Efficient and Privacy-Preserving Carpooling Using Blockchain-Assisted Vehicular Fog Computing

Meng Li, *Graduate Student Member, IEEE*, Liehuang Zhu, *Member, IEEE*, and Xiaodong Lin, *Fellow, IEEE*

***Abstract*—Carpooling enables passengers to share a vehicle to reduce traveling time, vehicle carbon emissions, and traffic congestion. However, the majority of passengers lean to find local drivers, but querying a remote cloud server leads to an unnecessary communication overhead and an increased response delay. Recently, fog computing is introduced to provide local data processing with low latency, but it also raises new security and privacy concerns because users' private information (e.g., identity and location) could be disclosed when these information are shared during carpooling. While they can be encrypted before transmission, it makes user matching a challenging task and malicious users can upload false locations. Moreover, carpooling records should be kept in a distributed manner to guarantee reliable data auditability. To address these problems, we propose an efficient and privacy-preserving carpooling scheme using blockchain-assisted vehicular fog computing to support conditional privacy, one-to-many matching, destination matching, and data auditability. Specifically, we authenticate users in a conditionally anonymous way. Also, we adopt private proximity test to achieve one-to-many proximity matching and extend it to efficiently establish a secret communication key between a passenger and a driver. We store all location grids into a tree and achieve get-off location matching using a range query technique. A private blockchain is built to store carpooling records. Finally, we analyze the security and privacy properties of the proposed scheme, and evaluate its performance in terms of computational costs and communication overhead.**

***Index Terms*—Blockchain, carpooling, fog computing, security and privacy, user matching.**

I. Introduction

Carpooling [1] is an effective solution to tackle the problem of increasing traffic congestion, especially at peak hours or in an extreme weather when taxi services are insufficient. In the carpooling service, drivers can post their destinations or driving routes on a platform to find potential passengers with similar travel plans, and passengers can hail vacant vehicles or share vehicles with other passenger(s) simultaneously.

Carpooling has significant social benefits [2]: it offers convenient travels and saves traveling time for passengers, and reduces the number of vehicles on road, thereby improving city mobility and alleviating traffic congestion. It also renders environmental benefits [3] through lowering vehicle emissions and noises. Given these benefits as well as commercial profits, some companies [4], [5] have been devoting great efforts to promoting the development of carpooling systems.

However, carpooling data is usually location sensitive and the delivery of carpooling queries to a remote cloud server [6]–[8] consumes a heavy bandwidth resulting in an unnecessary increased response delay. Meanwhile, the number of carpooling data is enormous: a report [9] shows that Didi Chuxing, the largest ride-hailing service provider in China, already has 450 million registered users and it provides 20 million rides everyday. Recently, fog computing [10] was proposed to extend the cloud computing's capabilities to the network edge. It can locally handle collected data from users (including passengers and drivers) by fog nodes, i.e., road-side units (RSUs) with communication and computation functions. These new features will bring about benefits such as location awareness, low latency, and geo-distribution [11]. Fog computing has already been used in vehicular networks where fog nodes locally process sensing data to provide real-time services, like navigation services [11] and surface condition monitoring [12], saving unnecessary bandwidth for transmitting data to a remote cloud server.

Unfortunately, along with the high convenience of fog computing, it raises some critical security and privacy concerns [13]–[15] from users. For instance, a continuous observation of a passenger's locations could reveal her/his frequently visited places including home, work, and restaurants since the carpooling data (e.g., identity, location, and destination) is highly sensitive. A malicious driver may report a false location to pick up more passengers in advance and break the system fairness. Meanwhile, misuses of these data have been reported: Uber tracked passengers by "God View" [16], carpooling data were leaked to unauthorized and untrusted third parties through the cloud server [17], and in a worse case, a driver was murdered by a passenger [18].

To protect privacy, passengers and drivers can use anonymous authentication and encrypt their data before they are matched with drivers and passengers. However, anonymization and encryption will make it difficult for the carpooling system to achieve one-to-many proximity matching. Here, by one-to-many proximity matching, we mean that an RSU should be able to find several available drivers for a passenger in her/his

Manuscript received May 31, 2018; revised August 24, 2018; accepted August 28, 2018. Date of publication August 31, 2018; date of current version June 19, 2019. This work was supported in part by the China National Key Research and Development Program under Grant 2016YFB0800301, and in part by the National Natural Science Foundation of China under Grant 61872041. *(Corresponding author: Liehuang Zhu.)*

M. Li and L. Zhu are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: menglibit@bit.edu.cn; liehuangz@bit.edu.cn).

X. Lin is with the Department of Physics and Computer Science, Wilfrid Laurier University, Waterloo, ON N2L 3G1, Canada.

Digital Object Identifier 10.1109/JIOT.2018.2868076

proximity at the same time. Although proximity test [19] can help a passenger find nearby drivers in a privacy-preserving way, but this method alone cannot achieve get-off location matching. While encrypting the driver's all possible destinations and matching them with the passenger's encrypted get-off location can solve this problem, it will incur a high computational cost and communication overhead for the driver. Therefore, how to match a passenger' get-off location with a driver' set of potential destinations is also a challenging task. Meanwhile, to secure the communication between a passenger and a driver after they are matched, existing work resorts to a public key approach [19], [20]. Ideally, a temporary secret key [21] should be established efficiently for the matched passenger and driver to negotiate about a specific pick-up location as well as a get-off location in their follow-up communications.

Another issue is that existing schemes [20], [22], [23] cannot provide reliable data auditability for carpooling records in case the central cloud server crashes or confronts data tampering. For instance, if the cloud server crashes, all the carpooling records will be lost, then a passenger will not be able to retrieve a lost item from the previously matched driver. If the cloud server confronts data tampering, the carpooling records are rewritten, then the law enforcements will not be able to obtain a user's evidence of crime from these records if the user has conducted criminal activities during carpooling [18]. Blockchain [24], [25] technique maintained by the cloud server, RSUs, and users can provide a possible solution to achieve data auditing. Public blockchain is not suitable in this scenario because carpooling records are related to business intelligence and user privacy such that commercial corporations, like Uber and Lyft, consider them highly confidential. Even if the corporations put encrypted data on the public blockchain, it still will expose their operation situations and statistical data [26]. Moreover, if we put too much data on the blockchain, it will require more storage space from RSUs which have limited storage capabilities. Thereby, a private ledger with limited access control [27] and low maintenance cost is more preferable.

Motivated by the aforementioned challenges, we propose an efficient and privacy-preserving carpooling (FICA) scheme using blockchain-assisted vehicular fog computing. To the best of our knowledge, this is the first work combining carpooling with fog computing and blockchain. Specifically, the main contributions of this paper are as follows.

1) We define a new security model for the carpooling system using blockchain-assisted vehicular fog computing where fog nodes are introduced to locally match passengers with drivers and a private blockchain is constructed by RSUs. Under this model, the cloud server and fog nodes are semi-honest, and a part of passengers and drivers can launch location cheating attacks. Based on this model, we propose an efficient and privacy-preserving carpooling scheme with conditional privacy, one-to-many proximity matching, destination matching, and data auditability.
2) We adopt a private proximity test with location tags [19] to achieve one-to-many proximity matching regarding current locations and extend it to establish a unique secret key between a passenger and a driver; we divide carpooling region into grids and achieve get-off location matching by a range query technique [29] efficiently.
3) We construct a private blockchain [30], [31] formed by RSUs into our carpooling system to record carpooling processes in a verifiable and immutable ledger to guarantee data auditability [32]. We store the encrypted carpooling data at the cloud server and put its hash value on the blockchain to reduce RSUs' storage costs. Combined with the trusted authority's ability to recover users' real identities, the private blockchain can keep carpooling records reliably and prevent users' potential illegal activities.
4) We prove the security and privacy properties of FICA, and evaluate the performance of FICA regarding computational costs, matching efficiency, and communication overhead.

The remainder of this paper is organized as follows. We review related work in Section II. In Section III, we formalize the system model, threat model, and design goals. Then, we revisit the preliminaries in Section IV. Section V presents our FICA scheme, followed by security and privacy analysis in Section VI and performance evaluation in Section VII. Finally, we conclude this paper in Section VIII.

II. Related Work

Ni *et al.* [20] proposed an anonymous mutual authentication (AMA) protocol by utilizing the BBS+ signature [35] such that a passenger and a driver can mutually authenticate each other without disclosing real identities. AMA also can trace the real identity of a passenger/driver by a trusted judge if a passenger/driver complains about the misbehavior of the driver/passenger. However, they used plaintext information including meeting time, location, and price to match passengers with drivers, which is a privacy breach for users.

Sherif *et al.* [22] proposed a privacy-preserving ride sharing scheme with three subschemes which used a group signature scheme to ensure user anonymity and a similarity measurement technique to preserve trip data privacy. Specifically, the entire region is divided into cells and each cell is represented by one bit in a binary vector. Each user transforms trip data into a binary vector and submits an encryption of the vector to the cloud server. The cloud server can measure the similarity of the users' encrypted trip data and match users without knowing the plaintext data. However, this application is limited because the ride can only be shared among acquaintances such as friends and employees. Besides, primary users have to report different data in different scenarios to complete matching which is an obstacle to the implementation because the primary users do not when to report the require data.

Pham *et al.* [23] presented a privacy-preserving ride-hailing service ORide which enabled the service provider to efficiently match riders with drivers without leaking either their identities or their locations, while providing accountability to deter misbehavior. Specifically, ORide used a homomorphic encryption system and applied optimizations for ciphertext packing and transformed processing. However, the riders' pick-up zone are exposed to the service provide and the service provider in ORide can revoke the identities of misbehaving users from digital certificates issued by itself, which could bring potential privacy risks to users because the service provider is a semi-honest entity. Moreover, each user has to manually request an anonymous credential from the service provider for each carpooling which is not convenient for users.

