During the election period, all the follower nodes vote for one of the candidates. After the election period timeout, the candidate which collects more votes becomes a leader. The Raft consensus algorithm has been used in Quorum blockchain of JPMorgan system [33]. We refer to [29]–[32] for more details on the consortium blockchain and the Raft consensus algorithm.

Furthermore, blockchain enables the essence of smart contracts [29], [30] which can be defined as a computer program code that is capable of facilitating, executing, and enforcing the execution of an agreement (i.e., contract) atop of a blockchain without any intervention from a third party. Each smart contract has a unique address on the blockchain. This address is used to identify the contract in the network, which enables users or other contracts to interact with it. Smart contracts can help users to exchange money, property, shares, or anything of value in a transparent way.

B. PRIVATE INFORMATION RETRIEVAL (PIR)

The PIR techniques [34] enable a user to retrieve or download a specific data (file) from a storage system (database) without revealing any information about the file being requested. This is very useful, when we want to preserve the user privacy from the storage system. Instead of storing encrypted offers, receiving encrypted requests from the users, and matching the encrypted requests with the encrypted offers, it is more efficient to store the data in the system without encryption, and allow the user to use a PIR technique to privately retrieve the required offers. The PIR techniques have been used in different applications, such as location-based services [35] and eHealth systems [36]. In location-based services, the PIR is used to enable a user to privately query a server to retrieve location information about his/her point of interests (e.g., locations of entertainment centers, shopping malls, restaurants, etc.). In eHealth systems, the PIR is used to privately retrieve medical records from a health care service provider. In our system, we use PIR to enable a driver to retrieve all parking offers in a geographic area, called a cell, without revealing the cell of interest in order to preserve the location information of drivers. This fits our model as every driver (user in PIR) needs to query the blockchain (distributed databases in PIR) for parking offers within a certain cell without revealing the driver's interest in a specific parking offer. Moreover, since the parking system is run by the parking lots, they may not respond to a query if it is outside their area. PIR technique can eliminate this problem by hiding the area of interest from the parking lots.

In our system, we modified the PIR scheme in [34] to fit our model. This scheme is an information-theoretic PIR scheme for retrieving data from maximum distance separable (MDS)-coded databases. In this scheme, one database is stored on several servers that may collude during the private information retrieval to infer information on the requested data. Also, some servers, called unresponsive servers, may not respond to users' queries, and some servers, called Byzantine servers, may return non-intentional erroneous responses during the private information retrieval. Note that the erroneous responses may exist because of the communication channel. The reason we leverage the scheme in [34] instead of the capacity achieving scheme in [37], which achieves the lowest communication overhead (size of downloaded data) when the size of the stored files is small, is that as the size of the stored files increases, the communication overhead exponentially increases [37]. However, the communication overhead in the scheme in [34] does not exponentially increase when the size of stored files increases. This is desirable in our system because the number of parking offers and their size may be large. The information-theoretic PIR scheme that we use can mitigate b Byzantine blockchain nodes and r unresponsive nodes without leaking any information about the requested offers to any set of t colluding nodes.

C. CRYPTOGRAPHIC PRIMITIVES

1) BILINEAR MAPPING

Given G_1, G_2 and G_T as three cyclic multiplicative groups of prime order p, where g_1 is the generator of G_1 and g_2 is the generator of G_2 [38], a bilinear map e is defined as $e : G_1 \times G_2 \rightarrow G_T$, and should have the following properties:

- $e(g_1, g_2)$ is computed efficiently for all $g_1 \in G_1$ and $g_2 \in G_2$.
- Bilinear: $e({g_1}^a, {g_2}^b) = e(g_1, g_2)^{ab}$, $\forall\, g_1 \in G_1,\ g_2 \in G_2$ and $a, b \in Z_q^*$, where Z_q^* is a finite field of order q.
- Non-degenerate: $e(g_1, g_2) \neq 1$.

2) SHORT RANDOMIZABLE SIGNATURE

To provide efficient anonymous authentication with conditional anonymity, [39] has proposed the short randomizable signature scheme. In our system, a driver uses this short randomizable signature to anonymously authenticate him/herself to parking lots during parking reservation. The signature scheme enables a driver to get a signature on committed messages (secret values) from KDC. Then, the received signature can be randomized several times and used as anonymous credentials. Note that these credentials cannot be linked by any entity in the system except KDC, which is desirable to revoke the anonymity and identify malicious drivers if needed. The short randomizable signature scheme is efficient in terms of communication and storage overheads because the credentials can be computed locally by the driver instead of computing and distributing them by the KDC and storing them by the driver's smart phone.

3) ZERO-KNOWLEDGE PROOF

The zero knowledge proof technique allows one entity *(prover)* to convince another entity *(verifier)* that she possesses a secret value s satisfying a public verifiable relation R without revealing any information about this secret. In our system, the short randomizable signature scheme uses the zero knowledge proof proposed in [40].

4) MULTI-SIGNATURE

The multi-signature schemes are used if we want multiple signatures from different signers on the same message. Instead of sending multiple signatures to prove that multiple parties approve a message, only one signature, called multi-signature, is sent to the verifier. To verify a message, the verifier should use the public keys of all signers. If the verification is valid, this means that all the signers signed the message correctly. The size of the multi-signature is the same as that of the individual signature. Moreover, the verifier has to verify only one signature instead of multiple signatures. Thus, the multi-signature scheme is efficient in terms of communication and computation overheads. In our system, the multi-signature scheme of [41] will be used in the *parking and payment* and *reputation update* phases.

III. NETWORK/THREAT MODELS AND DESIGN GOALS

In this section, we present the considered network model followed by the threat model, and then the design objectives of our system.

A. NETWORK MODEL

As illustrated in Fig. 1, the considered network model consists of four entities: a key distribution center (KDC), a consortium blockchain network, parking lots, and drivers.

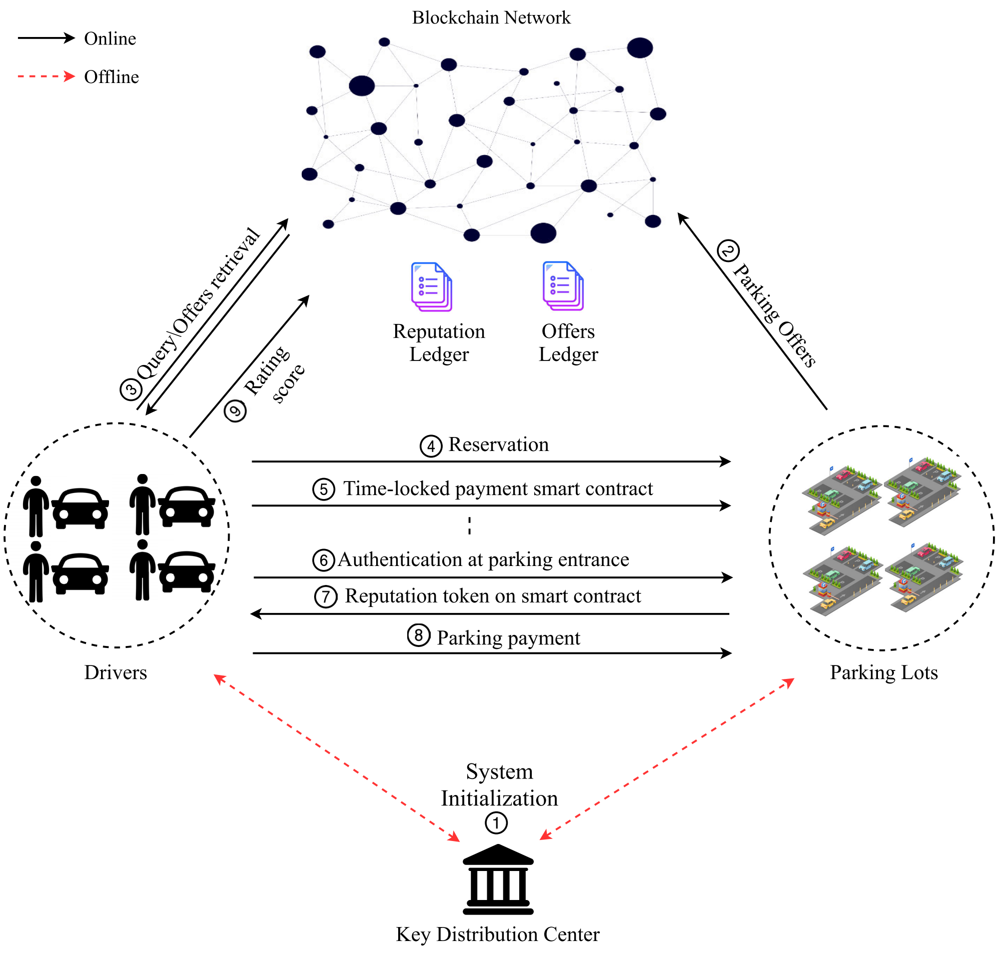

FIGURE 1. An illustration for the network model.

- **KDC:** The KDC is responsible for initializing the whole system including registering drivers and parking lots, generating cryptography public parameters, and distributing keys. Note that this role does not conflict with the decentralization nature of the system since the KDC is just a system initializer, which is not involved in managing the parking service. Practically, the KDC can be the Ministry of Transport, for example.
- **Consortium Blockchain Network:** We use the consortium blockchain to allow different parking lots that do not trust each other to act as validators to run the system. The consortium blockchain network creates/updates two ledgers for available parking slots and reputation scores of parking lots.
- **Parking Lots (PLs):** PLs publish their offers on the blockchain. They are owned by different parties that have conflicting goals and do not trust each others. They also act as validators in the blockchain network.
- **Drivers:** Drivers use their smartphones to communicate with the blockchain network to retrieve parking offers, and submit ratings for the service. Also, they communicate with PLs to make online reservations.

B. ADVERSARY AND THREAT MODEL

The KDC is trusted because it is operated by the government which is interested in the security of the smart parking system. We follow the standard blockchain threat model defined in [42], in which the blockchain network is regarded as a conceptual trusted party that is trusted for correctness and availability, but distrusted for privacy (i.e., the blockchain nodes may misuse the information stored in the ledger or try to infer more information). We assume the blockchain in our system is maintained by a set of validators, which execute the PIR technique correctly. However, at most t out of n blockchain validators may collude during the execution of the PIR to infer information about drivers' parking locations of interest. Also, at most b out of n blockchain validators may return non-intentional erroneous responses resulting from the communication channel, which we refer to them as Byzantine validators. Note that the threshold numbers (b and t) can be controlled in our system to achieve different security/privacy levels.

Some parking lots may wait to see the parking offers of the other lots before deciding their offers, and then they manipulate (increase/decrease) their prices to achieve financial gains unfairly. Also, some parking lots may not issue reputation tokens to the drivers who parked in their lots to prevent them from submitting rating scores. As will be explained later, drivers need these tokens to be able to rate parking service. In addition, some parking lots may try to know the individual ratings of the drivers.

Some drivers may be malicious. Specifically, a malicious driver may reserve multiple parking slots without commitment for these reservations, causing financial losses to the parking lots. Also, drivers may attempt to park without paying the parking fees. They may also try to pollute the reputation scores by submitting ratings for parking events that did not happen. Furthermore, a malicious driver who parked in a parking lot and received a reputation token, may attempt to use the token to submit multiple ratings to the reputation management scheme for the same parking event to unfairly reduce/increase the parking lot's reputation score significantly. A malicious driver may also use the reputation token to rate other parking lots.

External attackers may try to access the system to get the parking service without registration. Also, they may eavesdrop on the communications in the system to infer drivers' sensitive information or launch impersonation and forgery attacks.

C. DESIGN GOALS

We have used the same methodology used in [43], [44] to define the security and privacy goals of our system as follows.

- **Decentralization.** The proposed system should not rely on a central party to manage the parking service. As discussed earlier, centralized systems are vulnerable to single point of failure and DDoS attacks and suffer from lack of transparency.
- **Preserving drivers' privacy.** To preserve drivers' privacy, we have used the methodology used in [43], [44] to define the following goals that should be achieved by our system.
  - **Driver anonymity.** The drivers' real identities should be protected from blockchain nodes, parking lots, and external adversaries, i.e., no one can know the driver's real identity from his transmitted messages.
  - **Driver untraceability.** Even if the driver's real identity is concealed, one can try to guess it by tracing the driver's locations or by linking the transmitted messages sent from the driver. Thus, our system should achieve the following objectives.
    1) The drivers' visited locations or parking information (e.g., desired parking destination, times, and periods) are preserved from blockchain nodes, parking lots, and external adversaries. Only the selected parking lot knows the driver's parking information when his/her vehicle presents and parks in the lot.
    2) The messages of available parking queries and parking reservations of a driver at different occasions should not be linked by blockchain nodes, parking lots, and external adversaries.
    3) The payment is done without learning that a driver parked in a particular parking lot.
    4) The rating scores of a driver are anonymous and confidential, and cannot be linked to a driver.
- **Anonymous authentication.** Only legitimate drivers should be allowed to participate in the system (i.e., do reservations, and submit rating scores) anonymously without revealing their real identities.
- **Discouraging uncommitted reservations.** The system should discourage drivers from reserving multiple parking slots without commitment to these reservations because this may deprive other drivers from available parking slots and cause financial loses to the parking lots.
- **Accountability.** The KDC should be able to trace and disclose the real identity of a misbehaved driver if needed.
- **Fair parking rates.** If parking lots know the offers of other lots, they may deliberately manipulate (increase/decrease) their prices to achieve financial goals unfairly. Therefore, the system should prevent a parking lot from learning the offers of the competitors before it decides its offer to ensure fair parking rates.
- **Secure and privacy-preserving reputation management.** The quality of the service provided by the parking lots should be continuously evaluated using a reputation management scheme so that parking lots which provide bad service can be identified by low reputation scores. However, this scheme should not jeopardize the drivers' privacy and should be secure against the malicious actions from both the parking lots and the drivers.
- **Transparency.** The process of managing parking service should be transparent to participants. Also, the process of updating the reputation scores should be verifiable and transparent to all parking lots.

