Delft University of Technology

Mnemosyne

Privacy-Preserving Ride Matching With Collusion-Resistant Driver Exclusion

Li, Meng; Gao, Jianbo; Zhu, Liehuang; Zhang, Zijian; Lal, Chhagan; Conti, Mauro; Alazab, Mamoun

DOI
10.1109/TVT.2022.3225175

Publication date
2022

Document Version
Final published version

Published in
IEEE Transactions on Vehicular Technology

Citation (APA)
Li, M., Gao, J., Zhu, L., Zhang, Z., Lal, C., Conti, M., & Alazab, M. (2022). Mnemosyne: Privacy-Preserving Ride Matching With Collusion-Resistant Driver Exclusion. *IEEE Transactions on Vehicular Technology*, *72*(4), 5139-5151. https://doi.org/10.1109/TVT.2022.3225175

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

https://www.openaccess.nl/en/you-share-we-take-care

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

Mnemosyne: Privacy-Preserving Ride Matching With Collusion-Resistant Driver Exclusion

Meng Li, *Senior Member, IEEE*, Jianbo Gao, *Student Member, IEEE*, Zijian Zhang, *Member, IEEE*, Liehuang Zhu, *Senior Member, IEEE*, Chhagan Lal, Mauro Conti, *Fellow, IEEE*, and Mamoun Alazab, *Senior Member, IEEE*

***Abstract*—Ride-Hailing Service (RHS) has drawn plenty of attention as it provides transportation convenience for riders and financial incentives for drivers. Despite these benefits, riders risk the exposure of sensitive location data during ride requesting to an untrusted Ride-Hailing Service Provider (RHSP). Our motivation arises from *repetitive matching, i.e., the same driver is repetitively assigned to the same rider*. Meanwhile, we introduce a driver exclusion function to protect riders' location privacy. Existing work on privacy-preserving RHS overlooks this function. While Secure k Nearest Neighbor (SkNN) facilitates efficient matching, the state-of-the-art neglects a collusion attack. To solve this problem, we formally define repetitive matching and strong location privacy, and propose Mnemosyne: privacy-preserving ride matching with collusion-resistant driver exclusion. We extend the simple integration of equality checking and item exclusion to a dynamic integration. We concatenate each prefix of an acceptable identity range to each location code when generating a ride request, i.e., secure mix index. We process each prefix of the driver identity to generate a ride response, i.e., a mix token. We build an indistinguishable Bloom-filter as an index to query the token. When matching riders with drivers, the colluding parties cannot distinguish identity prefixes from location codes. We build a prototype of Mnemosyne based on servers, smartphones, and a real-world dataset. Experimental results demonstrate that Mnemosyne outperforms existing work regarding strong location privacy and computational costs.**

***Index Terms*—Ride-hailing service, repetitive matching, privacy, driver exclusion, collusion attack.**

Manuscript received 24 May 2022; revised 24 August 2022 and 15 September 2022; accepted 24 November 2022. Date of publication 28 November 2022; date of current version 18 April 2023. The work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62002094, in part by Anhui Provincial Natural Science Foundation under Grant 2008085MF196 in part by the National Key Research and Development Program of China under Grant 2021YFB2701200, in part by theNational Natural Science Foundation of China (NSFC) under Grants 62172040, U1836212, and 61872041, and in part by EU LOCARD Project under Grant H2020-SU-SEC-2018-832735. The review of this article was coordinated by Prof. Xiaojiang Du. *(Corresponding authors: Zijian Zhang; Liehuang Zhu.)*

Meng Li and Jianbo Gao are with the Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, School of Computer Science and Information Engineering, Hefei University of Technology, Anhui Province Key Laboratory of Industry Safety and Emergency Technology, and Intelligent Interconnected Systems Laboratory of Anhui Province (Hefei University of Technology), Hefei, Anhui 230601, China (e-mail: mengli@hfut.edu.cn; jianbogao@mail.hfut.edu.cn).

Zijian Zhang and Liehuang Zhu are with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China, and also with the Southeast Institute of Information Technology, Beijing Institute of Technology, Fujian 351100, China (e-mail: zhangzijian@bit.edu.cn).