III. Problem Statement

A. System Model

The model of FICA consists of a trusted authority (TA), a cloud server, R RSUs, n_1 passengers, and n_2 drivers which are depicted in Fig. 1. We use the term user to indicate a passenger and a driver. The key notations are listed in Table I.

1) *TA* initializes the whole carpooling system, and generates public parameters and cryptographic keys for RSUs, passengers, and drivers. TA will remain offline after initialization until a user files a reasonable complaint about another user. Then TA can disclose the identify of an targeted user and sanctions this user. This role does not conflict with blockchain because it only serves as a parameter initializer and an identity discloser which stays offline for most of the time.
2) *Cloud server* collects passengers' carpooling queries and drivers' carpooling reports from distributed RSUs, helps each RSU answer different drivers' traffic queries, and assists in monitoring traffic conditions.
3) *RSU* is considered as a fog node with computing and communicating capabilities. An RSU collects real-time carpooling queries from passengers and carpooling reports from drivers, authenticates users, verifies data integrity, locally matches passengers with drivers, maintains a private blockchain, and then uploads encrypted carpooling data to the cloud server. The network is synchronous in the sense that an upper bound can be determined during which any RSU is able to communicate with other RSUs.
4) *Passenger* sends an encrypted carpooling request to a local RSU and seeks a driver and other potential passengers to share a vehicle. A passenger's carpooling request contains a pseudo identity, two timestamps, an embedding key, a hashed location, an encrypted get-off location, and a random number.
5) *Driver* awaits carpooling requests from a local RSU and provides carpooling services by sending an encrypted carpooling response to the RSU if there are available seats in the vehicle. A driver's carpooling report contains a pseudo identity, an encrypted identity, a set of location proofs, a hashed location, and a set of encrypted get-off locations.

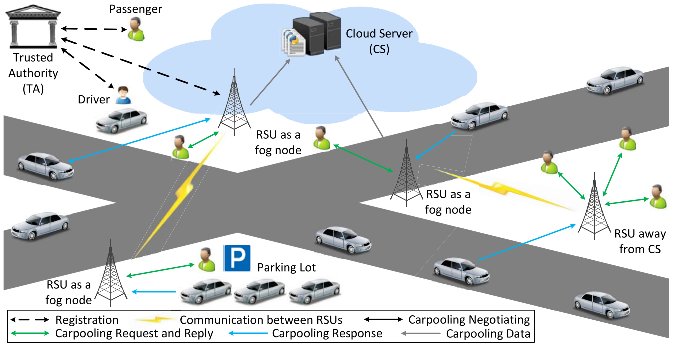


Fig. 1. System architecture.

TABLE I
Key Notations

Notation	Definition
k, q	security parameter, prime number
$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T; g_1, g_2$	multiplicative cyclic group; group generator
$a, b; \tilde{A}_1, \tilde{A}_2, \tilde{B}$	master private key; master public key
$H, \mathcal{H}$	Hash function, set of hash functions
ϕ, f_1	filtering function, length of Bloom filter
$l, g(i)$	length of public keys, ith grid
w, f_2	length of grid index, length of Bloom filter
$HMAC$	keyed hash function
R, n_1, n_2	number of RSUs, passengers, drivers
$P_i, (s_i, S_i)$	identity of a passenger, anonymous key
$k_1, k_2, ..., k_o; rv_i$	secret keys, reputation value
$x_{i1}, ..., x_{il_i}; Y_{i1}, ..., Y_{il_i}$	user's private keys, public keys
$x_{ID_{\hat{R}}}, Y_{ID_{\hat{R}}}$	RSU's secret key and public key
$pid_{P_i}; t_1, t_2$	pseudo identity; starting/ending timestamp
loc_i, go_i	current location, get-off location
REQ_i	carpooling request
x_i, Y_i	one-time secret key and public key;
$\hat{B}_{i1}, \hat{B}_{i2}$	location tag, encrypted current location
$\mathbf{g}(P_i)$	minimal set of grids covering vicinity
sk_i, pk_i	RSA secret key and public key
en_i, sh_i	encoded public key, encoded current location
$pf_i, \mathcal{PF}(g(i))$	prefix, prefix family
$r_i, \hat{B}_{i3}$	random number, encrypted get-off location
$Cert_i, \sigma_i$	anonyumous certificate, signature on REQ_i
SYN_i, D_j	sychronization message, identity of a driver
RES_j, eid	carpooling response, encrypted identity
res_i, sk_j^{com}	set of matched eids, communication key
tok_{ij}, loc_i'	authentication token, pick-up location
pay_{ij}, t_{ij2}	payment, payment timestamp
E_i, σ_i'	encrypted carpooling data, signature on E_i
$\mathcal{R}, \mathcal{C}$	carpooling record, private blockchain
$ts, \mathsf{F}(\cdot), B$	time slot, leader selection function, block
st, bn_{ts}, HR_{ts}	stake, block number, merkle hash root
H_{ts}	hash of previous blockheader
N	number of carpooling records per block,
$t_{ts}; n_1, n_2$	timestamp; number of passengers/drivers
$hatn_2, p$	number of remaining drivers, order
z, o, t	number of prefixes/secret keys/hash functions
d_1, d_2	matrix row/column [22], vector length [23]
n_3	number of points [23]

B. Threat Model

Security threats in carpooling system come from both internal and external adversaries. First, the TA is trustworthy and it is not possible to be compromised. Second, the cloud server and RSUs are honest-but-curious, and they will try to probe into users' privacy such as identity and location [22] given carpooling data. Third, most of the users are honest and will report carpooling data faithfully. A small portion of passengers and drivers can be malicious. For instance, a passenger may perform a location cheating attack, i.e., report a false location [36] to an RSU, to match a driver in advance such that the driver has to travel a long way to pick up the passenger. A driver may perform a location cheating attack (to unfairly match more passengers) and detour to gain more profits. In addition, a passenger or driver may also conduct criminal activities [18], [23]. Next, an external adversary can eavesdrop on the communication channels between any two entities and launch attacks such as impersonation attack [37], replay attack [38], and forgery attack. Last, since the passengers are not cloud server' employees and drivers are independent contractors in most carpooling systems [23], collusion attacks are not considered. In fact, if a driver colludes with the cloud server, and the driver is matched with a passenger, then the fog node will know what the driver knows including the passenger's location, destination, and everything possibly recorded by a cellphone during their face-to-face interaction.

C. Design Goals

1) *User Authentication:* Any user should be authenticated in an anonymous way and no adversary can impersonate a registered user.
2) *Location Authentication:* An RSU needs to verify whether a user is actually at the location indicated in a carpooling message before accepting the carpooling message reported by this user.
3) *Data Confidentiality and Integrity:* The contents of any carpooling message should be protected from the cloud server, RSUs, and other entities. All accepted messages should be transmitted without being altered.
4) *User Anonymity:* The identities of the users should be protected from the cloud server, RSUs, and other users (except the matched user). Given two carpooling messages, no entity can identify whether they are sent by a same user.
5) *Traceability:* The TA can track the identity of a user in case a dispute happens or a passenger tries to fetch a lost item in a previously carpooled vehicle.
6) *Efficiency:* The computational cost and the communication overhead for users and RSUs should be low in order to save processing time and network bandwidth.

IV. Preliminaries

In this section, we briefly revisit the preliminaries in building FICA, including anonymous authentication, private proximity test [19], privacy-preserving range query [29], private blockchain [30], [31], and proof of stake (PoS) [33].

A. Anonymous Authentication

The anonymous authentication scheme [28] consists of five algorithms.

1) Setup*(1^k):* takes as input a security parameter 1^k and returns system parameters params, two master private key u, v, three master public key $\tilde{A}_1, \tilde{A}_2, \tilde{B}$, and a hash function H.
2) KeyGen*(params, i):* takes as input public parameters params and an identity i, and returns an anonymous key $\mathrm{AK}_i = (s_i, S_i)$.
3) PseudoGen*(params, AK):* takes as input public parameters params and an anonymous key AK, and returns temporary private keys $(x_1, x_2, \ldots, x_l)$ and corresponding public keys $(Y_1, Y_2, \ldots, Y_l)$.
4) Sign*(params, S, x, Y, m):* Given public parameters params, an anonymous key S, a temporary private key x, a temporary public key Y, and a message m, computes an anonymous certificate Cert and a signature σ on m.
5) Verify*(params, m, σ, Cert):* Given public parameters params, a message m, a signature σ, and an anonymous certificate Cert, verifies the validity of Cert and σ. Outputs 1 if the verifications hold, or $\perp$ otherwise.
6) Open*(params, Cert):* Given public parameters params and an anonymous certificate Cert, outputs an anonymous key S.

B. Private Proximity Test With Location Tags

A private proximity test with location tags [19] contains two phases.

1) *Location Based Handshake:* A user Alice sends to the cloud server a request message REQ_A containing a candidate filtering function φ, a location tag filtering function ϕ, two timestamps t_1 and t_2, and a hash function H. Upon receiving REQ_A, the cloud server identifies candidates based on φ and broadcasts to all candidates a synchronization message SYN. After receiving SYN, each user collects a set of environmental signals and extracts a location tag. Alice inserts observations into an empty Bloom filter [39]. Alice computes a pair of RSA secret and public keys, and embeds the public key into the location tag. Then Alice sends an embedding message EMB_A to the cloud server. The cloud server broadcasts $\mathrm{EMB}_{\mathrm{Alice}}$ to all candidates. A candidate Bob in Alice's proximity can extract pk_A correctly.
2) *Private Proximity Test:* Alice defines the vicinity region and inserts each grid index into another empty Bloom filter and sends a proximity test message $\mathrm{PRO}_{\mathrm{Alice}}$ to the cloud server. Bob encrypts her/his identity with Alice's public key to form $\mathrm{eid}_{\mathrm{Bob}}$, broadcasts it, and collects other users' eid. Then Bob sends a response message $\mathrm{RSP}_{\mathrm{Bob}}$ to the cloud server. The cloud server filters candidates for Alice using Bloom filter query function and sends remaining eid to Alice. Alice decrypts eid to get the candidate's identity and retrieves the public key from a certificate authority.