IV. PROPOSED SYSTEM

In this section, we present our blockchain-based and privacy-preserving smart parking system.

A. OVERVIEW

As demonstrated in Fig. 1, our system consists of the following phases: *system initialization*, *parking offers submission*, *parking offers retrieval*, *online parking reservation*, *payment and parking*, and *reputation update*. During the *system initialization* phase, the KDC distributes public key certificates to parking lots and anonymous credentials to drivers. In the *parking offers submission* phase, the parking lots provide periodic parking offers to the blockchain network. Then, the validators verify the transactions and record them on the ledger. Then, in the *parking retrieval* phase, a driver sends a query using PIR to the blockchain to privately retrieve the offers in a desired cell. In the *online parking reservation* phase, the driver anonymously sends a reservation request to the parking lot. In the *payment and parking* phase, the driver sends a time-locked payment smart contract for the selected parking lot. Then, at the parking entrance, the driver authenticates to the parking lot to prove that he/she made a reservation. At the end of the parking, the parking lot submits a proof to the smart contract to confirm that the parking has ended to receive the payment. Also, this proof is used as a reputation token to enable the driver to submit a feedback (rating score) about the parking service. Finally, in the *reputation update* phase, the driver anonymously sends a rating to the parking service. Note that the notations used in the coming sections are given in Table 1.

TABLE 1. Notations.

Notation	Description
$(g_1, g_2, p, G_1, G_2, e, H, H_1)$	System public parameters
$(x, y), (g_2, \tilde{X}, \tilde{Y})$	KDC's group secret key, group public key
$\mathcal{PL}_j,\ ID_j$	Parking lot, real identity of the parking lot
$PK_{\mathcal{PL}_j}, SK_{\mathcal{PL}_j}$	$\mathcal{PL}_j$ public/private key pair
$Add_{\mathcal{PL}_j}$	$\mathcal{PL}_j$ Ethereum address
$\mathcal{D}_i,\ ID_{\mathcal{D}_i}$	Driver, real identity of the driver
$gsk_{\mathcal{D}_i}$	$\mathcal{D}_i$ group secret key
$(\sigma^{[1]}_{\mathcal{D}_i}, \sigma^{[2]}_{\mathcal{D}_i}, \sigma^{[3]}_{\mathcal{D}_i})$	$\mathcal{D}_i$ anonymous credential
$PK_{\mathcal{D}_i}, SK_{\mathcal{D}_i}$	$\mathcal{D}_i$ one-time public/private key pair for reservation
$\sigma_{SK_{\mathcal{D}_i}}$	$\mathcal{D}_i$ digital signature using his/her private key
$C^R_{\mathcal{D}_i}$	$\mathcal{D}_i$ encrypted reservation request
$Sig_{\mathcal{D}_i}(C^R_{\mathcal{D}_i})$	$\mathcal{D}_i$ short randomizable signature on $C^R_{\mathcal{D}_i}$
n	Number of blockchain validators
t, b, r	Number of colluding, byzantine, unresponsive validators
PK_v, SK_v	Public/private key pair of an aggregator in the reputation management scheme
PK	Public key computed from the multiplication of aggregators' public keys
T_i	Reputation token
$rs_{i,j}$	Rating score from $\mathcal{D}_i$ to the parking lot $\mathcal{PL}_j$
$\Lambda_{i,j}$	Encrypted rating score from $\mathcal{D}_i$ to the parking lot $\mathcal{PL}_j$
l_{add}	Address of the payment smart contract
$\mathcal{RS}_{\mathcal{PL}_j}$	Total reputation score of the parking lot $\mathcal{PL}_j$
$\mathcal{T}_f$	A threshold number of uncommitted reservations

B. SYSTEM INITIALIZATION

In the *system initialization* phase, the KDC generates the public key certificates for parking lots and anonymous credentials for drivers. Each parking lot, $\mathcal{PL}_j$, creates public and private keys ($PK_{\mathcal{PL}_j}$ and $SK_{\mathcal{PL}_j}$) (e.g., using the Schnorr multi-signature algorithm in [41]) and sends $PK_{\mathcal{PL}_j}$ to the KDC to issue a certificate ($Cert_j$) and send it to $\mathcal{PL}_j$. Moreover, the KDC initializes the short randomizable signature [39] as follows:

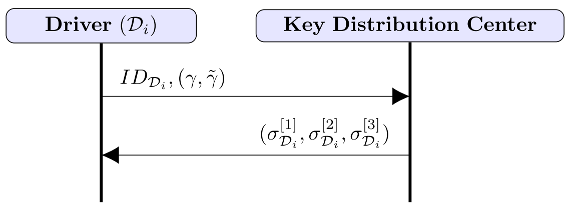
Driver $(\mathcal{D}_i)$
Key Distribution Center
$ID_{\mathcal{D}_i}, (\gamma, \tilde{\gamma})$
$(\sigma^{[1]}_{\mathcal{D}_i}, \sigma^{[2]}_{\mathcal{D}_i}, \sigma^{[3]}_{\mathcal{D}_i})$

FIGURE 2. An illustration for the driver registration.

Consider $e : G_1 \times G_2 \rightarrow G_T$ a cryptographic bilinear map with generators $g_1 \in G_1$ and $g_2 \in G_2$, where G_1, G_2 and G_T are three cyclic multiplicative groups of order p which is a large prime number. Firstly, the KDC generates the public parameters $(g_1, g_2, p, G_1, G_2, e, H, H_1)$, where H is a collision resistant hash function like $SHA-2$, i.e., H: $\{0, 1\}^* \rightarrow \{0, 1\}^{224}$ and H_1 is a hash function that maps to Z_p, i.e., H_1: $\{0, 1\}^* \rightarrow Z_p$, where Z_p is a finite field of order p. Then, it selects two random numbers $(x, y) \in Z_p^2$ as group secret key. After that, the KDC computes $(\tilde{X}, \tilde{Y}) \leftarrow (g_2^x, g_2^y)$, and sets the group public key as $(g_2, \tilde{X}, \tilde{Y})$.

As shown in Fig. 2, a driver $\mathcal{D}_i$, whose real identity is $ID_{\mathcal{D}_i}$, can register to the KDC to obtain credentials as follows. He/She generates a secret key by randomly selecting $a_1 \in Z_p$ and computes a public key $A = g_1^{a_1}$. The driver randomly selects $a_2 \in Z_p$, computes the pair $(\gamma, \tilde{\gamma}) \leftarrow (g_1^{a_2}, \tilde{Y}^{a_2})$ and uses the secret key (a_1) to generate a signature $\eta \leftarrow \sigma_{a_1}(\gamma)$. Then, he/she sends $ID_{\mathcal{D}_i}$, $(\gamma, \tilde{\gamma})$ and η to the KDC through a secure communication channel. The KDC verifies the signature η by checking $e(\gamma, \tilde{Y}) \overset{?}{=} e(g_1, \tilde{\gamma})$. Then, the driver invokes an interactive zero knowledge proof scheme, such as Schnorr's scheme [40], to prove the knowledge of a_2 to the KDC. If the proof is valid, the KDC randomly selects $k \in Z_p$ to compute $(\sigma^{[1]}_{\mathcal{D}_i}, \sigma^{[2]}_{\mathcal{D}_i}, \sigma^{[3]}_{\mathcal{D}_i})$ as follows:

$$\sigma^{[1]}_{\mathcal{D}_i} = g_1^k \tag{1}$$

$$\sigma^{[2]}_{\mathcal{D}_i} = (g_1^x \cdot \gamma^y)^k \tag{2}$$

$$\sigma^{[3]}_{\mathcal{D}_i} = e(g_1^k, \tilde{Y}) \tag{3}$$

The KDC stores $\{ID_{\mathcal{D}_i}, \gamma, \eta, \tilde{\gamma}\}$ in a tracking list and returns $(\sigma^{[1]}_{\mathcal{D}_i}, \sigma^{[2]}_{\mathcal{D}_i}, \sigma^{[3]}_{\mathcal{D}_i})$ to the driver. This list is used when KDC traces a signature to identify the signer, as will be explained later. Then, the driver sets his/her group secret key as:

$$gsk_{\mathcal{D}_i} = (a_2, \sigma^{[1]}_{\mathcal{D}_i}, \sigma^{[2]}_{\mathcal{D}_i}, \sigma^{[3]}_{\mathcal{D}_i}) \tag{4}$$

Note that $(\sigma^{[1]}_{\mathcal{D}_i}, \sigma^{[2]}_{\mathcal{D}_i}, \sigma^{[3]}_{\mathcal{D}_i})$ can be randomized and used as anonymous credentials by the driver for anonymous authentication, as will be explained later.

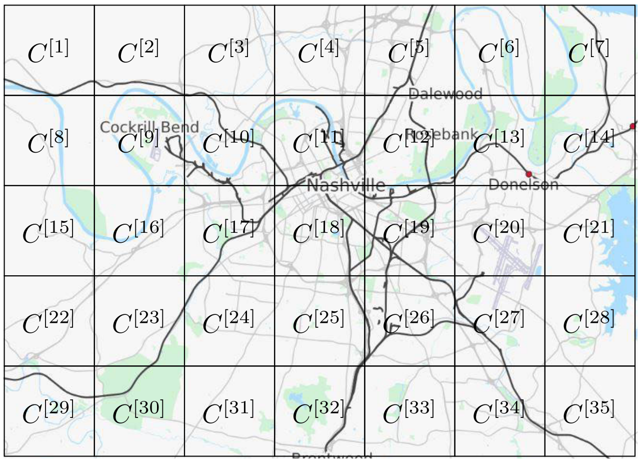

FIGURE 3. The map of Nashville city, TN, USA is divided into multiple geographical areas (cells) and $C^{[m]}$ is the identifier of cell m.

C. PARKING OFFERS SUBMISSION

In this phase, $\mathcal{PL}_j$ submits parking offers to the blockchain validators. We assume that an area $\mathcal{A}$ (e.g., a city), where the smart parking system is deployed, is divided into small geographic areas, called cells, as shown in Fig. 3 and each cell has an identifier, e.g., $C^{[m]}$ is the identifier of cell m. To submit parking offers every period of time, $\mathcal{PL}_j$ composes a parking offer that includes the following information: real identity (ID_j), public key certificate ($Cert_j$), number of available parking slots (N), cell Identifier ($C^{[m]}$), exact location of the parking lot (loc), available services (S) (e.g., charging station, car wash, etc.), price (pr), vector for available parking times (t_a), and its recent reputation score ($\mathcal{RS}_{\mathcal{PL}_j}$).

$$Offer : < ID_j, Cert_j, N, C^{[m]}, loc, S, pr, t_a, \mathcal{RS}_{\mathcal{PL}_j} > \quad (5)$$

Then, $\mathcal{PL}_j$ commits to the parking offer by computing $H(Offer \| Nonce)$, where $Nonce$ is a random number. Note that the commitment prevents a parking lot from learning the offers of competitors before deciding its offers to ensure fair parking rates. If a parking lot knows the offers of other parking lots, it may deliberately manipulate (increase or decrease) its prices to boost its chance to book its parking slots. Then, each committed offer is signed with the secret key ($SK_{\mathcal{PL}_j}$) of $\mathcal{PL}_j$ and broadcasted as a transaction to the validators of the blockchain network. Then, the validators verify the signatures to ensure that the commitment of a parking offer is coming from an authorized parking lot.

After the signature verification, each $\mathcal{PL}_j$ opens its committed offer by sending the offer packet given in Eq. 5 and the $Nonce$ value to the validators. Then, the validators verify whether the reputation score ($\mathcal{RS}_{\mathcal{PL}_j}$) in the parking offer is similar to the score recorded on the reputation ledger.