Chhagan Lal is with the Delft University of Technology, 2628 CD Delft, Netherlands (e-mail: c.lal@tudelft.nl).

Mauro Conti is with the Department of Mathematics and HIT Center, University of Padua, 35131 Padua, Italy, and also with the Department of Intelligent Systems, CyberSecurity Group, TU Delft, 35122 Delft, Zuid-Holland, Netherlands (e-mail: conti@math.unipd.it).

Mamoun Alazab is with the College of Engineering, IT and Environment, Charles Darwin University, Casuarina, Northern Territory 0810, Australia (e-mail: alazab.m@ieee.org).

Digital Object Identifier 10.1109/TVT.2022.3225175

I. Introduction

RIDE-Hailing Services (RHSs) have attracted plenty of attention from both academia [1], [2], [3] and industry [4], [5], [6]. A rider sends a ride request to a Ride-Hailing Service Provider (RHSP), which informs drivers nearby and match responding drivers with the rider. Being one of the most popular vehicular services [7], [8], [9], RHSs enable 78 million people to enjoy rides using the Uber app on a monthly basis [10]. For the RHS to keep running, ride matching plays an important role. Firstly, it finds an optimal driver for a rider and saves the rider's waiting time. Secondly, it helps a cruising driver pick up a requesting driver and increases the driver's income. Thirdly, it assists the RHSP to match riders with drivers in different service areas and maintain good system efficiency.

To complete the user matching, riders upload real-time locations to the RHSP. This induces privacy risks [11], [12], [13], [14], [15], [16], [17], [18] since location information is highly related to user activities such as leaving home, dining in an Italian restaurant, and attending a political gathering. To solve this problem, secure k nearest neighbour (SkNN) [19] is proposed. There are several works [20], [21], [22] proposing secure query processing over encrypted data. They are constructed upon prefix-encoding, prefix-free encoding, Bloom filter, and space encoding to realize strong privacy protection as well as high efficiency.

We observe that, since a rider hails a ride from the same location repetitively, it is possible that the RHSP assigns the same driver to her more than once. It is a collusion attack that is initiated for some secret agreement between the RHSP and the driver who seeks unfair profiting. We call it *repetitive matching*. For example, as depicted in Fig. 1, a rider Alice requests a ride k times near her home in the morning. A driver Bob who lives nearby and starts picking up riders around the same time is assigned to Alice multiple times. The consequences of repetitive matching are quite severe for riders, especially when the matching driver is mischievous or malicious. First, the rider Alice will have a bad user experience if she is once again matched to the

0018-9545 © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

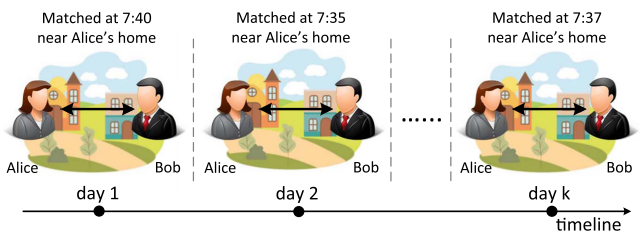


Fig. 1. An example of repetitive matching in RHS.

driver Bob who provided a poor service to Alice before. Second, Bob will acquire Alice's sensitive locations (home, work) by observing the building or community around pick-up/drop-off location with a high probability. Third, a malicious Bob who knows Alice's departure time infers that maybe no one is in Alice's home.

To the best of our knowledge, no existing work tackles the repetitive matching problem in RHSs. In other words, we need to design a privacy-preserving RHS with collusion-resistant driver exclusion. In our previous work, we have proposed one-time, oblivious, and unlinkable query processing over encrypted data [23] to address the repetitive problem in SkNN. However, we did not solve this problem in RHSs that has unique characteristics and we did not consider the collusion attack between the RHSP and driver. The rationale of the collusion attack is to determine whether a target rider in a specific area requests to exclude a previously matched driver in the current ride request. The attack is launched as follows. The RHSP recruits or bribes a group of drivers or to respond to ride requests in the area. If a rider holds an exclusion requirement, it is likely that the rider lives nearby or frequently visits this location. Besides the equality checking for matching the nearest driver, a rider and a driver go through item exclusion, i.e., the rider inserts an identity range into a bloom filter, and the driver converts an ID into a token [23]. When the rider is matched to a colluding driver, the driver will inform the RHSP of which item is used to perform the driver exclusion, thus leaking the exclusion requirement of the rider.