C. Privacy-Preserving Range Query

A privacy-preserving range query scheme [29] has five steps.

1) *Prefix Encoding:* Given a number n with a binary format $b_1 b_2, \ldots, b_w$, data owner computes the prefix family $PF(n)$ which is the set of $w + 1$ prefixes $\{b_1 b_2 \ldots b_w, \ldots, b_1 * \ldots **, ** \ldots *\}$; given a range $[lb, rb]$, the data user computes a minimum set of prefixes $\mathcal{MS}([lb, rb])$ such that the union of prefixes is $[lb, rb]$.
2) *Tree Construction:* Given d prefixes families, data owner constructs a highly balanced binary search tree PBTree pbt in a top-down fashion.

3) *Node Randomization:* The data owner and the data user share o secrets $k_1, \ldots, k_o$. For each prefix pr in a node on pbt, data owner computes o secure keyed hashes $\mathrm{HMAC}(k_1, pr), \ldots, \mathrm{HMAC}(k_o, pr)$, randomize them by computing $\mathrm{HMAC}(r, \mathrm{HMAC}(k_1, pr)), \ldots, \mathrm{HMAC}(r, \mathrm{HMAC}(k_o, pr))$ with a random number r, and then inserts them into a Bloom filter. Finally, the data owner sends encrypted data items and pbt to a data center.
4) *Trapdoor Computation:* Given a query range $[lb, rb]$ whose $\mathrm{MS}([lb, rb])$ has z prefixes $pr_1, \ldots, pr_z$. For each prefix, the data user computes t hashes as in last step, and the trapdoor for range query $[lb, rb]$ is a matrix $M_{[lb,rb]}$ of $z \cdot t$ hashes. Finally, the data user sends $M_{[lb,rb]}$ to the data center.
5) *Query Process:* Given a range query trapdoor $M_{[lb,rb]}$, the data center searches the PBTree pbt and checks whether there is a row j in $M_{[lb,rb]}$ such that the result of hashing $(r, \mathrm{HMAC}(k_i, pr_j))$ into the Bloom filter is 1.

D. Private Blockchain and Proof of Stake

Blockchain [24] is a distributed ledger, first used in the Bitcoin cryptocurrency [34] for economical purposes. It achieves data auditing by adding authenticated blocks to the chain; in addition, since it works in a peer-to-peer network, it is not controlled by a centralized entity and it allows anyone to add a new block using a proof-of-work mechanism.

In some applications, like database management and auditing within a company, public readability may not be preferable in consideration of business intelligence and user privacy. Therefore, the private blockchain is introduced. A private blockchain [30], [31] is a blockchain where write permissions only belong to the owner and read permissions are public or restricted to permitted parties. Specifically, a company or owner sets up a private blockchain which is a permissioned network and participants have to obtain a permission to join the network. Once a participant has joined the network, it will maintain the private blockchain in a decentralized manner.

An alternative consensus mechanism relies on the concept of PoS [33]. Instead of requiring nodes to spend computational resources in order to be the winning miner, it runs a process that randomly selects one of nodes proportionally to the stake that each node possesses as recorded on the blockchain.

V. Proposed Scheme

In this section, we propose the efficient and privacy-preserving carpooling scheme with six phases: 1) system initialization; 2) entity registration; 3) carpooling requesting; 4) carpooling responding; 5) carpooling termination and cancellation; and 6) user tracking.

A. System Initialization

First, given a security parameter 1^k, TA generates three multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of the same prime order q with two generators g_1 for $\mathbb{G}_1$, g_2 for $\mathbb{G}_2$ and an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Then TA chooses two random numbers $a, b \in \mathbb{Z}_q^*$ as master private keys, chooses a hash function $H : \{0, 1\}^* \rightarrow Z_q^*$, and computes $\tilde{A}_1 = g_1^a, \tilde{A}_2 = g_2^a, \tilde{B} = g_1^b$. Second, TA chooses a filtering function ϕ, a length f_1, and a set of hash functions $\mathcal{H}$ for the Bloom filter, and a length l for public keys. TA divides the carpooling area into small grids $g(i)$s and they are constructed into a tree. For instance, leaf nodes $g(1), g(2), g(3)$, and $g(4)$ are linked to a parent node $g(17)$, and $g(17), g(18), g(19), g(20)$ are linked to a parent node $g(21)$. Third, TA sets a length w for grid indexes and a length f_2 for Bloom filters, and chooses a secure keyed hash function HMAC. Finally, TA publishes the system parameters $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \tilde{A}_1, \tilde{A}_2, \tilde{B}, w, f_1, f_2, l, \mathbf{g}(i), \phi, H, \mathcal{H}, \mathrm{HMAC})$.

B. Entity Registration

When a passenger with identity P_i joins the carpooling system, TA chooses a random number $s_i \in \mathbb{Z}_q^*$ with $s_i + a \neq 0 \bmod q$ and random number $k_1, k_2, \ldots, k_o$, and computes $S_i = g_1^{1/(s_i+a)}$. TA stores (P_i, S_i^a) in its tracking list and returns secret keys $k_1, k_2, \ldots, k_o$, authorized anonymous key $\mathrm{AK}_i = (s_i, S_i)$ and an initial reputation value rv_i signed by itself to P_i. Then P_i chooses l_i random numbers $(x_{i1}, x_{i2}, \ldots, x_{il_i})$ as private keys and computes public keys $Y_j = g_1^{x_j}$ for $j = 1, 2, \ldots, l_i$. Similarly, a driver D_j obtains $k_1, k_2, \ldots, k_o, \mathrm{AK}_j, rp_j$, and $\{x_{io}, Y_{io}\}_{o=1}^{l_j}$. An RSU $\mathrm{ID}_{\hat{R}}$ also registers to the TA and obtains a secret key $x_{\mathrm{ID}_{\hat{R}}}$ and a public key $Y_{\mathrm{ID}_{\hat{R}}}$. The main phases of FICA is shown in Fig. 2.

C. Carpooling Requesting

A passenger P_i with a time-varying pseudo identity pid_{P_i}, starting and ending timestamp t_1, t_2 to synchronize the generation of location tags, a current location loc_i, a get-off location go_i, a one-time secret key x_i and its corresponding public key Y_i, and a reputation value $\{rv_i\}_{\mathrm{TA}}$ generates a carpooling request REQ_i as follows.

1) P_i collects a set of environmental signals $\mathcal{X}_i(t_1, t_2)$ and extracts a location tag $\mathcal{Y}_i(t_1, t_2) = \phi(\mathcal{X}_i(t_1, t_2))$. P_i inserts observations $y_i(t_1, t_2)$ into an empty Bloom filter $\{0\}^{f_1}$ through $\mathcal{H}$ to obtain a location tag $\hat{B}_{i1} = \mathsf{Insert}(H(y_i(t_1, t_2)), \hat{B}_{i1})$. Then P_i transforms loc_i into $\mathbf{g}(P_i)$ to be the minimal set of grids covering the vicinity region and inserts each grid index into another Bloom filter $\hat{B}_{i2} = \mathsf{Insert}(H(\mathbf{g}_i(P_i)||pk_i), \hat{B}_{i2})$. In this way, the passenger has encrypted the current location.
2) P_i computes a pair of RSA keys $sk_i, pk_i \in \{0, 1\}^l$ and embeds pk_i into $\hat{B}_{i1}$ using the code-offset construction for fuzzy extractors [40]: $en_i = \mathsf{Encode}(f_1, l, pk_i)$ and a shift $sh_i = en - \hat{B}_{i1}$. By doing so, the passenger has embedded the public key for one-to-many proximity matching.
3) P_i finds go_i in a grid with index $g(i)$ and transfers $g(i)$ into binary format to form a prefix family $\mathcal{PF}(g(i))$ [29]. P_i chooses a random number r_i and computes $\{\mathrm{HMAC}(r_i, \mathrm{HMAC}(k_j, pf_i)\}_{j=1}^o$ for each prefix pf_i. P_i inserts them into an empty Bloom filter $\{0\}^{f_2}$ to obtain