1) BLOCKS GENERATION

To generate a new block, the Raft consensus algorithm [32] is run by all validators to agree on the content of the ledger. As mentioned in section II-A, the leader is responsible for updating the blockchain ledger in the Raft algorithm. When the leader receives new parking offers, it creates a new block containing these offer transactions so that offers are organized based on the cell identifier $C^{[m]}$, where $m \in \{1, \cdots, M\}$ as shown in Fig. 4. Then, the leader broadcasts the block to the followers (i.e., the rest of the blockchain validators). Then, it waits for acknowledgements (votes) from the majority of the followers to append this new block to the offers ledger. This new block is signed with the leader's secret key as shown in Fig. 4, and broadcasted to the followers to be appended in their local copies of the offers ledger. The leader's signature guarantees the integrity of the ledger because no follower node would be able to rewrite the content because they do not have the leader's secret key to compute its signature. For the PIR technique to work efficiently, the parking offers in each cell are represented in the form of $L \times 1$ matrix on the ledger, where L is the number of symbols representing the offers.

D. PARKING OFFERS RETRIEVAL

In this phase, a driver $\mathcal{D}_i$ wants to retrieve the parking offers in the d^{th} cell from n blockchain validators without leaking

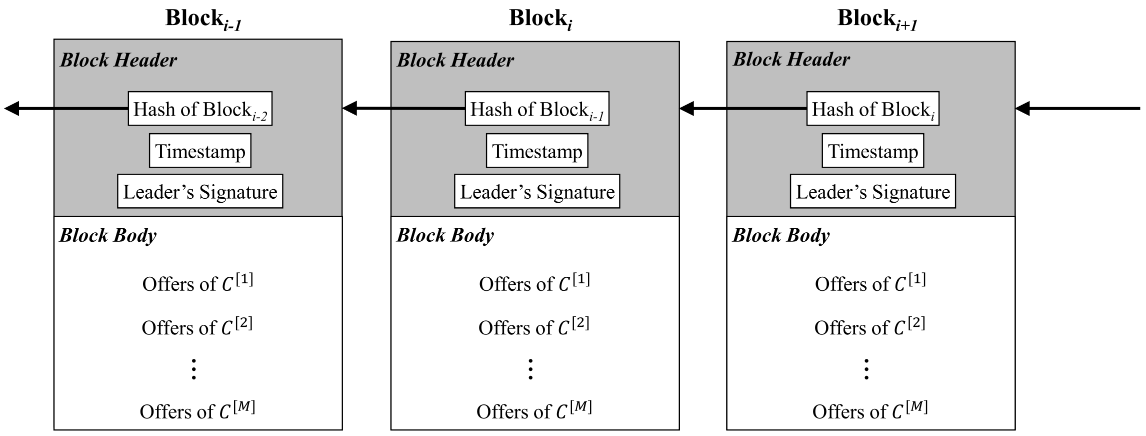

FIGURE 4. The structure of the offers ledger.

any information (in information-theoretic sense) about the identifier of the requested cell (d). We preserve the privacy of the drivers from any group of t colluding validators even if there exist b Byzantine validators that respond with erroneous responses and r unresponsive validators.

To that end, we assume that all the parking offers in one cell are represented by$L$ symbols, $O^{[d]} = \{O_1^{[d]}, \cdots, O_L^{[d]}\}$, where $L = n-t-2b-r$. To retrieve the offers in the desired cell $C^{[d]}$, a driver $\mathcal{D}_i$ chooses independent and identically distributed (i.i.d.) and uniformly distributed codewords from a query code $\mathcal{C}_q$, which is an $[n, t]$ Reed-Solomon code. The purpose of this randomness is to hide the identifier of the cell of interest from any t colluding validators. The codewords can be represented as evaluations of a polynomial $\beta_\ell^m(z)$, where $\ell \in \{1, \cdots, L\}$, and $m \in \{1, \cdots, M\}$. The query polynomial, $\mathcal{Q}_\ell^m(z)$ can be written as:

$$\mathcal{Q}_\ell^m(z) = \begin{cases} \beta_\ell^m(z) + z^{n-2b-r-\ell} & m = d \\ \beta_\ell^m(z) & m \neq d \end{cases} \tag{6}$$

Now, $\mathcal{D}_i$ prepares the query to the jth blockchain validator by evaluating these polynomials at $z = \alpha_j$, where $\alpha_j \in \mathbb{F}$, which is a finite field with a sufficiently large alphabet (to realize the Reed-Solomon codes). Hence, the query vector to the jth validator $\mathcal{Q}_j$ is given by:

$$\mathcal{Q}_j = (\mathcal{Q}_1^1(\alpha_j), \cdots, \mathcal{Q}_L^1(\alpha_j), \cdots, \mathcal{Q}_1^M(\alpha_j), \cdots, \mathcal{Q}_L^M(\alpha_j)) \tag{7}$$

When the jth blockchain validator receives the query, it performs an inner product between $\mathcal{Q}_j$ and the vector of content (the parking offers) $\mathcal{Y}_j = (O_1^{[1]}, \cdots, O_L^{[1]}, \cdots, O_1^{[M]}, \cdots, O_L^{[M]})$. Hence, the response of the jth validator is:

$$\begin{aligned} \mathcal{R}_j &= \mathcal{Q}_j^T \mathcal{Y}_j \\ &= \sum_{m=1}^{M} \sum_{\ell=1}^{L} \mathcal{Q}_\ell^m(\alpha_j) O_\ell^{[m]} \\ &= \sum_{m=1}^{M} \sum_{\ell=1}^{L} \beta_\ell^m(\alpha_j) O_\ell^{[m]} + \sum_{\ell=1}^{L} \alpha_j^{n-2b-r-\ell} O_\ell^{[d]} \end{aligned} \tag{8}$$

Eq. 8 can be rewritten as an evaluation of the polynomial $\mathcal{R}(z)$ as:

$$\mathcal{R}(z) = \sum_{m=1}^{M} \sum_{\ell=1}^{L} \beta_\ell^m(z)\, O_\ell^{[m]} + \sum_{\ell=1}^{L} z^{n-2b-r-\ell} O_\ell^{[d]} \tag{9}$$

To show the decodability of $\mathcal{R}(z)$ received from the blockchain validators to get the parking offers in the desired cell, we note that the degree of $\mathcal{R}(z)$ is $n-2b-r-1$, hence, the responses of the n blockchain validators are codewords from an $[n, n-(2b+r)]$ Reed-Solomon code. An $[n, n-(2b+r)]$ Reed-Solomon code is capable of correcting b errors (which results from b Byzantine validators) and r erasures (which results from r unresponsive validators). Therefore, with applying Reed-Solomon decoding techniques, $\mathcal{D}_i$ can decode the parking offers in the desired cell $C^{[d]}$ correctly.

For the retrieval rate, the driver can retrieve L symbols representing the parking offers in the cell of interest from $n-r$ responsive validators, consequently, the retrieval rate is given by:

$$R_{PIR} = \frac{L}{n-r} = \frac{n-t-2b-r}{n-r} \tag{10}$$

E. ONLINE PARKING RESERVATION

In this phase, once the driver retrieves all the available parking offers in the desired cell, he/she selects one offer and starts the online parking reservation process as follows.

First, $\mathcal{D}_i$ generates a *fresh* public/private key pair $(PK_{\mathcal{D}_i}, SK_{\mathcal{D}_i})$ for the schnorr multi-signature algorithm [41] and sends a reservation request to the selected $\mathcal{PL}_j$. Note that the key pair is used only once to avoid linking reservation requests and $\mathcal{D}_i$ can select the best offer based on the distance to the desired destination, price, parking lot reputation score, and offered service (e.g., charging station, car wash, etc.). Then, $\mathcal{D}_i$ composes a parking reservation request that includes the public key $(PK_{\mathcal{D}_i})$, estimated parking period time (t_p), and timestamp (TS). Then, he/she encrypts this information using the $\mathcal{PL}_j$ public key $(PK_{\mathcal{PL}_j})$ as follows:

$$C_{\mathcal{D}_i}^R = Enc_{PK_{\mathcal{PL}_j}}(PK_{\mathcal{D}_i}, t_p, TS) \tag{11}$$

Then, $\mathcal{D}_i$ uses the group secret key $gsk_{\mathcal{D}_i}$ to generate a short randomizable signature on $C_{\mathcal{D}_i}^R$ as follows. First, $\mathcal{D}_i$ randomizes the credential $(\sigma_{\mathcal{D}_i}^{[1]}, \sigma_{\mathcal{D}_i}^{[2]}, \sigma_{\mathcal{D}_i}^{[3]})$ received from the KDC by selecting two random numbers $r_1, r_2 \in Z_p^2$ and computing the following values:

$$\sigma_{\mathcal{D}_i}^{[1]'} = (\sigma_{\mathcal{D}_i}^{[1]})^{r_1} \tag{12}$$

$$\sigma_{\mathcal{D}_i}^{[2]'} = (\sigma_{\mathcal{D}_i}^{[2]})^{r_1} \tag{13}$$

$$\sigma_{\mathcal{D}_i}^{[3]'} = (\sigma_{\mathcal{D}_i}^{[3]})^{r_1\ r_2} \tag{14}$$

$$c_{\mathcal{D}_i} = H_1(\sigma_{\mathcal{D}_i}^{[1]'}, \sigma_{\mathcal{D}_i}^{[2]'}, \sigma_{\mathcal{D}_i}^{[3]'}, C_{\mathcal{D}_i}^R) \tag{15}$$

$$s = r_2 + c_{\mathcal{D}_i} \cdot a_2 \tag{16}$$

where $a_2 \in Z_p$ is the secret random number which is included in his/her $gsk_{\mathcal{D}_i}$. Note that the tuple $(\sigma_{\mathcal{D}_i}^{[1]'}, \sigma_{\mathcal{D}_i}^{[2]'}, c_{\mathcal{D}_i}, s)$ is the driver's short randomizable signature on the ciphertext $C_{\mathcal{D}_i}^R$, which can be abbreviated as $(Sig_{\mathcal{D}_i}(C_{\mathcal{D}_i}^R))$.

Then, as shown in Fig 5, the driver $\mathcal{D}_i$ sends the parking reservation request, which includes both the ciphertext $C_{\mathcal{D}_i}^R$ along with the short randomziable signature $Sig_{\mathcal{D}_i}(C_{\mathcal{D}_i}^R)$ to the $\mathcal{PL}_j$. Then $\mathcal{PL}_j$ verifies $Sig_{\mathcal{D}_i}(C_{\mathcal{D}_i}^R)$ to ensure that the request

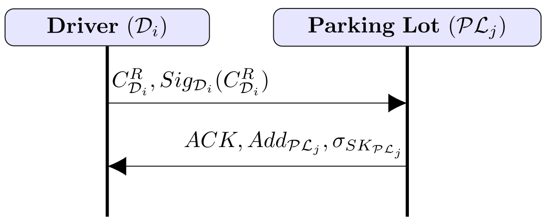
Driver $(\mathcal{D}_i)$
Parking Lot $(\mathcal{PL}_j)$
$C_{\mathcal{D}_i}^R, Sig_{\mathcal{D}_i}(C_{\mathcal{D}_i}^R)$
$ACK, Add_{\mathcal{PL}_j}, \sigma_{SK_{\mathcal{PL}_j}}$

FIGURE 5. An illustration for the *online parking reservation* phase.

is sent from a legitimate driver, i.e., registered driver. To do so, $\mathcal{PL}_j$ computes:

$$V = e\left(\sigma^{[1]'}_{\mathcal{D}_i}, \tilde{X}\right)^{c_{\mathcal{D}_i}} \cdot e\left(\sigma^{[2]'}_{\mathcal{D}_i}, g_2\right)^{-c_{\mathcal{D}_i}} \cdot e\left(\sigma^{[1]'}_{\mathcal{D}_i}, \tilde{Y}\right)^{s} \quad (17)$$

Then, it checks the following equality:

$$c_{\mathcal{D}_i} \overset{?}{=} H_1(\sigma^{[1]'}_{\mathcal{D}_i}, \sigma^{[2]'}_{\mathcal{D}_i}, V,\ C^R_{\mathcal{D}_i}) \quad (18)$$

Then, $\mathcal{PL}_j$ decrypts the ciphertext $C^R_{\mathcal{D}_i}$ and sends an acknowledgement (ACK) including its Ethereum address ($Add_{\mathcal{PL}_j}$) and its signature to the driver. Note that the signature can prove to $\mathcal{D}_i$ that the message is sent from $\mathcal{PL}_j$ and thus the Ethereum address is correct.