To defend against the collusion attack, there is a technical challenge to be tackled with. **Technical challenge**: how to hide the exclusion requirement from the colluding RHSP and driver when they are the matching executor and participant, respectively. Basically, we need to hide the rider's exclusion requirement before user matching. In other words, the technical challenge is how to secretly mix the exclusion problem with the equality checking problem given that they are handled separately. Only if we manage to confuse the colluding RHSP and driver regarding which item is used for equality checking and which item is used for driver exclusion, can we successfully achieve collusion-resistant driver exclusion.

In this work, we propose Mnemosyne: privacy-preserving ride matching with collusion-resistant driver exclusion. We extend the simple integration of equality checking and item exclusion to a dynamic integration of the two problems. Specifically, we inherit the operations of equality checking in [23]. We prevent the colluding ones from knowing whether a target ride has a need to rule out a driver by designing a mix index and a mix token. A rider concatenates each prefix of an acceptable driver identity range to each code of pick-up location when generating a ride request index. The rider builds an indistinguishable Bloom-filter as the mix index. A driver processes each prefix of driver identity to generate a mix token. When matching riders with drivers, the colluding RHSP searches the index with the token, while not distinguishing the identity prefix from the location code, which achieves collusion-resistant driver exclusion. We frame the key contributions as follows.

- We identify a new problem *repetitive matching* in RHS. We consider a stronger adversary model under which the RHSP colludes with a group of drivers to violate the riders' location privacy. To cope with the attack, we define strong location privacy for riders, i.e., we aim to protect the riders' motivation of driver exclusion at a specific time and a specific location.
- We propose a privacy-preserving ride matching scheme Mnemosyne to achieve collusion-resistant driver exclusion. We use a projection-based method to realize space encoding and leverage prefix-free encoding to process riders' locations. Next, we adopt prefix encoding to handle drivers' identity range and link each prefix of the range to the location codes. We insert the linked codes into an Indistinguishable Bloom Filter (IBF) as a secure mix index. A driver's response includes location and identity which are encrypted into a mix token similarly. We complete user matching by querying drivers' tokens on the IBF.
- We formally analyze the privacy of Mnemosyne. To evaluate the feasibility and efficiency of Mnemosyne, we build a prototype based on two servers, two smartphones, two Android virtual machines, and a real-world dataset. We evaluate its computational costs and communication overhead and compare it with existing RHS schemes.

The remainder of this paper is organized as follows. We review some related work in Section II. Section III introduces some preliminaries. Section IV formalizes the problem. In Section V, we present Mnemosyne in detail, followed by the privacy analysis in Section VI and performance evaluation in VII, respectively. Finally, we provide some discussions in Section VIII and conclude the paper in Section IX.

II. Related Work

In this section, we review some related work on SkNN and privacy-preserving RHS built from similar cryptographic primitives.

A. SkNN

Li et al. [20] presented the first range query processing protocol, which achieved index indistinguishability under the indistinguishability against chosen keyword attack (IND-CKA). A data owner converts each data item dt_i by prefix encoding [24] and organizes each prefix family of encoded item $F(di_i)$ into a PBTree. Then the data owner makes the PBtree privacy-preserving by a keyed hash message authentication code HMAC and Bloom filters [25]. For each prefix pr_i, the data owner computes several hashes $\mathsf{HMAC}(k_j, pr_i)$, and inserts a

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

randomized version $\mathsf{HMAC}(r, \mathsf{HMAC}(k_j, pr_i))$ into a Bloom filter. Each r corresponds to a node and each node relates to a prefix family, i.e., data item. Next, a data user converts a range into a minimum set of prefixes and computes several hashes $\mathsf{HMAC}(k_j, pr_i)$ for each pr_i as a trapdoor. The service provider searches in the PBtree to find a match by using the trapdoor.