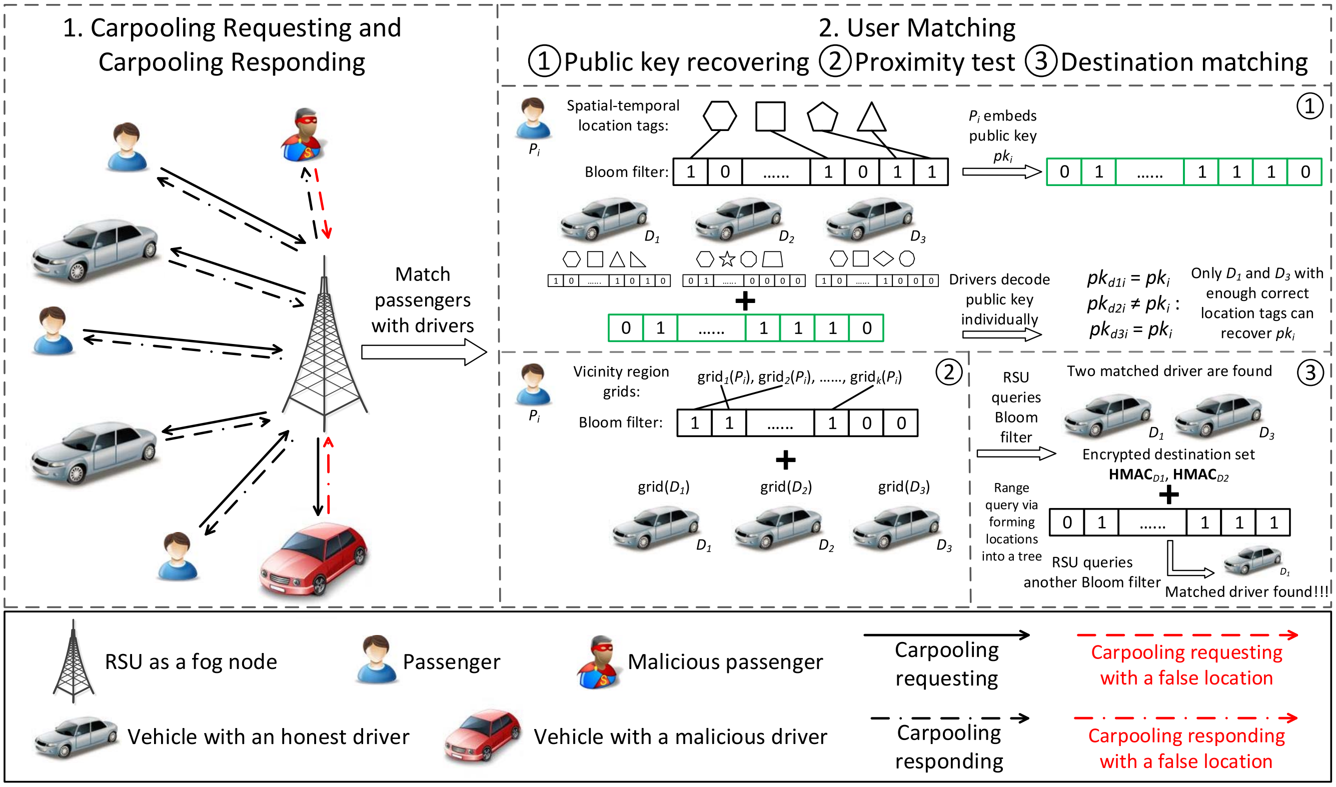

1. Carpooling Requesting and Carpooling Responding
Match passengers with drivers
2. User Matching
①Public key recovering ②Proximity test ③Destination matching
Spatial-temporal location tags:
P_i
Bloom filter: 1 0 ...... 1 0 1 1
P_i embeds public key pk_i
0 1 ...... 1 1 1 0
①
D_1 D_2 D_3
0 1 ...... 1 1 1 0
Drivers decode public key individually
$pk_{d1i} = pk_i$
$pk_{d2i} \neq pk_i$:
$pk_{d3i} = pk_i$
Only D_1 and D_3 with enough correct location tags can recover pk_i
Vicinity region grids:
grid$_1$(P_i), grid$_2$(P_i), ..., grid$_k$(P_i)
②
Bloom filter: 1 1 ...... 1 0 0
grid(D_1) grid(D_2) grid(D_3)
RSU queries Bloom filter
Two matched driver are found
③
Encrypted destination set
HMAC$_{D1}$, HMAC$_{D2}$
Range query via forming locations into a tree
0 1 ...... 1 1 1
RSU queries another Bloom filter
Matched driver found!!!
RSU as a fog node
Passenger
Malicious passenger
Carpooling requesting
Carpooling requesting with a false location
Vehicle with an honest driver
Vehicle with a malicious driver
Carpooling responding
Carpooling responding with a false location

Fig. 2. Carpooling requesting, carpooling responding, and user matching.

$\hat{B}_{i3}$. By this means, the passenger has encrypted the destination for get-off location matching. Finally, P_i has formed a carpooling request

$$\mathrm{REQ}_i = \{\mathrm{pid}_{P_i}, t_1, t_2, sh_i, \hat{B}_{i2}, \hat{B}_{i3}, r_i, \{rv_i\}_{\mathrm{TA}}\}. \quad (1)$$

Next, P_i computes an anonymous certificate Cert_i as follows. P_i chooses four random numbers $r_0, r_1, r_2, r_3 \in \mathbb{Z}_q^*$ and computes $V_1 = A_1^{r_0}$, $V_2 = S_i \cdot B^{r_0}$, $\eta_0 = r_0 \cdot x_i \bmod q$, $\eta_1 = A_1^{r_1}$, $\eta_2 = V_1^{r_2}/A_1^{r_3}$, $\eta_3 = e(V_2, g_2^{r_2})/e(B, A_2^{r_1 \cdot g_2^{r_3}})$, $C_i = H(A_1|| B|| Y_i|| V_1||V_2||\eta_1||\eta_2|| \eta_3)$, $ss_1 = r_1 + C \cdot r_0 \bmod q$, $ss_2 = r_2 + C \cdot x_i \bmod q$, $ss_3 = r_3 + C \cdot \eta_0 \bmod q$. Finally, P_i sets an anonymous certificate $\mathrm{Cert}_i = \{Y_i||V_1||V_2||C||ss_1||ss_2||ss_3\}$ and signs REQ_i by computing a signature $\sigma_i = g_2^{1/(x_i + H(\mathrm{REQ}_i))}$, and sends $\{\mathrm{REQ}_i, \mathrm{Cert}_i, \sigma_i\}$ to a local RSU $\mathrm{ID}_{\hat{R}}$.

D. Carpooling Responding

After receiving an encrypted carpooling request $\{\mathrm{pid}_{P_i}, \mathrm{REQ}_i, \mathrm{Cert}_i, \sigma_i\}$, the local RSU $\mathrm{ID}_{\hat{R}}$ first checks the validity of Cert_i by computing $\eta'_1 = A_1^{ss_1}/V_1^C$, $\eta'_2 = V_1^{ss_2}/A_1^{ss_3}$, $\eta'_3 = ([e(V_2, g_2^{ss_2} \cdot A_2)]/[e(B, A_2^{ss_1} \cdot g_2^{ss_3})e(g_1, g_2^C)])$ and checks $C \overset{?}{=} H(A_1||B||Y_i||V_1||V_2||\eta'_1||\eta'_2||\eta'_3)$. If it holds, the RSU verifies the validity of σ_i by checking $e(Y_i \cdot g_1^{H(\mathrm{REQ}_i)}, \sigma_i) \overset{?}{=} e(g_1, g_2)$. If either of them fails, it discards the query, or broadcasts a synchronization message

$$\mathrm{SYN}_i = \{t_1, t_2, sh_i\}_{\mathrm{ID}_{\hat{R}}} \quad (2)$$

to drivers within its coverage area. Note that rvs are used to sort the broadcasting order of carpooling requests such that a passenger with a higher reputation value will be served earlier.

After receiving SYN_i, each available driver D_j with a time-varying pseudo name pid_{D_j}, a current location loc_j, a set of get-off locations $\mathbf{go}_j$, a one-time secret key x_j and its corresponding public key Y_j, and a reputation value $\{rv_j\}_{\mathrm{TA}}$ generates a carpooling response RES_j as follows.

1) D_j computes $\hat{B}_{j1}$ and $\hat{B}_{j2}$ as the passenger did. D_j computes $en_j = sh_i - \hat{B}_{j1}$ and decodes en_j by calculating $pk'_i = \mathsf{Decode}(f_1, l, en_j)$. In this way, only when D_j is in P_i's vicinity, i.e., $\mathsf{Ham}(en_i, en_j) <$ err where err is the maximum number of bits that BCH coding can correct, will D_j recover pk_i. Meanwhile, D_j cannot verify whether he has retrieved pk_i or not.
2) D_j chooses a secret communication key sk_j^{com} and computes $\mathrm{eid}_j = \mathsf{ENC}(pk'_i, \mathrm{pid}_{D_j}||sk_j^{\mathrm{com}})$ using RSA encryption and broadcasts eid_j through Wi-Fi or Bluetooth. D_j collects eids shared by nearby drivers to form a set of location proofs $\mathcal{P}_j$. Then D_j transforms loc_j into a grid $g(D_j)$ and computes $H(g(D_j)||pk'_i)$. By doing so, D_j has encrypted the current location.
3) D_j finds $\mathbf{go}_j$ into a grid set $\mathbf{g}(D_j)$ which includes the minimal grid $g_{\min}(D_j)$ and the maximal grid $g_{\max}(D_j)$. D_j converts $[g_{\min}(D_j), g_{\max}(D_j)]$ into a minimum set of prefixes consisting of z prefixes pr_i, $1 < i < z$. D_j computes a set of hashes $\mathcal{HMAC}_j = \{\mathrm{HMAC}(r_i, \mathrm{HMAC}(k_1, pr_1)), \ldots, \mathrm{HMAC}(r_i, \mathrm{HMAC}(k_t, pr_z))\}$. By this means, the driver has encrypted trapdoors for get-off location matching. Finally, D_j forms a carpooling response

$$\mathrm{RES}_j = \{\mathrm{pid}_{D_j}, \mathrm{eid}_j, \mathcal{P}_j, H(g(D_j)||pk'_i), \mathcal{HMAC}_j\}. \quad (3)$$

Then D_j computes Cert_j, generates σ_j, and sends $\{\text{RES}_j, \sigma_j, \text{Cert}_j\}$ to the local RSU $\text{ID}_{\hat{R}}$.

E. Carpooling Matching and Uploading

After receiving an encrypted carpooling response: $\{\text{pid}_{D_j}, \text{RES}_j, \text{Cert}_j, \sigma_j\}$ from D_j, the local RSU $\text{ID}_{\hat{R}}$ first checks the validity of Cert_j and σ_j. If they pass verification, $\text{ID}_{\hat{R}}$ will perform user matching as follows.