F. PAYMENT AND PARKING

In this phase, we discuss the time-locked payment that is used for confirming the reservation, the authentication of the driver at the parking lot entrance, and finalizing the payment. After $\mathcal{D}_i$ receives the ACK from $\mathcal{PL}_j$, $\mathcal{D}_i$ initializes a time-locked smart contract with sufficient amount of coins as a deposit to be used for the parking payment. Algorithm 1 gives the pseudocode of the time-locked parking payment contract. In the smart contract, $\mathcal{D}_i$ includes the amount of payment (balance), his/her Ethereum address and reservation public key $\mathcal{PK}_{\mathcal{D}_i}$, and the parking lot Ethereum address $Add_{\mathcal{PL}_j}$ and its public key $PK_{\mathcal{PL}_j}$ (See Algorithm 1 and Fig. 6). After a pre-specified time, the smart contract transfers a partial amount of the balance as a down payment to the parking lot address. This can be done when the parking lot $\mathcal{PL}_j$ calls the *FineDriver* function after time t_e (See line 31 in Algorithm 1, and Fig 7). Note that this down payment is needed to confirm the driver's commitment to the parking reservation.

When $\mathcal{D}_i$ arrives to the parking lot, $\mathcal{PL}_j$ makes sure that this driver is the one who made the parking reservation by authenticating the driver. As shown in Fig. 6, the authentication is done as follows. First, $\mathcal{PL}_j$ sends a random number Γ as a challenge to $\mathcal{D}_i$ that uses the reservation secret key $SK_{\mathcal{D}_i}$ corresponding to the public key $PK_{\mathcal{D}_i}$ that was sent in the reservation request to generate an authentication message $(\Gamma, \sigma_{SK_{\mathcal{D}_i}}(\Gamma))$ and sends it to $\mathcal{PL}_j$. After that, $\mathcal{PL}_j$ verifies the signature using the public key $PK_{\mathcal{D}_i}$. Then, $\mathcal{PL}_j$ opens the parking lot gate and allows the driver to park.

After $\mathcal{D}_i$ finishes parking, a Schnorr multi-signature algorithm [41] is used by the parking lot and the driver to compute a signature and send it to the smart contract as a proof of finishing the parking to transfer the balance to the parking lot account. It is possible for the proof to be two signatures from the parking lot and the driver, but using the multi-signature is more efficient in terms of communication and verification time. To compute the multi-signature, $\mathcal{D}_i$ computes a signature $\sigma_{SK_{\mathcal{D}_i}}(TS)$ on a timestamp (TS) and sends it to $\mathcal{PL}_j$ that also signs the signature using its secret key ($SK_{\mathcal{PL}_j}$) as shown in Fig 6. Then, the multi-signature can be written as:

$$<\sigma_{SK_{\mathcal{PL}_j},\, SK_{\mathcal{D}_i}}(TS)> \quad (19)$$

Algorithm 1 Pseudocode of the Time-Locked Parking Payment Contract in Our System

```
1  contract TimeLockedPayment
2      uint Public Balance // balance to withhold driver
           deposit
3      Address Driver // driver address
4      Address PL // parking lot address
5      uint $PK_{\mathcal{D}_i}$ // driver reservation public key
6      uint $PK_{\mathcal{PL}_j}$ // parking lot public key
7      uint $t_s$ // time to start parking
8      uint $t_e$ // estimated time to end parking
9      uint $pr$ // parking rate/price
10     uint $multi$ // multi-signature that should be
           received to transfer the parking payment
       // Constructor
11     function TimeLockedPayment(_Driver, _PL, _Balance,
       _$PK_{\mathcal{D}_i}$, $PK_{\mathcal{PL}_j}$, _$t_s$, _$t_e$, _$pr$)
12         Driver ← _Driver; // The address of the
               driver.
13         PL ← _PL; // The address of the selected
               parking lot.
14         Balance ← _Balance; // This deposit as a
               proof to commit for the reservation.
15         $PK_{\mathcal{D}_i}$ ← _$PK_{\mathcal{D}_i}$; // public key to verify
               multi-signature
16         $PK_{\mathcal{PL}_j}$ ← _$PK_{\mathcal{PL}_j}$; // public key to verify
               multi-signature
17         $t_s$ ← _$t_s$ ;
18         $t_e$ ← _$t_e$ ;
19         $pr$ ← _$pr$ ;
20         return;
21     function Payment(_$multi$)
           // This function called by parking lot after
               constructing the multi-signature to
               transfer the remaining balance to parking
               lot account.
22         $multi$ ← _$multi$ ;
23         if ($multi$ is valid)
24         // Check if the submitted multisignture proof
               is valid.
25          transfer(Balance, PL);
26          // transfer the whole Balance to the PL.
27         return;
28     function FineDriver()
           // This function called by parking lot after
               pre-specified ($t_e$) period of time to
               transfer the down payment to the parking
               lot account as a fine to the driver.
29         if (msg.sender $\neq$ PL)  return;
30         if (currenttime >= $t_e$)  // check if the
               pre-specified time has expired.
31          transfer((0.15*Balance), PL);
32          // Transfer balance to the parking lot
               account.
33          Balance = 0.85*Balance ;
34         return;
```

Note that the driver's signature proves that the driver agrees to transfer the balance to the parking lot, and the parking lot's signature proves that the parking lot has got a permission from the driver to transfer the balance to its account. After constructing the multi-signature, $\mathcal{PL}_j$ calls the *payment* function to receive the parking payment. Then, the smart contract verifies the signature using the public keys ($PK_{\mathcal{D}_i}$, $PK_{\mathcal{PL}_j}$), which are stored in the smart contract. Then, the smart contract automatically transfers the remaining balance to the $\mathcal{PL}_j$ account (See line 25 in Algorithm 1).

Finally, the parking payment is finalized and $\mathcal{D}_i$ uses a decentralised sharing application running a smart contract [45] on his smartphone to obtain a barcode with a numerical identifier. Then, he/she drives towards the gate of

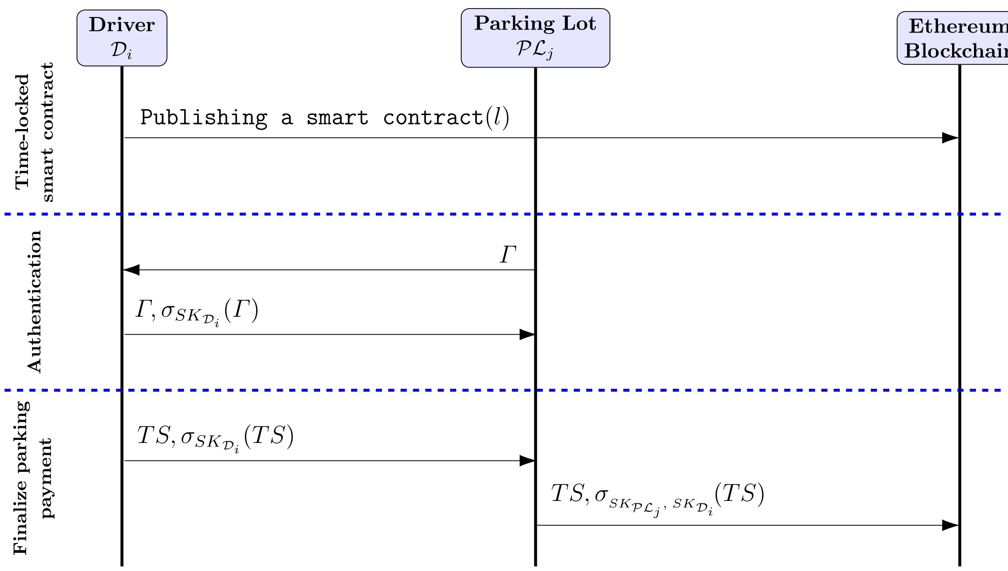
Driver $\mathcal{D}_i$
Parking Lot $\mathcal{PL}_j$
Ethereum Blockchain
Time-locked smart contract
Publishing a smart contract(l)
Authentication
Γ
$\Gamma, \sigma_{SK_{\mathcal{D}_i}}(\Gamma)$
Finalize parking payment
$TS, \sigma_{SK_{\mathcal{D}_i}}(TS)$
$TS, \sigma_{SK_{\mathcal{PL}_j}, SK_{\mathcal{D}_i}}(TS)$

FIGURE 6. An illustration for the *parking and payment* phase.

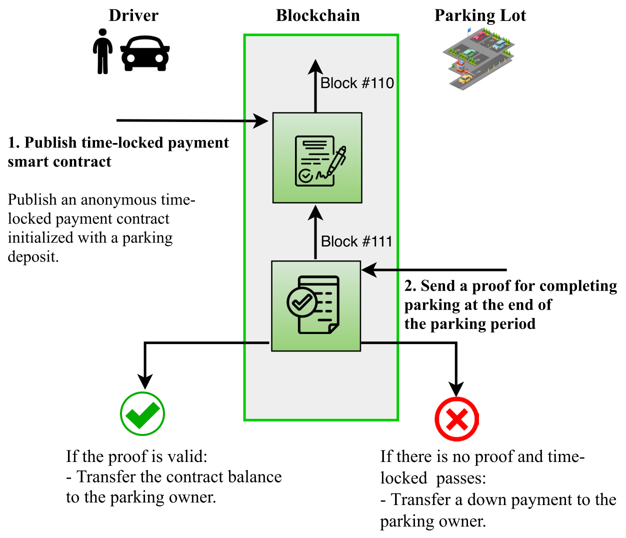
Driver
Blockchain
Parking Lot
Block #110
Block #111
1. Publish time-locked payment smart contract
Publish an anonymous time-locked payment contract initialized with a parking deposit.
2. Send a proof for completing parking at the end of the parking period
If the proof is valid:
- Transfer the contract balance to the parking owner.
If there is no proof and time-locked passes:
- Transfer a down payment to the parking owner.

FIGURE 7. An illustration for the time-locked payment.

the parking lot and scans the barcode to open the gate at the parking exit to leave the parking lot.

G. REPUTATION UPDATE

In the *reputation update* phase, the drivers anonymously send their feedback (rating scores) to the blockchain to evaluate the parking lots service. The drivers evaluate the parking service based on several measures, such as punctuality of reservation times, cleanliness of the parking area, and quality of provided services (e.g., charging for electrical vehicles and car wash), etc. These ratings are used to update the reputation scores of the parking lots, and these scores are used by drivers to select a good parking lot. We integrate the reputation management scheme proposed in [46] into our system. In this scheme, a number of validators, called aggregators, should run the scheme. These aggregators can be the parking lots with the highest reputation scores in each reputation update period.

The multi-signature computed by driver $\mathcal{D}_i$ and parking lot $\mathcal{PL}_j$ at the end of the parking can be used by $\mathcal{D}_i$ as a reputation token (T_i) to be able to submit a rating score for the parking lot. The token proves that there was a parking event at a specific time and the parking payment has been transferred to the parking lot. Then, to compose an anonymous rating transaction, $\mathcal{D}_i$ does the following. He/she chooses a rating score $rs_{i,j}$, where $rs_{i,j}$ is an integer in the range [0, 5]. Then, he/she encrypts the score using a public key computed from the aggregators' public keys: $PK = \prod_{v=1}^{V} PK_v$, where V is the number of aggregators running the reputation management scheme, PK_v is the public key of aggregator v, which is computed as: ${g_2}^{SK_v}$, and $SK_v \in Z_p$ is the secret key of aggregator v.

Note that $\mathcal{D}_i$ encrypts the rating score $rs_{i,j}$ and no one should know the score to avoid any negative consequences in case of low rating scores. For example, parking lots may prevent a driver from parking his/her vehicles in the future if he/she submits low rating scores.

For $\mathcal{D}_i$ to encrypt the rating score $rs_{i,j}$ using the public key PK, he/she selects a random number $z \in Z_p$ and computes:

$$\Lambda_{i,j} = (\Lambda_{i,j,1}, \Lambda_{i,j,2}) = (PK^z\, g_2^{rs_{i,j}}, g_2^z) \qquad (20)$$

Then, $\mathcal{D}_i$ sends an anonymous rating transaction to the blockchain. The transaction has the following: the encrypted rating score ($\Lambda_{i,j}$), the reputation token (T_i), the payment smart contract address (l_{add}), and his/her reservation public key $PK_{\mathcal{D}_i}$, and the public key of the parking lot ($PK_{\mathcal{PL}_j}$) that should receive the rating score.

$$Rating: < \Lambda_{i,j}, T_i, l_{add}, PK_{\mathcal{D}_i}, PK_{\mathcal{PL}_j} > \tag{21}$$

Note that the anonymous rating transaction is signed by the driver's reservation secret key $SK_{\mathcal{D}_i}$ to produce the signature $(\sigma_{SK_{\mathcal{D}_i}}(Rating))$. Also, the payment smart contract address is necessary to check if the balance has been transferred to $\mathcal{PL}_j$ account as a proof for completing the parking. After the aggregators receive the rating transaction from $\mathcal{D}_i$, they check the following: (i) the rating transaction signature and the multi-signature which is included in the reputation token T_i are valid, (ii) T_i is recorded in the time-locked payment smart contract, (iii) the time-locked payment has been transferred to the parking lot as a proof that the parking event has occurred, and (iv) T_i has not been used before by verifying that it is not recorded in the shared reputation ledger.