Li et al. [21] concerned processing conjunctive queries including keyword conditions and range conditions in a privacy-preserving way and presented a privacy-preserving conjunctive query processing protocol supporting adaptive security, efficient query processing, and scalable index size at the same time. Specifically, they adopt prefix encoding in their earlier work [20] and design an indistinguishable Bloom filter (IBF), i.e., twin Bloom filter, to replace the previous structure. A pseudo-random hash function H determines a cell location $H(h_{k+1}(h_j(w_i)) \oplus r)$, i.e., which twin cell stores '1'. Instead of using PBTree, they construct an IBTree as the secure index.

Lei et al. [22] presented a secure and efficient query processing protocol SecEQP. They leveraged some primitive projection functions to convert the neighbor regions of a given location. Given the codes of two converted locations, the service provider computes the proximity of the two locations by judging whether the two codes are the same. This is an improvement over their previous work [21] since the two-dimensional location data is projected to high-dimensional data which expands the location space to make the converted location more secure. The data owner further embeds the codes into a similar IBFTree in order to build a secure index. The data user computes similar trapdoors by a keyed hash message authentication code. The final secure query processing is the same as [21].

B. Privacy-Preserving RHS

Pham et al. [1] proposed a privacy-preserving RHS ORide to match riders with drivers without leaking users' identities or locations. ORide leveraged homomorphic encryption and optimizations for ciphertext packing and transformed processing. Li et al. [2] proposed a privacy-preserving RHS FICA to efficiently match users. It utilizes Road-Side Units (RSUs) as fog nodes to locally match riders and drivers by using anonymous authentication, private proximity test, and private range query. Li et al. [3] proposed a privacy-preserving collaborative RHS CoRide based on a consortium blockchain. Several RHS providers co-maintain a blockchain to form rides between riders and drivers from different platforms while not violating their privacy. Yu et al. [26] used road network embedding and homomorphic encryption to efficiently and securely compute the shortest distances between users. Luo et al. [8] proposed efficient computation of shortest road distance over ciphertexts based on road network embedding and a modified Paillier encryption system. Xie et al. [27] presented a ride-hailing matching protocol to enable private distance computation based on road network embedding and property-preserving hash without a trusted third party. Li et al. [28] proposed a ride-hailing matching scheme to achieve anonymous and collusion-resistant pairing by using SGX-protected smart contracts.

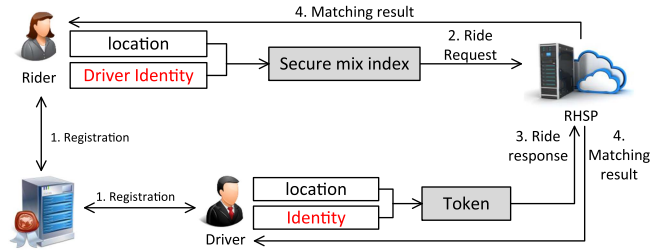


Fig. 2. System model of Mnemosyne.

The promotion over existing work in this paper is that Mnemosyne supports the riders' need to exclude a driver(s) privately under collusion attacks while guaranteeing anonymity, location privacy, and unlinkability.

III. Problem Statement

In this section, we introduce the system model, security model, and design objectives. Specifically, we give the formal definition of repetitive matching and strong location privacy under repetitive matching.

A. System Model

The system model consists of rider, driver, RHSP, and certificate authority (CA) as shown in Fig. 2.

Rider is a user who requests a ride by sending a ride request (including a pick-up location and a drop-off location) via a smartphone app to the RHSP and waiting for a matching result. After a driver is assigned to the rider, the rider gets on the driver's vehicle, and pays the driver a ride fare when arriving at the drop-off location. We formally define the repetitive matching as follows.

Definition 1 (Repetitive matching): The repetitive matching event is a single location-time predicate or a combination of location-time predicates linked by the Boolean operators [29]. Let $RO = (R, CL, DL, D, t)$ be a ride order and they are a quintuple of rider identity, current location, drop-off location, driver identity, and timestamp. Let $(R_i, CL_{ij}, DL_{ij}, D_{ij}, t_{ij})$ be the jth ride order of rider R_i. A repetitive matching event, denoted by ReMatch, is expressed as $(R_{ij} = R_{ik}) \wedge (CL_{ij} = CL_{ik}) \wedge (t_{ij} \neq t_{ik})$, where $RO_{ij} = (R_{ij}, CL_{ij}, DL_{ij}, D_{ij}, t_{ij})$ is a ride order among all ride orders $\{RO_{ij}\}$ of the rider R_i.