1) *One-to-Many Proximity Matching:* $\text{ID}_{\hat{R}}$ queries $H(g(D_j||pk_j'))$ into $\hat{B}_{i2}$ and checks if the query result is true. Let res_i be the set of remaining eids which exist in $\hat{B}_{i2}$.
2) *Get-Off Location Matching:* $\text{ID}_{\hat{R}}$ queries each item in $\mathcal{HMAC}_j$ into $\hat{B}_{i3}$ and checks if the query result is true. Then $\text{ID}_{\hat{R}}$ records a positive number pn for each driver and sort res_i based on pn.

To screen potential misbehaved drivers, $\text{ID}_{\hat{R}}$ constructs a directional graph $\mathcal{G}$ in which each vertex is an eid of a driver who passes the get-off location matching and each edge points from one driver to one of the eids in her/his location proofs. $\text{ID}_{\hat{R}}$ divides $\mathcal{G}$ into subgraphs using the eigenvector-based partition method. For each subgraph, $\text{ID}_{\hat{R}}$ eliminates the vertices with significantly fewer incoming edges than outgoing edges and updates res_i. Then $\text{ID}_{\hat{R}}$ replies to P_i the $\{\text{res}_i\}_{\text{ID}_{\hat{R}}}$ ordered by drivers' reputation values. P_i decrypts each eid in res_i by computing $\text{pid}_{D_j}||sk_j^{\text{com}} = \mathsf{Dec}(\text{eid}, sk_i)$. Finally, P_i can securely communicate with D_j using the secret key sk_j^{com} for follow-up communications, such as negotiating a specific pick-up location, how long for the driver to arrive and a specific get-off location.

After P_i and the matched driver D_j start their carpooling trip, they send to $\text{ID}_{\hat{R}}$ two confirmation messages $\{\text{ID}_{\hat{R}}, \text{pid}_{P_i}, \text{pid}_{D_j}, \text{eid}_j\}_i$ and $\{\text{ID}_{\hat{R}}, \text{pid}_{D_j}, \text{pid}_{P_i}, \text{eid}_j\}_j$, respectively. $\text{ID}_{\hat{R}}$ returns an authentication token $tok_{ij} = \mathsf{Sig}(x_{\text{ID}_{\hat{R}}}, \text{pid}_{P_i}||\text{pid}_{D_j}||t_{ij1})$ to P_i and D_j, where t_{ij1} is the negotiation timestamp. The token is used for authentication in uploading carpooling records and encrypted carpooling data in a new RSU' coverage area when the ride is over.

When P_i passenger arrives at the get-off location, P_i pays D_j with cash, debit/credit card, or mobile payment (e.g., Alipay and QR code). P_i encrypted identity i, pick-up location loc_i', get-off location go_i, payment pay_{ij}, and timestamp t_{ij2} with Y_i to obtain

$$E_i = \mathsf{Enc}(x_i, P_i||\text{loc}_i'||\text{go}_i||\text{pay}_{ij}||t_{ij2}||r_i') \tag{4}$$

(using AES encryption) and computes its hash value $H(E_i)$. Here, E_i is to be stored at the cloud server for detailed data auditing if necessary. D_j encrypted j, loc_{ij}', go_i, pay_{ij} and t_{ij2} with Y_j to obtain

$$E_j = \mathsf{Enc}(x_j, D_j||\text{loc}_j||\text{go}_j||\text{pay}_{ij}||t_{ij2}||r_j') \tag{5}$$

and computes its hash value $H(E_j)$. The passenger and driver also exchange their $H(E_i)$ and $H(E_j)$ in order to generate two signatures $\sigma_i' = \mathsf{Sig}(x_i,\ \hat{R}||\ \text{pid}_{P_i}||\ Y_i||\ H(E_i)||\ \text{pid}_{D_j}||\ Y_j||\ H(E_j))$ and $\sigma_j' = \mathsf{Sig}(x_j,\ \hat{R}||\ \text{pid}_{P_i}||\ Y_i||\ H(E_i)||\ \text{pid}_{D_j}||\ Y_j||\ H(E_j))$, respectively. Finally, D_i and P_j send carpooling records, encrypted carpooling data, and tokens from $\text{ID}_{\hat{R}}$ to a new local RSU.

Given a carpooling record (i.e., transaction)

$$\mathcal{R} = \{\text{ID}_{\hat{R}}, \text{pid}_{P_i}, Y_i, H(E_i), \sigma_i', \text{pid}_{D_j}, Y_j, H(E_j), \sigma_j'\} \tag{6}$$

as well as encrypted data E_i and E_j, $\text{ID}_{\hat{R}}$ sends $\mathcal{R}$ with a signature to the network for verification. Here, Y_i and Y_j include (V_1, V_2) which can be used to recover P_i's and D_j's real identities. We assume that there is a record pool containing all the records to be added on the blockchain. $\text{ID}_{\hat{R}}$ sends $\{\text{pid}_{P_i}, E_i, \text{pid}_{D_j}, E_j\}_{\text{ID}_{\hat{R}}}$ to the cloud server and prepares to put N carpooling records on the private blockchain.

Specifically, we consider each RSU as a stakeholder with its own stake which is the number of carpooling records. The cloud server, RSUs, and users construct a private blockchain $\mathcal{C}$ using PoS to avoid forks as follows.

1) *Basic Setting:* Time is divided into a sequence of time slots $\{ts_1, ts_2, \ldots\}$ and a private ledger is appended with one ledger block in each time slot. RSUs are equipped with synchronized clocks that indicate the current slot to carry out a distributed protocol intending to collectively add a block in current slot [33]. A leader selection function $\mathsf{F}(\cdot)$ is also assigned to each RSU. Each user is within one RSU's coverage area and he/she can access the blockchain through the RSUs.
2) *First Stage:* There is an initial stake distribution which is hardcoded into the genesis block B_0 including the RSUs' identities $\{\text{ID}_i\}_{i=1}^R$, public keys $\{Y_i\}_{i=1}^R$ and stakes $\{st_i\}_{i=1}^R$. The genesis block has an empty blockheader and a signature generated by the cloud server. Each RSU sets $\mathcal{C} = B_0$.
3) *Leader Selection:* For each time slot ts_i, each RSU collects first N carpooling records from the record pool and verifies the validity of each record $\mathcal{R}$ by checking $\mathsf{Vrf}(Y_i, \mathcal{R}, \sigma_i') \overset{?}{=} 1$ and $\mathsf{Vrf}(Y_j, \mathcal{R}, \sigma_j') \overset{?}{=} 1$. A slot leader L_i is selected who has the right to generate a new block and its probability p_i of being selected is proportional to its stake recorded in previous block. Each RSU $\hat{R}$ runs $\mathsf{F}(\cdot)$ which takes inputs $\{Y_{\text{ID}_j}\}_{j=1}^R$, $\{st_{\text{ID}_j}\}_{j=1}^R$, $p_{\hat{R}}$, and ts_i and outputs an RSU $\text{ID}_{\hat{W}} \in \{\text{ID}_1, \text{ID}_2, \ldots, \text{ID}_R\}$, where $p_{\hat{R}} = st_{\text{ID}_{\hat{R}}} / \sum_{j=1}^R st_{\text{ID}_i}$.
4) *Block Generation:* The selected RSU $\text{ID}_{\hat{W}}$ generates a new block B_{ts_i} consisting of a blockheader (including a block number bn_{ts_i}, a hash of previous blockheader H_{ts_i}, a merkle hash root HR_{ts_i} of Merkle tree constructed from N records, a timestamp t_{ts_i}), RSUs' updated stakes st_{ts_i}, N records and a signature

$$\sigma_{\text{ID}_{\hat{R}}}^{ts_i} = \mathsf{Sig}(x_{\hat{R}}, bn_{ts_i}, H_{ts_i}, HR_{ts_i}, t_{ts_i}, st_{ts_i}). \tag{7}$$

Then it adds this block to the private blockchain and notifies the network. The construction of the private blockchain is depicted in Fig. 3.

Discussions on Our Blockchain:

1) *PoW [41] Versus PoS:* We did not choose PoW in selecting a winning miner because a primary concern in PoW is the energy required for its execution and extensive hash functions in solving cryptographic puzzles will

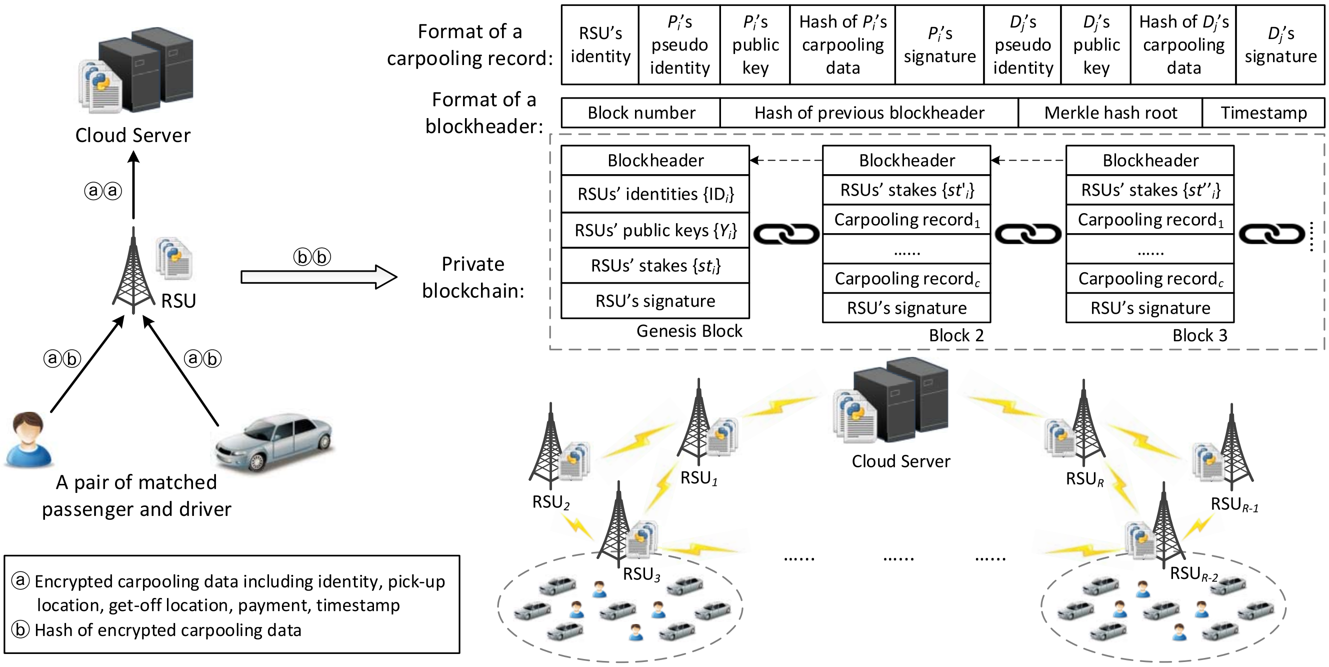


Fig. 3. Construction of the private blockchain.

consume a lot of RSU's energy and time, which is not necessary. Each RSU has its own number of carpooling records which can be deemed as its stake or balance. Therefore, we use PoS to reach consensus in FICA.