Then, the aggregators start the rating aggregation process at the end of the reputation update period. Note that the ratings of each parking lot received in a period of time are aggregated so that the parking lots cannot know the individual ratings of the drivers. To aggregate the encrypted ratings of each $\mathcal{PL}_j$, the aggregators calculate:

$$\begin{aligned} rs_j &= (rs_{j,1}, rs_{j,2}) \\ &= \left(\prod \Lambda_{i,j,1}, \prod \Lambda_{i,j,2}\right) \\ &= (PK^{\sum z}\, g_2^{\sum rs_{i,j}}, g_2^{\sum z}) \end{aligned} \tag{22}$$

for all encrypted ratings $rs_{i,j}$. After that, each aggregator computes a partial decryption token, $p_{j,v} = rs_{j,2}^{SK_v}$ and broadcasts it to the rest of the aggregators.

1) BLOCKS GENERATION

At the end of each reputation update period, the Raft consensus algorithm is run by the aggregators to elect a leader, which will be responsible for updating the reputation ledger. Then, the leader computes the aggregated rating of a parking lot $\mathcal{PL}_j$ as follows:

$$\begin{aligned} \frac{rs_{j,1}}{\prod\limits_{v=1}^{n} p_{j,v}} &= \frac{g_2^{\sum z \,\cdot\, \sum_{v=1}^{n} SK_v} \cdot g_2^{\sum rs_{i,j}}}{g_2^{\sum z \,\cdot\, \sum_{v=1}^{n} SK_v}} \\ &= g_2^{\sum rs_{i,j}} \end{aligned} \tag{23}$$

Given $g_2^{\sum rs_{i,j}}$, the aggregated rating $\sum rs_{i,j}$ is computed by taking the discrete logarithm of $g_2^{\sum rs_{i,j}}$. Note that this calculation is not difficult because $\sum rs_{i,j}$ is not a large number. There are different ways to do this computation, but the easy way is by creating a lookup table. Then, the aggregated rating is used to update the reputation score of $\mathcal{PL}_j$ ($\mathcal{RS}_{\mathcal{PL}_j}$) stored in the reputation ledger.

For the leader to propose a new block, it has to create a list containing all the valid reputation tokens received in the current reputation update period and a list containing the updated reputation scores of all parking lots. Note that the structure of the reputation ledger is similar to that of Fig. 4 except that the block body contains reputation tokens and reputation scores instead of parking offers. Then, the leader broadcasts a proposed block to the followers and waits for acknowledgements from the majority of the followers. After that, the leader signs the block and broadcasts it to the blockchain validators to update their local copies of the reputation ledger.

H. DRIVERS ANONYMITY REVOCATION

In our system, the drivers are anonymous to preserve their privacy. However, some drivers may abuse this anonymity and deliberately launch attacks in order to disrupt the proper operation of the parking system. Therefore, the proposed system should enable *anonymity-yet-accountability* to identify the malicious drivers and revoke them from the system. As an example for one misbehavior, we assume that any driver exceeds a certain threshold of uncommitted reservations ($\mathcal{T}_f$) should be tracked and revoked from the system. This is done as follows:

- If a driver $\mathcal{D}_i$ makes a reservation with the parking lot $\mathcal{PL}_j$ and publishes the time-locked payment smart contract and does not commit to that reservation, $\mathcal{PL}_j$ should report this action to the KDC. Specifically, $\mathcal{PL}_j$ sends the reservation request, which includes the driver's anonymous credential $(\sigma_{\mathcal{D}_i}^{[1]`}, \sigma_{\mathcal{D}_i}^{[2]`})$ and the smart contract address (l_{add}).
- The KDC verifies that only a partial amount of the time-locked payment (down payment) has been transferred to $\mathcal{PL}_j$ address because $\mathcal{D}_i$ made a reservation but he did not commit to it. To do so, the KDC can check whether the smart contract has received the multi-signature $(\sigma_{SK_{\mathcal{PL}_j},\, \mathcal{SK}_{\mathcal{D}_i}}(TS))$ as a proof of the parking event. If the multi-signature is not stored in the smart contract and $\mathcal{PL}_j$ has received the down payment only, then $\mathcal{D}_i$ did not commit to the reservation.
- The KDC traces the real identity of the driver by checking:

$$e(\sigma_{\mathcal{D}_i}^{[2]`}, g_2) \cdot e(\sigma_{\mathcal{D}_i}^{[1]`}, \tilde{X})^{-1} \overset{?}{=} e(\sigma_{\mathcal{D}_i}^{[1]`}, \tilde{\gamma}) \tag{24}$$

  for all entries $\{ID_{\mathcal{D}_i}, \gamma, \eta, \tilde{\gamma}\}$ until a match is found.
- Then, the KDC records the driver's data $\{ID_{\mathcal{D}_i}, \gamma, \eta, \tilde{\gamma}\}$ in the suspected drivers table.
- When the KDC receives $\mathcal{T}_f$ uncommitted reservations for the same driver, it permanently revokes the driver by publishing $\{ID_{\mathcal{D}_i}, \gamma, \eta, \tilde{\gamma}\}$ on the blockchain.
- Finally, all parking lots can identify the anonymous reservations of this driver using Eq. 24, and thus discard his request.

V. EVALUATIONS

A. SECURITY AND PRIVACY ANALYSIS

We follow the same methodology used in [47]–[50] to analyze the security and privacy of our system. We also follow the widely used approach for analyzing security of networks and systems [51], [52]. Our system uses a combination of techniques such as PIR, short randomizable signature, commitment technique, and other cryptosystems as explained in section IV. We assume that these cryptosystems are secure and their security proofs are detailed in their papers. Using this assumption, we demonstrate in this subsection how the combination of these techniques can achieve the security and privacy goals discussed in section III-C.

1) DECENTRALIZATION

The proposed system can manage the parking service without reliance on a centralized or trusted third party. This is achieved by using the blockchain which is responsible for publicizing the parking offers, and managing reputation scores and payment. This makes the system robust against the single point of failure and attack that the current centralized systems suffer from.

2) PRESERVING DRIVERS' PRIVACY

The drivers' privacy in terms of the driver anonymity and untraceability is preserved for the following reasons:

1) The diver's real identity is only used in the registration during the *system initialization* phase and after that neither the real identity nor its derivatives are used in any message transmitted from the driver. This way, no one except the KDC can know the driver's real identity.
2) In the *parking offers retrieval* phase, the real identity of a driver is replaced with temporary public/private key pair that is used only once so that no one can link the requests sent from the same driver at different occasions. Moreover, the drivers' daily parking activities are preserved from blockchain validators, parking lots, and external adversaries by using information-theoretic PIR technique, so that no one with background information can trace drivers. The drivers can privately retrieve parking offers without revealing their desired parking location. For a driver $\mathcal{D}_i$ to retrieve the offers in the desired cell $C^{[d]}$, he chooses i.i.d. and uniformly distributed codewords from a query code $\mathcal{C}_q$, which is an $[n, t]$ Reed-Solomon code. This randomness confuses the blockchain validators and hides the identifier of the cell of interest from any t colluding validators because $\mathcal{C}_q$ is an $[n, t]$ MDS code, and hence, the distribution of any t queries is uniform and independent from d in the same manner of Shamir's secret sharing [53]. Also, one of the advantages of the PIR technique is that it encourages validators (parking lots) to run the system and respond to the drivers' queries because they do not know whether the drivers query the parking offers of their parking lots or not.
3) In the *reservation* phase, by using the short randomizable signature, only legitimate drivers are able to make reservations without revealing their real identities. Moreover, the parking reservation request from a driver $\mathcal{D}_i$ is $(Enc_{PK_{\mathcal{PL}_j}}(PK_{\mathcal{D}_i}, t_p, TS), (\sigma_{\mathcal{D}_i}^{[1]})^{r_1}, (\sigma_{\mathcal{D}_i}^{[2]})^{r_1}, c_{\mathcal{D}_i}, s)$. Only $(PK_{\mathcal{D}_i}, (\sigma_{\mathcal{D}_i}^{[1]})^{r_1}, (\sigma_{\mathcal{D}_i}^{[2]})^{r_1}, c_{\mathcal{D}_i}, s)$ are driver's specific parameters that can be used to link the driver's requests. However, $PK_{\mathcal{D}_i}$ is a temporary public key that is used only once and the driver uses different values for r_1 in randomizing the signature $((\sigma_{\mathcal{D}_i}^{[1]})^{r_1}, (\sigma_{\mathcal{D}_i}^{[2]})^{r_1})$ for each reservation request. These randomized signatures are unlinkable because linking $((\sigma_{\mathcal{D}_i}^{[1]}), (\sigma_{\mathcal{D}_i}^{[2]}))$ to $((\sigma_{\mathcal{D}_i}^{[1]})^{r_1}, (\sigma_{\mathcal{D}_i}^{[2]})^{r_1})$ for some $r_1 \in Z_p$ is equivalent to solving the decisional Diffie–Hellman (DDH) problem in G_1, which is known to be a hard problem [54]. Also, $c_{\mathcal{D}_i}$ and s are derived parameters from $((\sigma_{\mathcal{D}_i}^{[1]})^{r_1}, (\sigma_{\mathcal{D}_i}^{[2]})^{r_1})$ and cannot be used to link the requests. This means that all the fields of the reservation request are either dynamic parameters or derived from random values. Thus, if a driver sends several reservation requests to the same parking lot at different occasions, it cannot learn whether these requests are sent from the same driver or not, because the requests have no invariant field that can be used to link the requests.
4) In the *payment* phase, our system does not depend on traditional centralized payment schemes, e.g., credit/debit cards that can breach the driver's privacy because the payment is executed through a third party (bank) that knows the driver's real identity. Thus, the bank can trace the driver's locations through the payments it makes to the different parking lots. Instead, drivers pay anonymously for the parking service using smart contracts and Ethereum public blockchain. In particular, the real identity of the driver is replaced with an Ethereum address, and he uses a different address for each payment to avoid linking his payments at different occasions and thus linking his parking reservations at different parking lots.
5) In the *reputation management* phase, a driver uses a different public/secret key pair each time he submits a rating so that ratings sent from the same driver at different occasions cannot be linked through these key pairs. Also, the privacy of a driver's rating is preserved by encrypting it with the aggregators' public keys, i.e., no one can learn the individual ratings but the aggregated rating of each parking lot can be computed and used to update its reputation score stored in the reputation ledger. For an adversary to know the rating score of a certain driver $\mathcal{D}_i$ from his encrypted score, $(PK^z\, g_2^{rs_{i,j}}, g_2^z)$, he has either to solve the DDH problem to get z and $rs_{i,j}$, which is known to be a hard problem or to collude with all the aggregators to calculate $rs_{i,j}$ as follows:

$$\frac{\Lambda_{i,j,1}}{\prod_{v=1}^{n}(\Lambda_{i,j,2})^{SK_v}} = \frac{PK^z\, g_2^{rs_{i,j}}}{g_2^{z\cdot\sum_{v=1}^{n} SK_v}} = \frac{g_2^{z\cdot\sum_{v=1}^{n} SK_v}\cdot g_2^{rs_{i,j}}}{g_2^{z\cdot\sum_{v=1}^{n} SK_v}} = g_2^{rs_{i,j}}, \tag{25}$$

which is not possible, given the security of the blockchain [46].

3) DRIVERS' ANONYMOUS CREDENTIALS UNFORGEABILITY

The security of the anonymous authentication used in our system is based on the existential unforgeability under chosen message attacks of the short randomizable signature of [39], i.e., the inability of falsely replicating another person's signature on a message that is not signed by him, even with having unlimited access to the signing oracle. This was proven under Assumption 2 of [39], similar to the Lysyanskaya Rivest Sahai Wolf (LRSW) assumption in [39]. Simply, for an adversary to forge the signature of a driver $\mathcal{D}_i$ on a message m^* given the short randomizable signature, $(\sigma_{\mathcal{D}_i}^{[1]`}, \sigma_{\mathcal{D}_i}^{[2]`}, c_{\mathcal{D}_i}, s)$, on a message m, he has to compute the random number a_2 selected by the driver in the *system initialization* phase, which is not possible under the mentioned assumptions.

4) SECURE PARKING MANAGEMENT

By using the time-locked payment, our system discourages making multiple reservations without commitment to them. In particular, to enable a driver to make a parking reservation, he has to make a time-locked deposit to prove his good will to the parking lot, and in case that he does not commit to his reservation, a down payment is transferred to the parking lot. Moreover, our system is secure against malicious drivers, who do not make a reservation to park in a parking lot, but try to hijack a reservation made by other driver. This is because when a driver arrives to a parking lot, he has to authenticate to the parking lot to be able to park. In particular, the parking lot sends a random number to the driver and he has to sign this number with the secret key of the public key sent during the reservation.