Driver is a user who owns a vehicle and offers a ride by sending a ride response (including a current location) via a smartphone app or a On-Board Unit (OBU) to the RHSP and waiting for a matching result. The pick-up area is assumed to be a set of locations. After a rider is assigned to the driver, the driver takes the rider from the pick-up location, drives toward the drop-off location, and requests a ride fare from the rider when the rider gets off the vehicle. The connection between users and the RHSP are 4 G and 5 G communications [30], [31], [32], [33].

RHSP is a service provider building a ride matching platform to provide riders with ride services from drivers. It charges drivers an amount of service fees when a ride is complete. The main task of RHSP is to match riders with drivers. User matching is related to system efficiency and company profit.

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

CA is an authority responsible for system initialization and user registration. Before sending the first ride request/response, the rider/driver obtains public parameters, a set of certificates [1], and secret keys from the CA.

B. Security Model

The threats mainly come from internal adversaries [34], [35], [36], [37], [38], [39], [40]. Specifically, we consider the collusion attack from the RHSP and a group of drivers. We do not consider physical tracking of a user, e.g., taking photos by a smartphone camera.

Rider. A rider is an honest-but-curious entity that follows the predefined protocol but is curious about the pick-up/drop-off locations of other riders.

Driver. Most drivers are honest-but-curious. A small group of drivers is malicious and they try to reveal the pick-up/drop-off locations of riders matched with other drivers. They also collude with the RHSP to acquire whether a rider has the need to exclude some driver.

RHSP. The RHSP may recruit a group of drivers and ask them to take riders in a certain area. Its malicious agenda is to probe into riders' location privacy by checking whether they need to exclude their previously matched drivers. The RHSP is malicious mainly because of a malicious employee who sells the locations of riders to an advertiser or even black market.

CA. The CA is fully trusted [41], [42]. It is assumed not to be breached by any adversary. The assumption of a secure CA or TTP, as a common practice, is widely acknowledged in privacy-preserving vehicular schemes, such as: Trusted Authority in NRS/TRS [43], Crypto Provider in pRide [8], Authority in lpRide [26], Trusted Authority in PAM [9], and Trusted Server in W^3-$tess$ [45].

C. Design Objectives

Basic location privacy. The pick-up/drop-off location of riders and the pick-up area of honest drivers should be protected from the RHSP. The two objectives correspond to index privacy and token privacy.

Strong location privacy. The colluding parties cannot know whether a rider excludes a driver in her/his ride request, i.e., the motivation behind driver exclusion at a specific location is protected. Furthermore, the colluding parties cannot know whether a rider excludes the same driver. Formally, we give the definition of strong location privacy as follows.

Definition 2 (Strong location privacy): Given a rider R_i and an adversary $\mathcal{A}$ that is assumed to be the RHSP or the driver D matched to R_i, we say that a ride matching scheme π achieves strong location privacy if the following two inequations hold:

(1) The difference between two probabilities that $\mathcal{A}$ observes the execution of π to distinguish whether R_i excludes a driver D from N_D drivers is negligible:

$$\begin{aligned}&|\Pr[\mathcal{A}(\pi(N_R, N_D, (R_i, CL_{ij}, DL_{ij}, \emptyset;, t_{ij}))) = 1]\\&\quad - \Pr[\mathcal{A}(\pi(N_R, N_D, (R_i, CL_{ij}, DL_{ij}, D, t_{ij})))] = 1|\\&\leq \text{negl}(N_D),\end{aligned} \tag{1}$$