2) *The Existence of TA:* The TA is deployed to disclose a targeted user's real identity which we will explain in Section V-G. This setting does not conflict with the blockchain decentration because: a) blockchain decentration means the data is stored among nodes in a distributed ledger; b) the TA's task is to register entities and disclose real identity when needed which has nothing to do with the ledger; and c) the TA stays offline for most of the time which brings no extra response delay to the network.
3) *Compromised RSU:* We note that if an RSU is *compromised* by an adversary to add a new block containing meaningless data to the blockchain, we can detect this injection attack by verifiable computing [42].

F. Carpooling Termination and Cancellation

When the passenger P_i arrives at the get-off location, P_i generates several e-cashes [43] with a timestamp signed by pk_i and sends it to the driver D_j. Later, D_j can cash it from the cloud server. Moreover, if a user U decides to drop off from the carpooling process before one-to-many proximity matching, U can send a cancellation message to the RSU which will remove U's carpooling request or response from its processing queue.

G. User Tracking

If a complaint has been filed against a malicious user with a pseudo name pid_{P_i} and an anonymous certificate Cert_i, the TA computes $V_2^u/V_1^v = S_i^a \cdot B^{ar_0}/A_1^{br_0} = S_i^a \cdot g_1^{abr_0}/g_1^{abr_0} = S_i^a$ and tracks the identity of driver i through looking up its tracking list $\{P_i, S_i^a\}_{i=1}^{n_1+n_2}$. If a matched record is found, TA could sanction the user by notifying the cloud server of decreasing the user's credibility in the carpooling system, denying the next attendance to the system and adding this user into a blacklist.

VI. Security and Privacy Analysis

A. User Authentication

We utilized the anonymous authentication scheme [28] to achieve anonymous authentication, and the RSU cannot recover a user i's S_i from (V_1, V_2), but it acknowledges the anonymous driver though a zero-knowledge proof of knowledge of s_i.

B. Location Authentication

We deem that a driver D_j successfully launches a location cheating attack if one of the following two conditions holds [19]: 1) $\mathcal{L}_j$ is different from $\mathcal{L}_i$, but D_j successfully extracts pk_i and 2) $\hat{g}(D_j)$ is outside of $\hat{\mathbf{g}}(P_i)$, but D_j convinces P_i that $\hat{g}(D_j)$ is within $\hat{\mathbf{g}}(P_i)$. The first scenario indicates that the symmetric difference between $\mathcal{L}_i$ and $\mathcal{L}_j$ is bigger that a threshold T. Due to the false positives of Bloom filters, there is a possibility that $\mathsf{Ham}(\hat{B}_i, \hat{B}_j) \leq \text{err}$. In this case, D_j can extract pk_i from far away. But P_i can reduce the possibility by increasing the size of the Bloom filter. In the second scenario, although D_j could use another $\hat{g}'(D_j) \in \hat{\mathbf{g}}(P_i)$ to pass the Bloom filter querying, still D_j cannot provide a valid location proof $\mathcal{P}_j$ to identify other drivers present in $\hat{g}'(D_i)$. Therefore, FICA is robust against location cheating from malicious users, thus providing location authentication. In addition, if several malicious drivers collude with each other and use mutual proofs to trick an RSU, the RSU can use certified location information for the proximity test [44] to defend this attack.

C. Data Confidentiality and Integrity

First, a passenger P_i's current location is transformed into a fuzzy extractor helper string $\hat{s}_i$ and a Bloom filter $\hat{B}_i$. Since the cloud server or the RSU are not physically near P_i, they cannot obtain P_i public key pk_i and use it to query $\hat{B}$. Therefore, they remain oblivious of P_i [19]. Any driver D_j cannot tell whether she/he has extracted pk_i correctly, due to the nature of the BCH decoder. Therefore, D_j cannot locate P_i whether he is within the coverage of P_i's location tag. A driver's current location is transformed into K hash values $h(\hat{g}'(B)||pk_i')$. Similarly, the cloud server and RSUs cannot learn D_j's location without P_i's public key. Additionally, since the cloud server does not forward $h(\hat{g}'(B)||pk_i')$ to P_i, P_i only learns D_j's identity without knowing D_j's exact location if the proximity test returns positive. Second, the users' get-off locations are inserted and randomized into Bloom filters which are index indistinguishability under the indistinguishability against chosen keyword attack [29]. Third, the data integrity is guaranteed by users' signatures [28].

D. User Anonymity

The user's anonymous certificate does not leak any information about user's identity and it is a time-varying value. We require that users have to change their pseudo names in each carpooling requesting and responding. Therefore, the user anonymity is guaranteed.

E. Traceability

When a dispute happens, TA first has the capability to track the identity of a misbehaved user from anonymous certificate Cert and its tracking list $\{(P_i, S_i)\}$ given a carpooling record $\{\text{pid}_{P_i}, \text{pid}_{D_j}, \text{ID}_{\hat{R}}, E(H_i), E(H_j), t_3\}_{\text{ID}_{\hat{R}}}$. Then the privacy of the user can be revoked. Next, if the complaint initiator is the passenger P, TA recovers sk_j^{com} from sk_i provided by P and eid_j to disclose negotiation details; if the driver files an complaint, TA recovers sk_j^{com} from encrypting $\text{pid}_{D_j}||sk_j^{\text{com}}$ by pk_j' and compare it with eid_j.

F. Data Auditability

We build a private blockchain which adept at keeping carpooling records in an easily verifiable and timestamped tamper-proof manner. First, participants (RSUs) need to obtain an invitation or permission to join the network and no adversaries can add false blocks into the blockchain since the private blockchain is a permission network. Second, each RSU maintains a replica of a shared append-only ledger of carpooling records in case the cloud server crashes. Third, the immutability of transactions on the blockchain means that carpooling records between passengers and drivers will not be subject to tampering, guaranteeing that carpooling data can be audited based on the blockchain.

Finally, we give Table II to show the comparison of security properties between FICA and existing work.

VII. PERFORMANCE EVALUATION

In this section, we first evaluate the performance of FICA in carpooling regarding computational costs and communication overhead, and then conduct experiments on the private blockchain. We conduct experiments on a laptop with Intel Core i7-7500 CPU @ 2.70GHz and 8.00GB memory. The cryptographic toolset we use is Miracl [45]. The elliptic curve is defined as $y^2 = x^3 + 1$ over $\mathbb{F}_p$, where $|p|$ is 512 bits. The detailed values of experimental parameters are listed in Table III.

TABLE II
COMPARISON OF SECURITY PROPERTIES

Property	AMA	DAP	ORide	FICA
User Authentication	√	√	√	√
Location Authentication	×	×	√	√
Data Confidentiality	×	√	√	√
Data Integrity	√	√	×	√
User Anonymity	√	√	√	√
Traceability	√	√	√	√
Data Auditability	×	×	×	√