5) ANONYMITY-YET-ACCOUNTABILITY

As explained earlier, our system achieves driver anonymity. However, we want our system to identify and penalize misbehaving drivers. To achieve this balance, only the KDC is able to revoke the anonymity and identify malicious drivers in case of misbehavior from the signature $(\sigma_{\mathcal{D}_i}^{[1]}, \sigma_{\mathcal{D}_i}^{[2]})$ because $\tilde{\gamma}$ is only known to the KDC as described earlier. After that the KDC publishes the malicious driver's information on the blockchain, so that the parking lots can identify his reservations and discard him. This accountability property encourages parking lots to join the system without worrying about misbehaviors from malicious drivers. Also, driver's revocation property discourages drivers from misbehaving.

6) FAIR PARKING RATES

Since some parking lots may manipulate (increase/decrease) the rates of their parking offers if they know the other competitors' rates to achieve financial gains unfairly, our system ensures fair parking rates by using commitment technique during parking offers submission. In this technique, a parking lot cannot learn the parking rates of other competitors before they decide their rates. Also, since the hash functions used in the commitment technique are collision free, parking lots cannot change the committed offers during opening the commitments.

7) SECURE AND PRIVACY-PRESERVING REPUTATION MANAGEMENT

By linking the reputation management scheme with the payment scheme, a driver is only able to make a rating for a parking lot if there is an actual parking event and the payment is finalized. A driver needs to obtain a reputation token from the parking lot to submit a rating, and the parking lot needs to submit this token to the blockchain to get the payment. Therefore, the parking lot is compelled to send the reputation token to the driver in order to get paid. Also, the token is signed by both the driver and the parking lot so that the driver cannot use it to rate other parking lots. Moreover, for a driver to generate a rating transaction, he needs to sign the transaction with the secret key $(SK_{\mathcal{D}_i})$ used to sign the reputation token to prove that he is the owner of the token, which prevents an adversary to send a rating transaction on behalf of the token's owner. In addition, the rating transaction contains the reputation token to enable the aggregators to check if this token has been used before or not. This prevents a malicious driver from using the same token to make multiple rating transactions. Regarding privacy preservation, each driver encrypts his rating before sending it to the aggregators, and it is infeasible for any adversary to know the individual ratings of drivers. However, the aggregators aggregate all the encrypted ratings of each parking lot at the end of each reputation update period and compute the aggregated rating.

8) TRANSPARENCY

By using blockchain technology, no single entity or authority can monopolize the system for its own benefit. The rules of validating parking offers transactions are predefined and executed in a completely distributed manner. A greedy blockchain validator (parking lot) cannot omit a valid parking offer received from a competitive parking owner because the parking offer transaction is broadcasted to the blockchain network and verified by the majority of the validators, given the assumption that the majority of the validators are honest. Thus, the valid parking offers of all parking lots are recorded in a shared ledger, where the content of the ledger is agreed on by the blockchain validators through the Raft consensus algorithm and cannot be changed once recorded in the ledger. Moreover, the reputation scores are recorded on the shared ledger and they are publicly verifiable so a parking lot cannot cheat about its reputation score because its offer transaction will not be accepted by the blockchain network unless the reputation score included in the offer matches the one recorded in the reputation ledger. In addition, the process of updating the reputation scores is transparent and its correctness can be verified by the blockchain network given the rating transactions of each parking lot and the partial decryption keys of the aggregators.

TABLE 2. Size of the data in the parking offers.

Data	Size (bytes)	Data	Size (bytes)
Real identity (ID_j)	6	Available services index (S)	2
Certificate ($Cert_j$)	72	Price (pr)	2
Number of available parking slots (N)	2	Reputation score ($\mathcal{RS}_{\mathcal{PL}_j}$)	20
Cell identifier ($C^{[m]}$)	2	Time availability vector (t_a)	8
Location coordinates (loc)	6		

B. PERFORMANCE EVALUATION

In this subsection, we evaluate the performance of our system in terms of communication, computation and storage overheads.

1) COMMUNICATION OVERHEAD

The communication overhead is measured by the size of transmitted messages in *bytes*. We will measure the size of the following messages.

1) The message between a parking lot and the blockchain in the *parking offers submission* phase.
2) The message between a driver and the blockchain in the *parking offers retrieval* phase.
3) The message between a driver and a parking lot in the *parking reservation* phase.
4) The message between a driver and the blockchain in the *reputation update* phase.

In the *parking offers submission* phase, the items of the parking offer sent from each parking lot to the blockchain validatros and their sizes are given in Table 2. Thus, the total size of a parking offer is equal to 120 bytes. In the *parking offers retrieval* phase, the size of the total downloaded data is as follows.

$$\text{Size of downloaded data} = \frac{\text{Size of offers in the cell}}{\mathrm{R}_{PIR}} \tag{26}$$

Note that the retrieval rate, R_{PIR} is given by Eq. (10), so Eq. (26) can be rewritten as follows.

$$\text{Size of downloaded data} = \frac{\text{Size of offers in the cell}(n-r)}{n-t-2b-r}, \tag{27}$$

where, we consider that $n = 44$, and $t + b + r = 3$.

Fig. 8, shows the communication overhead in the *parking offers retrieval* phase in our system compared with the anonymous smart-parking and payment scheme (ASAP) in [21] and the naive solution (i.e., downloading all the parking offers of all the city). The figure shows that downloading the parking offers using the naive solution exhibits high communication overhead compared to the PIR technique. For instance, according to Eq. 27, the communication overhead in the *parking offers retrieval* phase in our system is about 1.161 *kbytes* assuming that the number of offers in each cell is 9 and there are 50 cells. However, the communication overhead to download all the parking offers in the naive solution is more than 50 *kbytes*. This proves the efficiency of the PIR and its ability to preserve the driver privacy with low communication overhead. Moreover, the naive solution incurs much communication overhead as the number of cells and offers in each cell increase.

We compare our PIR-based system with the ASAP scheme since the drivers in the ASAP need to retrieve parking offers in a certain cloaked area from a server which is similar to our case. Fig. 8 indicates that our system has less communication overhead compared to ASAP. For example, in our system, about 1.161 *kbytes* are needed to download 9 offers using PIR, while more than 20 *kbytes* are needed to download the same number of offers in ASAP. This is because the PIR technique is efficient and ASAP uses an encryption scheme that exhibits large communication overhead.

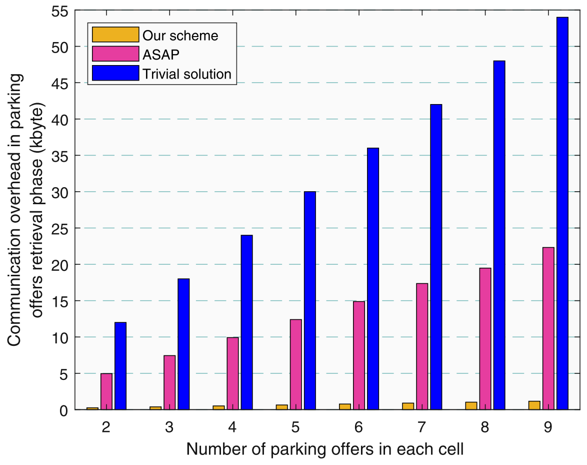

FIGURE 8. Communication overhead in *parking offers retrieval* phase versus the number of retrieved parking offers when the number of blockchain validators is 44 and the number of cells is 50.

In addition, we have measured the communication overhead in the *parking offers retrieval* phase with different number of blockchain validators and fixed number of offers. As can be seen in Fig. 9, when the number of blockchain validators increases at fixed number of offers (75 offers), the size of the total downloaded data decreases, i.e., the data retrieval rate (R_{PIR}) increases. This is because the effect of Byzantine validators is reduced when there is a fixed number of Byzantine blockchain validators ($b = 1$). For example, the communication overhead to retrieve the data of 75 offers from 9 validators (7 honest, 1 byzantine, and 1 unresponsive) is less than 15 *kbytes*. However, the communication overhead to download the same data from 14 validators (12 honest, 1 byzantine, and 1 unresponsive) is less than 12 *kbytes*.

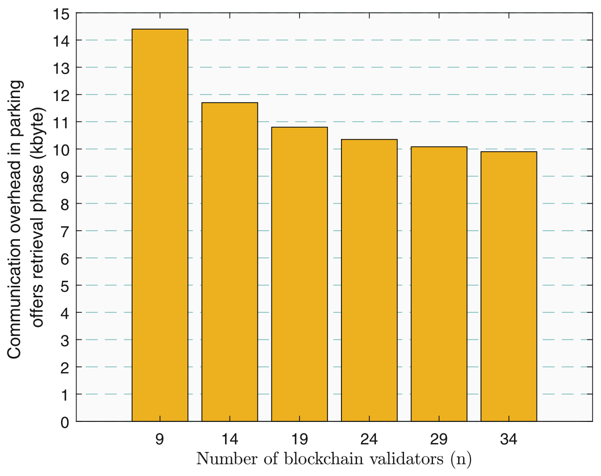

FIGURE 9. Communication overhead in *parking offers retrieval* phase versus the number of blockchain validators when the number of parking offers is 75.

In the *parking reservation* phase, the driver sends a reservation request containing a ciphertext $C^R_{\mathcal{D}}$ and a short randomizable signature $(\sigma^{[1]'}_{\mathcal{D}}, \sigma^{[2]'}_{\mathcal{D}}, c_{\mathcal{D}}, s)$. Thus, the size of the reservation request is $2 \times 20 + 4 \times 20 + 2 \times 32 = 184$ bytes.

In the *reputation update* phase, the driver composes a rating packet that contains 8 bytes for ciphertext $\Lambda_{i,j}$, 20 bytes for reputation token T_i, 32 bytes for smart contract address l_{add}, 20 bytes for the driver reservation public key $\mathcal{PK}_{\mathcal{D}_i}$, and 20 bytes for the parking lot public key $PK_{\mathcal{PL}_j}$. The total size of the rating packet is: $8 + 20 + 32 + 2 \times 20 = 100$ bytes. The communication overhead in bytes of the different messages exchanged between the system entities are summarized in Table 3.

TABLE 3. Communication overhead.

Message	Involved entity	Communication overhead (bytes)
Parking offer	Parking lot	120
Downloading the parking offers	Driver	1611
Parking reservation request	Driver	184
Anonymous rating packet	Driver	100

2) COMPUTATION OVERHEAD

The computation overhead is defined as the time required to execute system operations or compose messages including the computation of cryptographic operations of our system, including bilinear pairing (Pairing), hashing (Hash), addition (Add), multiplication (Mul), and exponentiation (Exp).

We implemented the required cryptographic operations using Python charm cryptographic library [55] running on Raspberry Pi 3 devices with 1.2 GHz Processor and 1 GB RAM. We used supersingular elliptic curve with the asymmetric Type 3 pairing of size 160 bits (MNT159 curve) for bilinear pairing, and $SHA-2$ hash function. Table 4 gives the computation time of the cryptographic operations used in our system.

TABLE 4. Computation times of the cryptographic operations used in our system.

Cryptographic operation	Time (ms)
Pairing	3.138600
Hash	0.058359
Add	0.000227
Mul	0.000269
Exp	0.333714

In the *parking offers submission* phase, the parking lot has to compute a commitment and a signature to submit a parking offer. The commitment requires 1 Hash and the signature requires 4 Mul and 1 Add. Therefore, the total computation time of the parking offer is 0.059662 ms.

In the *parking offers retrieval* phase, the time needed by the driver to retrieve parking offers in the desired cell from blockchain validators can be given by Eq. (28).

$$\text{Retreival Time} = \frac{1}{R_{PIR} * \text{Channel Rate}} \tag{28}$$

Considering that the number of validators is 44 and the communication channel rate is 10 Mbps which is the rate of LTE cellular networks [56] assuming the drivers use cellular network to connect to the blockchain. Then, the time needed to retrieve parking offers from the blockchain validators is 0.107μs.

To compute a parking reservation request in the *parking reservation* phase, the driver has to compute 1 Enc which requires 2 Mul, and 1 Add, in addition to a short randomizable signature that requires 3 Exp, 1 Mul, 1 Add and 1 Hash. Therefore, the total computation time of parking reservation request is 1.003 ms.

To compute an anonymous rating transaction in the *reputation update* phase, the driver has to compute 1 Enc which requires 3 Exp and 1 Mul, and a signature which requires 1 Hash, 4 Mul and 1 Add. Therefore, the total computation time to compose a rating transaction is 1.06 ms. Also, the validators of the blockchain network verify the received rating transaction by verifying the signature. To do so, each validator has to compute 1 Hash, 4 Mul and 1 Add, which take 0.0597 ms. Fig. 10 shows the computation overhead of a blockchain validator in verifying different number of rating transactions. Fig. 10 shows that the computation overhead increases linearly with the number of rating transactions. However, with a large number of rating transactions such as 100, the computation burden on each validator is still low because it takes only 6 ms from the validator to verify these 100 transactions. Overall, it can be concluded that verifying the rating transactions is efficient which makes our system scalable, i.e., can serve many drivers and parking lots. The computation overhead in (ms) of the different operations in our system along with the involved entities are summarized in Table 5.