TABLE I
KEY NOTATIONS OF MNEMOSYNE

Notation	Definition
RHSP	Ride-hailing service provider
CA, R, D	Certificate authority, rider, driver
f, t	Projection functions, number of f
g, h	AND composition, OR-composition
a, b, d	Parameters of a primitive projection function
IBF, $\mathcal{B}$	Indistinguishable Bloom Filter
H_i, H; m	Hash function in IBF, number of twins in IBF
$k_1, \cdots, k_{u+1}$	Secret key of users
pk, sk	Public key, private key of RHSP
PL,	Pick-up location, drop-off location
$\mathcal{RQ}$, $\mathcal{RP}$	Ride request, ride response
fl, fa, str, $\mathcal{Q}$	Feasible location, feasible area, string, string set
n	Identity of the driver to be excluded
N_R, N_D	Number of drivers, number of drivers
num	Number of drivers to be excluded
$\mathcal{S}$	Minimum set of prefixes
$\mathcal{Q}'$, c	String set after concatenation, code in $\mathcal{Q}'$
p	Prefix obtained from prefix encoding
C, σ	Ciphertext, signature
CL, $\mathcal{F}$	Current location, prefix family
au, $\mathcal{MR}$	Authentication code, matching result
Sim, $\mathcal{A}$	Simulator, adversdary

where $\Pr[\mathcal{A}(\pi(N_R, N_D, (R_i, CL_{ij}, DL_{ij}, D, t_{ij}))) = 1]$ refers to the probability that $\mathcal{A}$, a distinguisher, guesses that driver D is assigned to rider R_i, and the probability is taken over uniform choice of R_i, CL, DL, and D. For simplicity, we use $|\Pr[\mathcal{A}(\pi(\emptyset;)) = 1] - \Pr[\mathcal{A}(\pi(D))] = 1| \leq \text{negl}$ to stand for inequation (1).

(2) The difference between two probabilities that $\mathcal{A}$ observes the execution of π to distinguish whether R_i excludes two drivers D, D' is negligible:

$$\begin{aligned}&|\Pr[\mathcal{A}(\pi(N_R, N_D, (R_i, CL_{ij}, DL_{ij}, D, t_{ij}))) = 1]\\&\quad - \Pr[\mathcal{A}(\pi(N_R, N_D, (R_i, CL_{ij}, DL_{ij}, D,' t_{ij}))) = 1]|\\&\leq \text{negl}(N_D).\end{aligned} \tag{2}$$

D and D' can be two different drivers or the same driver. For simplicity, we use $|\Pr[\mathcal{A}(\pi(D)) = 1] - \Pr[\mathcal{A}(\pi(D'))] = 1| \leq \text{negl}$ to stand for (2).

Anonymity. The identity of users should be protected. When a rider is requesting a ride or an honest driver is responding to a ride request, the RHSP cannot identify the user's identity.

Unlinkability. The RHSP cannot link two ride requests/responses from a rider/honest driver.

Efficiency. The proposed scheme Mnemosyne should be lightweight regarding computational costs and communication overhead. Specifically, we should consider the effect of different number of drivers to be excluded.

IV. PRELIMINARIES

In this section, we revisit some preliminaries that lays the foundation for the proposed Mnemosyne, namely space encoding [22], prefix encoding [20], [24], and IBF [21], [22].

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

A. Space Encoding

Space encoding leverages a composition of several primitive projection functions to encode a finite region. It enables proximity test between two locations by using projected codes.

Given a location $\vec{l}$ in the two-dimensional space, we first transforms it into an integer by using a primitive projection function $f: \mathbb{R}^2 \rightarrow \mathbb{Z}$. A feasible region of $\vec{l}$ regarding f is defined as including all locations $\vec{l'}$ such that $f(\vec{l}) = f(\vec{l'})$. There are two compositions for projection functions: AND-composition and OR-composition. Given t projection functions $f_1, f_2, \cdots f_t$. An AND-composition is denoted as $g = \text{AND}(f_1, f_2, \cdots, f_t)$. Given two locations $\vec{a}$ and $\vec{b}$, $g(\vec{a}) = g(\vec{b})$ only if $f_i(\vec{a}) = f_i(\vec{b})$ for all $1 \leq i \leq u$.

An OR-composition is denoted as $\text{OR}(f_1, f_2, \cdots, f_t)$. Given two locations $\vec{a}$ and $\vec{b}$, $f(\vec{a}) = f(\vec{b})$ if and only if $f_i(\vec{a}) = f_i(\vec{b})$ for at least one $i \in [1, v]$. Space encoding a location $\vec{l}$ by AND-composition $g = \text{AND}(f_1, f_2)$ produces a feasible region of $\vec{l}$, i.e., an intersection region. Space encoding a location $\vec{l}$ by AND-composition and then OR-composition $h = \text{OR}(g_1, g_2)$ produces a union of two feasible regions where g_1 and g_2 are two AND-compositions.