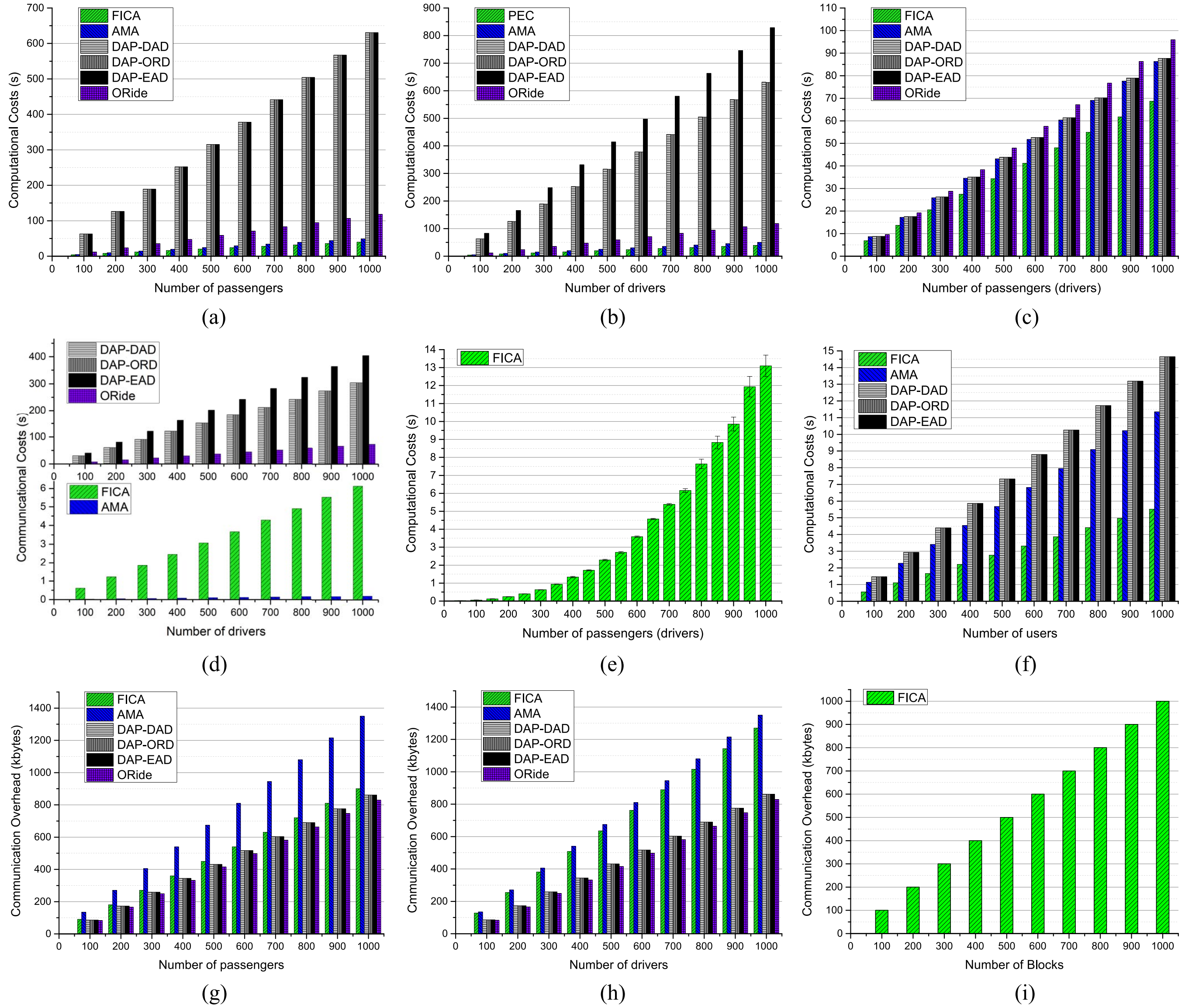
TABLE III
EXPERIMENTAL PARAMETERS

Parameters	Value
$n_1,\ n_2$	$[100,\ 1000]$
$q,\ p$	$\|p\| = 160,\ \|q\| = 512$
$H,$	$H:\ SHA256, \|pid\| = 10$
$t_1,\ t_2$	$\|t\| = \|tt\| = 2^{12}$
$z,\ o,\ t,\ \hat{n}_2$	$z = 5,\ o = 3,\ t = 5,\ \hat{n}_2 = 5$
$d_1,\ d_2,\ n_3$	$d_1 = 10,\ d_2 = 4096,\ n_3 = 4096$

A. Computational Cost in Carpooling

We simulate a vehicular scenario where an RSU receives massages from 1000 passengers and 1000 drivers. We count the number of the cryptographic operations to demonstrate the computational costs of FICA. In carpooling requesting, each passenger conducts $3|\mathcal{H}| + |\mathcal{Y}| + 1$ hash functions, 4 additions, 4 multiplications, 1 division, 1 addition in $\mathbb{G}_1$, 6 exponentiations in $\mathbb{G}_1$, 3 exponentiations in $\mathbb{G}_2$, 1 division in $\mathbb{G}_1$, 2 bilinear pairings, and 1 division in $\mathbb{G}_T$. In carpooling querying phase, it costs 31.4 ms for a passenger to generate a carpooling query. In carpooling responding phase, besides the generation of an anonymous certificate and a signature, each driver runs 1 exponentiation and $2tz + 1$ hash functions and it costs 32 ms for a driver to generate a carpooling response. In carpooling matching and uploading phase, each RSU first checks the validity of drivers' anonymous certificates and signatures and performs $1 + kz/2$ hash functions to find a match. Here, we assume that only a half of drivers passes the proximity test. Then the RSU calculates a Merkle root, generates a signature, and checks the validity of carpooling records for the other $R - 1$ RUSs. It costs 12 ms for an RSU in this phase. Here, we assume that each block includes 16 carpooling records.

We compare the computational cost of FICA with existing schemes: AMA [20] and three subschemes DAP-DAD, DAP-ORD, and DAP-EAD in [22]. Since the three subschemes in [22] do not have a driver in their system model, we choose to deem the primary passenger as the driver instead because they have same functions. The matching computation of AMA is not included because they directly used plaintext information to complete matching.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Fig. 4. Performance analysis. (a) Time cost in passengers' carpooling requesting. (b) Time cost in drivers' carpooling responding. (c) Time cost in carpooling verifying. (d) Time cost in carpooling matching. (e) Matching efficiency. (f) Time cost in user tracking. (g) Communication overhead for passengers. (h) Communication overhead for drivers. (i) Communication overhead for RSUs.

From comparison results Fig. 4(a)–(c), we can see that FICA outperforms other schemes in passengers' carpooling requesting, drivers' carpooling responding, and carpooling record verifying.

We analyze the carpooling matching phases in two cases: 1) an RSU matches one passenger with multiple drivers and 2) an RSU matches multiple passengers with multiple drivers. Given the comparison result of the former one shown in Fig. 4(d), FICA has a relatively low time cost. In the latter one, the number of passengers (drivers) is from 50 to 1000, and the result in Fig. 4(e) shows that the maximum time in matching them is less than 14 s when there are 1000 passengers and 1000 drivers.

Finally, we compare the time cost in user tracking and Fig. 4(f) shows the advantage of FICA in this phase. We do not include ORide here because ORide achieves fast user tracking just by searching through service provider's database, however, this is a direct privacy violation to users.

B. Communication Overhead in Carpooling

The binary length $|g| = 160$ bits and we use SHA256 for hash function with $|H| = 256$ bits. To report an encrypted carpooling request, each passenger has to send $\{\text{REQ}, \text{Cert}, \sigma_i\}$ to an RSU where the message length is $|\text{REQ}_i| = (10+12+12+1000+500+500+500+10+1020)+1020+3796$ bits $= 0.9$ kbytes. Each driver has to send $\{\text{RES}, \text{Cert}, \sigma\}$ to an RSU where the message length is $(10+320+3*320+256+256*4*4)+1020+3796$ bits $= 1.27$ kbytes. The comparison results in Fig. 4(g) and (h) indicate that FICA has a moderate communication overhead.

In matching a passenger drivers, each RSU $\text{ID}_{\hat{R}}$ has to broadcast an SYN, send a $\{res\}_{i=1}^{\hat{n}_2}$ of length $12+12+1000+1024+5*1024+1024$ bits $= 1$ kbytes. Since the other

TABLE IV
PERFORMANCE ANALYSIS WITH THE PRIVATE BLOCKCHAIN

Setting	#Tr	#R	#D	Average Setup (s)	Average Request (s)	Average Response (s)	Average Match (s)	Average Update (s)
1	5	10	10	0.012	0.794	0.218	2.646	0.008
2	10	10	10	0.012	1.158	0.289	3.314	0.017
3	15	10	10	0.012	1.282	0.318	3.626	0.026
4	5	20	20	0.016	0.936	0.146	2.481	0.008
5	10	20	20	0.016	1.771	0.261	4.458	0.017
6	15	20	20	0.016	1.782	0.263	4.773	0.024

#Tr.: Number of transactions per block; #P.: Number of passengers per RSU; #D.: Number of drivers per RSU.

schemes do not have RSU in their models, we evaluate the communication overhead for RSU in Fig. 4(i).

C. Experiments on Private Blockchain

Now we conduct experiments on the construction of the private blockchain. Specifically, we consider six settings. For instance, in setting 1, there are ten passengers and ten drivers within each RSU and the number of RSUs is 10; we assume that all blocks except the genesis block contains five carpooling records, the generation of carpooling request and response is random (i.e., 0 to 10 requests per second and 0 to 10 response per second). We create ten threads for the ten RSUs and each of them manages its own passengers and drivers. For simplicity, we do not consider the possibility that a passenger's request is not matched with any response from drivers.

The experimental results are recorded in Table IV. Specifically, we calculate the average time costs in five steps including Setup, Request, Response, Match, and Update. In Setup, the TA has to generate secret keys and public keys for itself, RSUs and users, and the average time cost is 16 ms when there are 10 RSUs, 20 passengers, and 20 drivers. In Request, the passengers generate carpooling requests and the average time cost per RSU is less than 2 s. In Response, the drivers generate carpooling responses and the average time cost per RSU is less than 0.4 s. In Match, each RSU matched its passengers with drivers and the average time cost per passenger is around 0.3 s. In Update, the selected RSU packs carpooling records and creates a new block to put on the private chain, and the average time cost is less than 25 ms. In summary, the computational costs caused by the introduction of our blockchain is acceptable, however, the blockchain provides a capability of auditing carpooling records.

VIII. CONCLUSION

In this paper, we have proposed FICA, an FICA scheme using blockchain-assisted vehicular fog computing, which supports conditional privacy, one-to-many matching, destination matching, and data auditability. This scheme is secure under a threat model where the cloud server and RSUs are semi-honest and users may upload false locations. FICA can protect users' privacy from curious cloud server and RSUs in a conditional way. We used an anonymous authentication scheme to authenticate users and recover malicious users' real identities, and we adopted a private proximity test with location tags to achieve one-to-many matching and extend it to establish a unique secret key between a passenger and a driver, then we achieved get-off location matching by leveraging a range query technique. We also built a private blockchain into our carpooling system to record carpooling records in a verifiable ledger to provide data auditability.

For the future work, we will consider that the RSUs can be compromised by adversaries and design an efficient and privacy-preserving carpooling scheme to support data verification. Although the blockchain provides a tamper-proof ledger, still we need to authenticate data feed [46] from users in case a dispute happens, which means that the claims and proofs from users should be verified by an authentication bureau or at least some honest witnesses through crowdsourcing [47]. The public blockchain technique is maturing with many innovative designs and applications in IoT, still we need to consider the need for a public blockchain in combination with carpooling. For example, what data should be stored on the public blockchain and how to achieve anonymous payment between passengers with drivers. Last, the consortium blockchain may provide a possible solution if different carpooling companies work together to serve users and user privacy will also be considered here as well as company privacy.