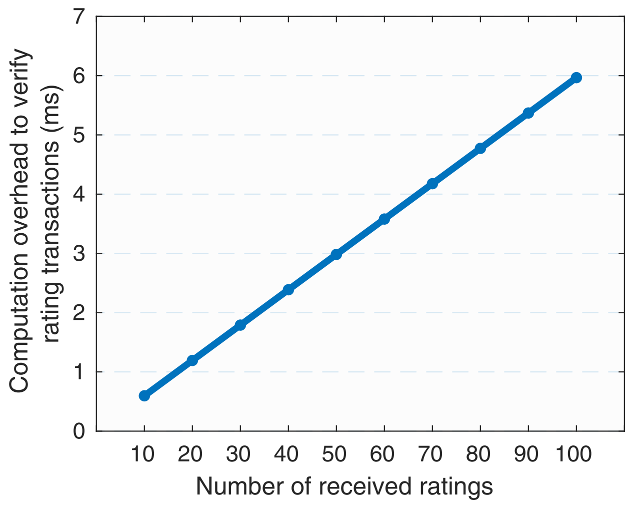

FIGURE 10. Computation overhead on blockchain validators in verifying anonymous raring transactions.

TABLE 5. Computation overhead.

Operation	Involved entity	Computation overhead (ms)
Submitting a parking offer	Parking lot	0.059662
Parking offers retrieval	Driver	0.000107
Parking reservation request	Driver	1.003
Anonymous rating generation	Driver	1.06
Anonymous rating verification	Validator	0.0597

3) ON-CHAIN STORAGE OVERHEAD

The on-chain storage overhead is defined as the storage space in bytes required at each blockchain validator to maintain the ledger. As calculated earlier, the size of the parking offer is 120 bytes. We consider that each cell has 50 offers, the number of cells is 40, the size of block header and trailer is 80 bytes and blocks are generated frequently every 15 minutes. Then, the size of the ledger after a complete year would be $(120\times50\times40)\times4\times24\times365 = 8.4$ GB, which is not a large space with respect to the storage resources of parking lots. Moreover, the parking lots can free up their storage on annual basis to reduce the storage overhead. Freeing up previous parking offers is not problematic since these offers are already expired and no longer needed.

VI. RELATED WORK

Smart parking systems have gained great interest recently. Different companies have established smart parking systems around the world to facilitate finding vacant parking slots [57]. For instance, ParkMobile [9] operates more than 30,000 parking lots in 400 different cities. Also, ParkWhiz [10] operates in over 200 cities and provides availability information about more than 800,000 parking slots. Moreover, SFpark [11] monitors 8,200 parking slots in 20 cities in the United States to provide online parking reservation service. However, these systems do not consider privacy, and therefore, several research works have been carried out to address privacy such as [18]–[21].

Lu *et al.* [18] have proposed a privacy-preserving parking management scheme for large parking lots. In this scheme, road side units (RSUs) are deployed at the parking lot to collect data from sensors installed at parking slots. This data is used to generate real-time parking information, including a map for the vacant parking slots in the parking lot. Specifically, when a vehicle arrives to the parking lot, it queries the RSUs in the lot for an available parking slot. Then, the RSUs guide the vehicle to the nearest parking slot. The scheme mainly focuses on preserving the privacy of drivers by hiding their real identities during the communication with the RSUs. However, online parking reservation service is not provided, where drivers need to drive to the parking lot to know if a parking space is available or not. In case there is no available parking slot in a parking lot, drivers have to drive to another parking lot to look for a vacant parking slot, which causes traffic congestions and wasting time and gas to find a parking lot.

Ni *et al.* [19] have proposed a privacy-preserving parking management scheme that enables a cloud server to guide vehicles to vacant parking slots using real-time parking information. The idea is that a driver sends an encrypted query that includes current location, destination, and current/arrival times to the nearest RSU. Then, the RSU forwards the query to the cloud server that decrypts the query and sends the available parking lot information (encrypted) to the RSUs the driver could pass by them on his way to the destination. Finally, the driver retrieves this information from an RSU and contacts the parking lot. However, the main focus of this scheme is on preserving the privacy of drivers from the RSUs and it assumes the cloud server is trusted to know the locations and identities of the drivers.

Huang *et al.* [20] have presented a privacy-preserving automated valet parking reservation scheme for autonomous vehicles (AVs). The idea is that after a passenger reaches her destination, the AV drops her and then drives to a parking slot. The scheme allows passengers of AVs to find the nearest parking lot in real time while preserving their location privacy. The scheme hides the passenger's identity using one-time and unlinkable pseudonyms. The scheme also preserves the passenger's location privacy using a location obfuscation mechanism, in which the exact location of the AV is generalized into a larger area. A parking service provider provides the passenger with a list of parking lots in the area of interest. Then, the passenger sends a reservation request to the service provider, which forwards the request to the selected parking lot. In double-reservation attacks, a malicious passenger reserves multiple parking slots without parking, which may cause financial loss to parking lots. To prevent these attacks, the passenger proves that she has parked to obtain a new pseudonym to be used in a new reservation request.

In [21], a privacy-preserving parking management system is proposed. The system anonymizes drivers and parking lots

TABLE 6. Comparison between our system and existing smart parking systems.

	Decentralization	Location privacy	Anonymous payment	Accountability	Fair parking rates	Reputation	Transparency
ParkMe [8]	×	×	×	×	×	×	×
SAVP [20]	×	⊕	○	×	×	×	×
ASAP [21]	×	⊕	○	×	×	×	×
ParkGene [58]	√	×	√	×	×	×	√
Our system	√	√	√	√	√	√	√

Note: √ a realized feature, × an unrealized feature, ○ a realized feature with the need to trusted party, and ⊕ leak coarse location information.

using group signature. Also, the location privacy is maintained by using a cloaking technique, where the parking location is blurred into large geographic area, called cell. In this scheme, the parking lot sends parking offers to the service provider and a driver sends a query to the service provider to retrieve parking offers in a certain cell. To achieve efficient and fast matching between the drivers' queries and the parking owners' offers, the service provider utilizes a HashMap technique in which all parking slots are stored in a hash tree. Moreover, drivers use anonymous digital coupons, that are issued by a third-party server, for the payment of parking fees.

However, the location obfuscation and cloaking techniques used in [20], [21] degrades the accuracy of selecting the nearest parking lots. Specifically, a passenger/driver cannot determine whether the selected parking lot is the closest to her desired destination or not. So, she might contact several parking lots in the obfuscated area to select a parking close to her destination. Also, coarse location information (cloaked area of the passenger/driver) are leaked to the service provider. If the cloaked area has only a few parking lots, information on the activities of the drivers can be revealed. The proposed schemes in [20] and [21] also depend on a central server to manage the service.

Furthermore, [58] is a blockchain-based smart parking system, in which the parking lots are public lots with large number of parking slots. However, [58] does not consider the drivers' location privacy, anonymous payment, accountability, fair parking rates, and the reputation of the parking lots. Different from the existing systems, the proposed system uses the blockchain to manage the parking service for the public parking lots, while ensuring the drivers' location privacy, anonymous payment, accountability, and fair parking rates, and considering the reputation of parking lots at the same time. In Table 6, we compare our system to the existing smart parking systems in terms of architecture and desirable features. The smart parking systems in [8], [20] and [21] rely on a central server to manage the parking service. Thus, they are vulnerable to single point of failure and attack, and suffer from lack of transparency. Moreover, they do not address accountability, fair parking rates, and the reputation of parking lots. The smart parking system in [8] does not consider privacy preservation and anonymous payment. The research works in [20] and [21] can hide the exact location of drivers but they leak coarse location information that may not be effective if there are a few parking lots in the coarse location area. Also, drivers use anonymous payment system for parking payment. However, the payment method relies on a central trusted authority (e.g., bank, ïˇAnancial institution, etc.), which is vulnerable to single point of failure. To the best of our knowledge, none of the relevant systems consider fair parking rates and reputation of the parking lots.

VII. CONCLUSION AND FUTURE WORK

In this article, we have proposed a privacy-preserving blockchain-based smart parking system. We have used commitment technique to ensure fair parking rates and prevent competing parking lots from manipulating their rates to achieve financial gains unfairly. To preserve the drivers' location privacy, we have used private information retrieval technique that allows drivers to retrieve parking offers from the blockchain validators without revealing any information about the parking locations of interest. We have leveraged the short randomizable signature to anonymously and efficiently authenticate the drivers during the online parking reservations. To preserve the privacy of drivers during payment, we have presented a time-locked anonymous payment system by leveraging smart contract and blockchain. Finally, our system can manage reputation anonymously using blockchain where drivers can rate the parking service without being tracked or retaliated. Our evaluations have shown that using the blockchain for publicizing parking offers, and managing payment and reputation can ensure that the system is secure against single point of failure and attack, and transparent. Moreover, our evaluations have demonstrated that the proposed system can ensure fair parking rates and preserve the drivers' privacy with low communication, computation and storage overheads.

Consequently, this article provides robust and efficient smart parking solution that allows the drivers to use the parking service without worrying about their privacy. This will increase the trust in the smart parking systems by the drivers and the dependence on them. As a result, this will mitigate the traffic congestion and the air pollution negatively affecting our communities. With the emerging of intelligent transportation system, modern vehicles are equipped with internet access capabilities and self-parking functions, and there are also self-driven cars. The smart parking system proposed in this article can be perfectly applied to all types of vehicles. The internet access can facilitate the communication between the vehicle and the parking system.

Moreover, once a car reaches its reserved parking slot, the self-parking functions can be activated to park the car.

In the future, we expect increasing demand on smart parking systems, so we will target a hybrid smart parking system that allows both public and private parking owners to participate in the system. Private parking owners such as home inhabitants can share their parking slots in case they are not used. This has the potential to increase the number of slots available for parking. However, in this case the privacy of the private parking owners should also be taken into account, and the parking system needs to be updated to address this challenge.

ACKNOWLEDGMENT

The authors, therefore, gratefully acknowledge the DSR technical and financial support.