B. Prefix Encoding

Prefix encoding [24] transforms numbers and ranges to a special representation and checks whether a number belongs to a range by searching common elements in two groups. It includes two sets of operations. The first one is for processing numbers and the other one is for ranges. Given a number a of w bits with a binary format being $a_1 a_2 \ldots a_w$, its prefix family $F(a)$ is the group of $w+1$ prefixes $\{a_1 a_2 \cdots a_l, a_1 a_2 \cdots a_{w-1}*, \cdots, a_1 * \cdots *, * * \cdots *\}$. For instance, the prefix family of number 9 of 5 bits is $F(9) = \{01001, 0100*, 010**, 01***, 0****, *****\}$. Given a range $[X, Y]$, we transform $[X, Y]$ to a minimum set of prefixes $M([X, Y])$ satisfying the condition that the union of the prefixes is the same as $[X, Y]$. For instance, $M([0, 16]) = 0****, 10000$. For a number a and a range $[X, Y]$, $a \in [X, Y]$ if and only if there is a prefix $pr \in M([X, Y])$ so that $a \in pr$ holds. For a number a and a prefix pr, $a \in pr$ if and only if $pr \in F(a)$. Hence, for a number a and range $[X, Y]$, $a \in [X, Y]$ if and only if $F(a) \cap M([X, Y]) \neq \emptyset$.

C. IBF

An IBF is an array $\mathcal{B}$ of m cell twins, u pseudo random hash functions $H_1, H_2, \cdots, H_u$, and a random oracle H. Each cell twin has two cells and each cell stores either '0' or '1'. H is used to determine which cell stores '1'. In initialization, the chosen cell is set to 0 and the unchosen cell is set to 1. A keyword w is hashed to u twin cells $\mathcal{B}[H_1(w)], \mathcal{B}[H_2(w)], \cdots, \mathcal{B}[H_u(w)]$. The u chosen cells are set to '1' and the unchosen cells are set to '0'. An IBF with m cell twins has m '1's and the chosen cell is randomly picked. Thus, any polynomial-time adversary $\mathcal{A}$ distinguishes the chosen cell from the other cell correctly with a negligible probability over $1/2$. Furthermore, the IBF achieves adaptive security.

Algorithm 1: Mnemosyne.

Input: R, PL, DL, D, RHSP
Output: pp, $\mathcal{SK}$, $\mathcal{CT}$, (pk, sk), $\mathcal{RQ}$, $\mathcal{RP}$.
1: **/*System Initialization*/**
2: CA generates public parameters pp and secret keys $\mathcal{SK}$ given security parameters 1^k;
3: **/*Entity Registration*/**
4: Rider R obtains pp, $\mathcal{SK}$, and a set of certificates $\mathcal{CT}$;
5: Driver D obtains pp, $\mathcal{SK}$, and $\mathcal{CT}$;
6: RHSP obtains pp and a public/private key pair (pk, sk);
7: **/*Ride Request*/**
8: R sends a ride request $\mathcal{RQ}$ to the RHSP;
9: **/*Ride Response*/**
10: The RHSP verifies $\mathcal{RQ}$ and broadcasts a ride requesting task;
11: Driver D sends a ride response $\mathcal{RP}$ to the RHSP;
12: The RHSP verifies a set of $\{\mathcal{RP}\}$ and matched $\mathcal{RQ}$ with $\{\mathcal{RP}\}$;
13: The RHSP returns a matching result $\mathcal{MR}$ to R and D;
14: **/*Ride Initiation and Completion*/**
15: D picks up R at the pick-up location PL;
16: R pays a ride fare to D at the drop-off location DL.
17: Return (pp, $\mathcal{SK}$, $\mathcal{CT}$, (pk, sk), $\mathcal{RQ}$, $\mathcal{RP}$, $\mathcal{MR}$).

V. The Proposed Scheme

At a high level, Mnemosyne contains five phases: system initialization, entity registration, ride request, ride response, and ride initiation and completion. Before we dive into the details of the above five phases, we first give an overview of the Mnemosyne by using Algorithm 1. It includes input and output parameters and the different steps that are carried out during the execution of Mnemosyne.