REFERENCES

[1] D. Zhang, T. He, Y. Liu, S. Lin, and J. A. Stankovic, "A carpooling recommendation system for taxicab services," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 254–266, Sep. 2014.

[2] I. B.-A. Hartman *et al.*, "Theory and practice in large carpooling problems," *Procedia Comput. Sci.*, vol. 32, no. 1, pp. 339–347, Jun. 2014.

[3] B. Caulfield, "Estimating the environmental benefits of ride-sharing: A case study of Dublin," *Transp. Res. D*, vol. 14, no. 7, pp. 527–531, 2009.

[4] *uberPOOL*. Accessed: Sep. 13, 2018. [Online]. Available: https://www.uber.com/en-SG/drive/singapore/resources/uberpool

[5] *Let's Go. Get a Link to Download the App*. Accessed: Sep. 13, 2018. [Online]. Available: https://www.lyft.com/line

[6] M. Li, L. Zhu, Z. Zhang, and R. Xu, "Achieving differential privacy of trajectory data publishing in participatory sensing," *Inf. Sci.*, vols. 400–401, pp. 1–13, Aug. 2017.

[7] L. Zhu, M. Li, Z. Zhang, and Q. Zhan, "ASAP: An anonymous smart-parking and payment scheme in vehicular networks," *IEEE Trans. Depend. Secure Comput.*, to be published.

[8] Z. Zhang, Z. Qin, L. Zhu, J. Weng, and K. Ren, "Cost-friendly differential privacy for smart meters: Exploiting the dual roles of the noise," *IEEE Trans. Smart Grid*, vol. 8, no. 2, pp. 619–626, Mar. 2017.

[9] (2017). *China's Didi Chuxing Raises $4B More for AI, International Expansion and Electronic Vehicles*. Accessed: Sep. 13, 2018. [Online]. Available: https://techcrunch.com/2017/12/20/chinas-didi-chuxing-raises-4b/

[10] F. Bonomi, R. A. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[11] 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.

[12] S. Basudan, X. Lin, and K. Sankaranarayanan, "A privacy-preserving vehicular crowdsensing based road surface condition monitoring system using fog computing," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 772–782, Jun. 2017.

[13] K. Gai, M. Qiu, Z. Ming, H. Zhao, and L. Qiu, "Spoofing-jamming attack strategy using optimal power distributions in wireless smart grid networks," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2431–2439, Feb. 2017.

[14] M. Li, F. Wu, G. Chen, L. Zhu, and Z. Zhang, "How to protect query and report privacy without sacrificing service quality in participatory sensing," in *Proc. IEEE Int. Perform. Comput. Commun. Conf.*, Dec. 2015, pp. 1–7.

[15] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for Internet of Things applications: Challenges and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 601–628, 1st Quart., 2018.

[16] (2014). *'God View': Uber Allegedly Stalked Users For Party-Goers' Viewing Pleasure (Updated)*. Accessed: Sep. 13, 2018. [Online]. Available: https://www.forbes.com/sites/kashmirhill/2014/10/03/god-view-uber-allegedly-stalked-users-for-party-goers-viewing-pleasure/#7e6865fa3141

[17] (2015). *Uber: The Big Data Company*. Accessed: Sep. 13, 2018. [Online]. Available: https://www.forbes.com/sites/ronhirson/2015/03/23/uber-the-big-data-company/#59d7974818c7

[18] (2017). *Drivers of Ride-Sharing Services Died After Being Attacked by Passengers*. Accessed: Sep. 13, 2018. [Online]. Available: http://lawyersfavorite.com/criminal-attorney/drivers-ride-sharing-services-died-attacked-passengers

[19] Y. Zheng, M. Li, W. Lou, and Y. T. Hou, "Location based handshake and private proximity test with location tags," *IEEE Trans. Depend. Secure Comput.*, vol. 14, no. 4, pp. 406–419, Jul./Aug. 2017.

[20] J. Ni, K. Zhang, X. Lin, H. Yang, and X. Shen, "AMA: Anonymous mutual authentication with traceability in carpooling systems," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.

[21] W. Tang, K. Zhang, J. Ren, Y. Zhang, and X. Shen, "Flexible and efficient authenticated key agreement scheme for BANs based on physiological features," *IEEE Trans. Mobile Comput.*, to be published.

[22] 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.

[23] A. Pham *et al.*, "ORide: A privacy-preserving yet accountable ride-hailing service," in *Proc. 26th USENIX Security Symp.*, 2017, pp. 1235–1252.

[24] 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. Security Privacy*, 2016, pp. 839–858.

[25] F. Gao *et al.*, "A blockchain-based privacy-preserving payment mechanism for vehicle-to-grid networks," *IEEE Netw.*, to be published.

[26] M. Shen, M. Wei, L. Zhu, and M. Wang, "Classification of encrypted traffic with second-order Markov chains and application attribute bigrams," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1830–1843, Aug. 2017.

[27] D. Liu, H. Li, Y. Yang, and H. Yang, "Achieving multi-authority access control with efficient attribute revocation in smart grid," in *Proc. IEEE Int. Conf. Commun.*, Dec. 2014, pp. 634–639.

[28] R. Lu, X. Lin, T. H. Luan, X. Liang, and X. Shen, "Pseudonym changing at social spots: An effective strategy for location privacy in VANETs," *IEEE Trans. Veh. Technol.*, vol. 61, no. 1, pp. 86–96, Jan. 2012.

[29] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar, "Fast range query processing with strong privacy protection for cloud computing," in *Proc. 40th Int. Conf. Very Large Data Base*, 2014, pp. 1953–1964.

[30] *Hyperledger Whitepaper*. Accessed: Sep. 13, 2018. [Online]. Available: https://www.yumpu.com/xx/document/view/55615753/hyperledger-whitepaper

[31] *The Difference Between Public and Private Blockchain*. Accessed: Sep. 13, 2018. [Online]. Available: https://www.ibm.com/blogs/blockchain/2017/05/the-difference-between-public-and-private-blockchain

[32] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: Secure certificateless public verification for cloud-based cyber-physical-social systems against malicious auditors," *IEEE Trans. Comput. Soc. Syst.*, vol. 2, no. 4, pp. 159–170, Dec. 2015.

[33] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. Int. Cryptol. Conf.*, 2017, pp. 357–388.

[34] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[35] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic *k*-TAA," in *Proc. Int. Conf. Security Cryptography Netw.*, 2006, pp. 111–125.

[36] W. He, X. Liu, and M. Ren, "Location cheating: A security challenge to location-based social network services," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, 2011, pp. 740–749.

[37] R. Lu, X. Lin, H. J. Zhu, P.-H. Ho, and X. Shen, "ECPP: Efficient conditional privacy preservation protocol for secure vehicular communications," in *Proc. IEEE Conf. Comput. Commun.*, 2008, pp. 1903–1911.

[38] J. Shao, X. Lin, R. Lu, and C. Zuo, "A threshold anonymous authentication protocol for VANETs," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1711–1720, Mar. 2016.

[39] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, "Beyond bloom filters: From approximate membership checks to approximate state machines," in *Proc. ACM SIGCOMM Conf. Appl. Technol. Architect. Protocols Comput. Commun.*, vol. 36, 2006, pp. 315–326.

[40] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," in *Proc. Int. Conf. Theory Appl. Cryptograph. Tech. Adv. Cryptol. (EUROCRYPT)*, 2004, pp. 523–540.

[41] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *Proc. IEEE Symp. Security Privacy*, May 2013, pp. 397–411.

[42] 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.*, to be published.

[43] D. Chaum, "Blind signatures for untraceable payments," in *Proc. Adv. Cryptography*, 1983, pp. 199–203.

[44] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *Proc. 10th Workshop Mobile Comput. Syst. Appl.*, 2009, p. 3.

[45] M. Scott. *MIRACL: Multi-Precision Integer and Rational Arithmetic C/C++ Library*. Accessed: Sep. 13, 2018. [Online]. Available: http://www.certivox.com/miracl

[46] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town Crier: An authenticated data feed for smart contracts," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2016, pp. 270–282.

[47] L. P. Cox, "Truth in crowdsourcing," *IEEE Security Privacy*, vol. 9, no. 5, pp. 74–76, Sep./Oct. 2011.

Meng Li (S'15–GS'15) is currently pursuing the Ph.D. degree at the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China.

He is a Visiting Ph.D. Student with Wilfrid Laurier University, Waterloo, ON, Canada. He was sponsored by the China Scholarship Council in 2017. His current research interests include applied cryptography, security and privacy, VANET, fog computing, and blockchain.

Mr. Li was a recipient of the National Graduate Student Scholarship in 2011.

Liehuang Zhu (M'11) is a Professor with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. He has been selected into the Program for New Century Excellent Talents at the University from Ministry of Education, China. His current research interests include cryptographic algorithms and secure protocols, Internet of Things security, cloud computing security, big data privacy, mobile and Internet security, and trusted computing.

Xiaodong Lin (M'09–SM'12–F'17) received the Ph.D. degree in information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada.

He was an Associate Professor of information security with the Faculty of Business and Information Technology, University of Ontario Institute of Technology, Oshawa, ON, Canada. He is currently an Associate Professor with the Department of Physics and Computer Science, Wilfrid Laurier University, Waterloo. His current research interests include wireless network security, applied cryptography, computer forensics, software security, and wireless networking and mobile computing.

Dr. Lin was a recipient of the Outstanding Achievement in Graduate Studies Award from the University of Waterloo.