REFERENCES

[1] *South China Morning Post*. Accessed: Jan. 5, 2020. [Online]. Available: https://www.scmp.com/
[2] *Drivers Spend an Average of 17 Hours a Year Searching for Parking Spots*. Accessed: Jan. 5, 2020. [Online]. Available: https://www.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/
[3] H. Li, K. Ota, and M. Dong, "Network virtualization optimization in software defined vehicular ad-hoc networks," in *Proc. IEEE 84th Veh. Technol. Conf. (VTC-Fall)*, Montreal, QC, Canada, Sep. 2016, pp. 1–5.
[4] A. A. Mohamed, M. Shawky, H. M. Abdel-Latif, and M. S. Sabry, "Enhancement of parking management system in Cairo using smartphones," *J. Eng. Adv. Technol.*, vol. 8, no. 6, pp. 3292–3300, Aug. 2019.
[5] D. C. Shoup, "Cruising for parking," *Transp. Policy*, vol. 13, no. 6, pp. 479–486, Nov. 2006.
[6] T. Giuffrè, S. M. Siniscalchi, and G. Tesoriere, "A novel architecture of parking management for smart cities," *Procedia-Social Behav. Sci.*, vol. 53, pp. 16–28, Oct. 2012.
[7] *Sources of Greenhouse Gas Emissions*, Dept. U.S. Environ., Washington, DC, USA, 2016.
[8] *Parkme*. Accessed: Jan. 5, 2020. [Online]. Available: https://www.parkme.com/
[9] *Parkmobile*. Accessed: Jan. 5, 2020. [Online]. Available: https://www.parkmobile.io/
[10] *Parkwhiz*. Accessed: Jan. 5, 2020. [Online]. Available: https://www.parkwhiz.com/
[11] *Fybrsfpark*. Accessed: Jan. 5, 2020. [Online]. Available: https://www.fybr.com/
[12] *Spothero*. Accessed: Jan. 5, 2020. [Online]. Available: https://spothero.com/
[13] K. Yang, K. Zhang, J. Ren, and X. Shen, "Security and privacy in mobile crowdsourcing networks: Challenges and opportunities," *IEEE Commun. Mag.*, vol. 53, no. 8, pp. 75–81, Aug. 2015.
[14] X. Pan, J. Xu, and X. Meng, "Protecting location privacy against location-dependent attacks in mobile services," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 8, pp. 1506–1519, Aug. 2012.
[15] H. Jiang, P. Zhao, and C. Wang, "RobLoP: Towards robust privacy preserving against location dependent attacks in continuous LBS queries," *IEEE/ACM Trans. Netw.*, vol. 26, no. 2, pp. 1018–1032, Apr. 2018.
[16] K. Rabieh, M. M. E. A. Mahmoud, and M. Younis, "Privacy-preserving route reporting schemes for traffic management systems," *IEEE Trans. Veh. Technol.*, vol. 66, no. 3, pp. 2703–2713, Mar. 2017.
[17] M. Nabil, A. Sherif, M. Mahmoud, A. Alsharif, and M. Abdallah, "Efficient and privacy-preserving ridesharing organization for transferable and non-transferable services," *IEEE Trans. Dependable Secure Comput.*, early access, Jun. 4, 2019, doi: 10.1109/TDSC.2019.2920647.
[18] R. Lu, X. Lin, H. Zhu, and X. Shen, "An intelligent secure and privacy-preserving parking scheme through vehicular communications," *IEEE Trans. Veh. Technol.*, vol. 59, no. 6, pp. 2772–2785, Jul. 2010.
[19] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Privacy-preserving smart parking navigation supporting efficient driving guidance retrieval," *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 6504–6517, Jul. 2018.
[20] C. Huang, R. Lu, X. Lin, and X. Shen, "Secure automated valet parking: A privacy-preserving reservation scheme for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11169–11180, Nov. 2018.
[21] L. Zhu, M. Li, Z. Zhang, and Z. Qin, "Asap: An anonymous smart-parking and payment scheme in vehicular networks," *IEEE Trans. Dependable Secure Comput.*, vol. 7, pp. 133496–133508, Sep. 2018.
[22] M. Baza, M. Nabil, N. Lasla, K. Fidan, M. Mahmoud, and M. Abdallah, "Blockchain-based firmware update scheme tailored for autonomous vehicles," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Marrakesh, Morocco, Apr. 2019, pp. 1–7.
[23] M. Baza, N. Lasla, M. M. Mahmoud, G. Srivastava, and M. Abdallah, "B-ride: Ride sharing with privacy-preservation, trust and fair payment atop public blockchain," *IEEE Trans. Netw. Sci. Eng.*, early access, Dec. 23, 2019, doi: 10.1109/TNSE.2019.2959230.
[24] E. Chukwu and L. Garg, "A systematic review of blockchain in healthcare: Frameworks, prototypes, and implementations," *IEEE Access*, vol. 8, pp. 21196–21214, 2020.
[25] J. Wang, L. Wu, K.-K.-R. Choo, and D. He, "Blockchain-based anonymous authentication with key management for smart grid edge computing infrastructure," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 1984–1992, Mar. 2020.
[26] T.-H. Kim, G. Kumar, R. Saha, M. K. Rai, W. J. Buchanan, R. Thomas, and M. Alazab, "A privacy preserving distributed ledger framework for global human resource record management: The blockchain aspect," *IEEE Access*, vol. 8, pp. 96455–96467, 2020.
[27] T.-H. Kim, R. Goyat, M. K. Rai, G. Kumar, W. J. Buchanan, R. Saha, and R. Thomas, "A novel trust evaluation process for secure localization using a decentralized blockchain in wireless sensor networks," *IEEE Access*, vol. 7, pp. 184133–184144, 2019.
[28] J. Xie, H. Tang, T. Huang, F. R. Yu, R. Xie, J. Liu, and Y. Liu, "A survey of blockchain technology applied to smart cities: Research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2794–2830, 3rd Quart., 2019.
[29] S. J. Alsunaidi and F. A. Alhaidari, "A survey of consensus algorithms for blockchain technology," in *Proc. Int. Conf. Comput. Inf. Sci. (ICCIS)*, Sakaka, Saudi Arabia, Apr. 2019, pp. 1–6.
[30] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Proc. IEEE Int. Congr. Big Data (BigData Congress)*, Honolulu, HI, USA, Jun. 2017, pp. 557–564.
[31] G.-T. Nguyen and K. Kim, "A survey about consensus algorithms used in blockchain," *J. Inf. Process. Syst.*, vol. 14, no. 1, pp. 101–128, 2018.
[32] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proc. USENIX Annu. Tech. Conf. (USENIX ATC)*, 2014, pp. 305–319.
[33] J. P. M. Chase. (2018). *A Permissioned Implementation of Ethereum*. [Online]. Available: https://github.com/jpmorganchase/quorum
[34] R. Tajeddine, O. W. Gnilke, D. Karpuk, R. Freij-Hollanti, and C. Hollanti, "Private information retrieval from coded storage systems with colluding, Byzantine, and unresponsive servers," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3898–3906, Jun. 2019.
[35] M. Grissa, A. A. Yavuz, and B. Hamdaoui, "Location privacy in cognitive radios with multi-server private information retrieval," *IEEE Trans. Cognit. Commun. Netw.*, vol. 5, no. 4, pp. 949–962, Dec. 2019.
[36] X. Wang, W. Meng, and M. Zhang, "A novel information retrieval method based on R-tree index for smart hospital information system," *Int. J. Adv. Comput. Res.*, vol. 9, no. 42, pp. 133–145, May 2019.
[37] K. Banawan and S. Ulukus, "The capacity of private information retrieval from Byzantine and colluding databases," *IEEE Trans. Inf. Theory*, vol. 65, no. 2, pp. 1206–1219, Feb. 2019.
[38] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer-Verlag, 2003, pp. 416–432.
[39] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Proc. Cryptographers Track RSA Conf.* San Francisco, CA, USA: Springer, 2016, pp. 111–126.
[40] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *Proc. Conf. Theory Appl. Cryptol.* Berlin, Germany: Springer-Verlag, 1989, pp. 239–252.
[41] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille, "Simple schnorr multi-signatures with applications to bitcoin," *Des., Codes Cryptogr.*, vol. 87, no. 9, pp. 2139–2164, Sep. 2019.

[42] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 839–858.
[43] A. B. T. Sherif, K. Rabieh, M. M. E. A. Mahmoud, and X. Liang, "Privacy-preserving ride sharing scheme for autonomous vehicles in big data era," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 611–618, Apr. 2017.
[44] M. Pazos-Revilla, A. Alsharif, S. Gunukula, T. N. Guo, M. Mahmoud, and X. Shen, "Secure and privacy-preserving physical-layer-assisted scheme for EV dynamic charging system," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3304–3318, Apr. 2018.
[45] A. Bogner, M. Chanson, and A. Meeuw, "A decentralised sharing app running a smart contract on the ethereum blockchain," in *Proc. 6th Int. Conf. Internet Things*, 2016, pp. 177–178.
[46] D. Liu, A. Alahmadi, J. Ni, X. Lin, and X. Shen, "Anonymous reputation system for IIoT-enabled retail marketing atop PoS blockchain," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3527–3537, Jun. 2019.
[47] D. Wang, D. He, P. Wang, and C.-H. Chu, "Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 4, pp. 428–442, Jul. 2015.
[48] M. M. E. A. Mahmoud, N. Saputro, P. K. Akula, and K. Akkaya, "Privacy-preserving power injection over a hybrid AMI/LTE smart grid network," *IEEE Internet Things J.*, vol. 4, no. 4, pp. 870–880, Aug. 2017.
[49] M. Mahmoud, K. Rabieh, A. Sherif, E. Oriero, M. Ismail, E. Serpedin, and K. Qaraqe, "Privacy-preserving fine-grained data retrieval schemes for mobile social networks," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 5, pp. 871–884, Sep. 2019.
[50] M. Baza, M. Nabil, M. M. E. A. Mahmoud, N. Bewermeier, K. Fidan, W. Alasmary, and M. Abdallah, "Detecting sybil attacks using proofs of work and location in VANETs," *IEEE Trans. Dependable Secure Comput.*, early access, May 11, 2020, doi: 10.1109/TDSC.2020.2993769.
[51] J. Qiu, D. Grace, G. Ding, J. Yao, and Q. Wu, "Blockchain-based secure spectrum trading for Unmanned-Aerial-Vehicle-Assisted cellular networks: An Operator's perspective," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 451–466, Jan. 2020.
[52] L. Li, J. Liu, L. Cheng, S. Qiu, W. Wang, X. Zhang, and Z. Zhang, "CreditCoin: A privacy-preserving blockchain-based incentive announcement network for communications of smart vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2204–2220, Jul. 2018.
[53] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[54] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.
[55] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, Jun. 2013.
[56] Y. Lin and G. Yue, "Channel-adapted and buffer-aware packet scheduling in LTE wireless communication system," in *Proc. 4th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Oct. 2008, pp. 1–4.
[57] *Distributor*. Accessed: Jan. 5, 2020. [Online]. Available: https://www.disruptordaily.com/
[58] *Parkgene*. Accessed: Jan. 5, 2020. [Online]. Available: https://parkgene.io/

MAHMOUD M. BADR received the B.S. and M.S. degrees in electrical engineering from Benha University, Cairo, Egypt, in 2013 and 2018, respectively. He is currently pursuing the Ph.D. degree with Tennessee Technological University, Cookeville, TN, USA. He is currently a Graduate Research Assistant with the Department of Electrical and Computer Engineering, Tennessee Technological University. He is also holding the position of a Lecturer Assistant with the Faculty of Engineering at Shoubra, Benha University, Egypt. His research interests include blockchain, cryptography, 5G networks, network security, and smart grids.

WESAM AL AMIRI received the B.S. degree in communication engineering from Al-Balqa Applied University, Amman, Jordan, in 2014, and the M.S. degree in electrical and computer engineering from Tennessee Technological University, Cookeville, TN, USA, in 2019. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Missouri University of Science and Technology, Rolla, MO, USA. His research interests include blockchain, cryptography, and network security.

MOSTAFA M. FOUDA (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Benha University, Egypt, in 2002 and 2007, respectively, and the Ph.D. degree in information sciences from Tohoku University, Japan, in 2011. He is currently an Assistant Professor with Idaho State University, ID, USA. He also holds the position of Associate Professor with Benha University, Egypt. He has served as an Assistant Professor with Tohoku University, Japan. He was a Postdoctoral Research Associate with Tennessee Technological University, Cookeville, TN, USA. He has published over 30 articles in IEEE conference proceedings and journals. His research interests include cyber security, machine learning, blockchain, the IoT, 5G networks, and smart grid communications. He has served on the technical committees of several IEEE conferences. He is also a Reviewer in several IEEE Transactions and Magazines and an Associate Editor of IEEE Access.

MOHAMED M. E. A. MAHMOUD (Senior Member, IEEE) received the Ph.D. degree from the University of Waterloo, in April 2011. From May 2011 to May 2012, he worked as a Postdoctoral Fellow with the Broadband Communications Research Group, University of Waterloo. From August 2012 to July 2013, he worked as a Visiting Scholar with the University of Waterloo and a Postdoctoral Fellow with Ryerson University. He is currently an Associate Professor with the Department Electrical and Computer Engineering, Tennessee Technological University, USA. He is the author for more than twenty three articles published in major IEEE conferences and journals, such as INFOCOM conference and the IEEE Transactions on Vehicular Technology, *Mobile Computing*, and *Parallel and Distributed Systems*. His research interests include security and privacy preserving schemes for smart grid communication networks, mobile ad hoc networks, sensor networks, and delay-tolerant networks. He has received the NSERC-PDF Award, the Best Paper Award from IEEE International Conference on Communications (ICC 2009), Dresden, Germany. He serves as an Associate Editor in Springer journal of peer-to-peer networking and applications. He served as a Technical Program Committee Member for several IEEE conferences and a Reviewer for several journals and conferences such as the IEEE Transactions on Vehicular Technology, the IEEE Transactions on Parallel and Distributed Systems, and the *Journal of Peer-to-Peer Networking*.

ABDULAH JEZA ALJOHANI (Member, IEEE) received the B.Sc. (Eng.) degree in electronics and communication engineering from King Abdulaziz University, Jeddah, Saudi Arabia, in 2006, and the M.Sc. (Hons.) and Ph.D. degrees, awarded with no corrections, in wireless communication from the University of Southampton, Southampton, U.K., in 2010 and 2016, respectively. He is currently an Assistance Professor with the Department of Electrical and Computer Engineering, King Abdulaziz University. His research interests include machine learning, and optimization, distributed source coding, free-space optical communication, channel coding, cooperative communications, and MIMO systems.

WALEED ALASMARY (Senior Member, IEEE) received the B.Sc. degree (Hons.) in computer engineering from Umm Al-Qura University, Saudi Arabia, in 2005, the M.A.Sc. degree in electrical and computer engineering from the University of Waterloo, Canada, in 2010, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, Canada, in 2015. During his Ph.D. degree, he was a Visiting Research Scholar with the Network Research Laboratory, UCLA, in 2014. He was a Fulbright Visiting Scholar with CSAIL Laboratory, MIT, from 2016 to 2017. He subsequently joined the College of Computer and Information Systems, Umm Al-Qura University, as an Assistant Professor of computer engineering, where he currently holds an associate professor position. His Mobility impact on the IEEE 802.11p article is among the most cited *Ad Hoc Networks* journal articles list from 2016 to 2018. His current research interests include mobile computing, ubiquitous sensing, intelligent transportation systems, privacy, and anonymity.

• • •