A. System Initialization

For the initialization of space encoding. The CA chooses t projection functions $f_1, f_2, \cdots f_t$, where $f_i(\vec{l}) = \lfloor \frac{\vec{a}_i \cdot \vec{l} + b_i}{d_i} \rfloor$, $\vec{a} = (\theta, 1)$, $\theta \in [0, 2\pi]$, $b \in [0, d]$, and d is the interval length. The t vectors $\{\vec{a}_i\}$ divide π equally. Define $g_{i,j}$ as an AND-composition of f_i, f_j denoted by $g_{i,j} = AND(f_i, f_j)$. Define $h_{i,j}$ as an OR-composition of two AND-composition g_i, g_j denoted as $h_{i,j} = OR(g_i, g_j)$. For the initialization of prefix encoding. The CA sets a length w of user identity in its binary presentation. For the initialization of IBF. The CA creates an array $\mathcal{B}$ of m twins, $u+1$ secret keys $\{k_1, k_2, \ldots, k_{u+1}\}$, and constructs u pseudo-random hash functions $H_1, H_2, \cdots H_u$ using the keyed-hash message authentication code HMAC, where $H_i = \text{HMAC}_{k_i}(\cdot)$, a pseudo-random hash function $H_{u+1}(.) = \text{HMAC}_{k_{u+1}}(\cdot)$, and a hash function H. Finally, the CA publishes public parameters $pp = (f_1, f_2, \cdots f_t, w, \mathcal{B}, H_1, H_2, \cdots H_u, H)$.

B. Entity Registration

For each period of time, e.g., one week, the CA updates $u+1$ secret keys $\mathcal{SK} = (k_1, k_2, \cdots k_{u+1})$. A rider R_i with an identity

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

being a smartphone number registers to the CA. The CA generates a set of new certificates $\mathcal{CT}_i = \{ct_{i1}, ct_{i2}, \cdots ct_{iv}\}$ and returns $(pp, \mathcal{CT}_i, \mathcal{SK})$ to R_i. A driver D_j with an identity being a license plate registers to the CA and obtains $(pp, \mathcal{CT}_j, \mathcal{SK})$. The certificates are deemed to be pseudonyms of users. The RHSP registers to the CA and obtains a public/private key pair (pk, sk).

C. Ride Request

A rider R_i is standing at a pick-up location PL_i and wishing to go to a drop-off location DL_i. R_i generates a ride request, i.e., secure mix index $\mathcal{RQ}_i$, as follows.

- R_i chooses two projection functions f_{i1}, f_{i2} and computes the first feasible location:

$$fl_{i1} = \text{AND}(f_{i1}(\overrightarrow{PL_i}), f_{i2}(\overrightarrow{PL_i})). \tag{3}$$

  We assume that a rider has to choose at least four projection functions in order to form a feasible location. Here, $\overrightarrow{PL_i}$ is a vector representation of PL_i.
- R_i computes another three feasible location $fa_{i2}, fa_{i3}, fa_{i4}$ with different projection functions similarly, and computes a feasible area:

$$fa_i = \text{OR}(fl_{i1}, fl_{i2}, fa_{i3}, fa_{i4}). \tag{4}$$

- R_i converts each feasible location fl into a string str. For example, fl_{i1} is converted as:

$$str_{i1} = f_{i1}(\overrightarrow{PL_i})||f_{i2}(\overrightarrow{PL_i})||f_{i3}(\overrightarrow{PL_i})||f_{i4}(\overrightarrow{PL_i}). \tag{5}$$

- R_i converts the obtained strings into a string set $\mathcal{Q}_i$ by using prefix-free encoding [22]. For example, fa_i is converted as:

$$\mathcal{Q}_i = \{00||str_{i1}, 01||str_{i2}, 10||str_{i2}, 11||str_{i4}\}. \tag{6}$$

- R_i retrieves the identity of a previously matched driver n and converts two ranges $[1, n-1] \cup [n+1, N]$ into a minimum set of prefixes $\mathcal{S}_i = \{p_i\}$ by using prefix encoding. Here, N is the number of drivers. (We note that if R_i has to exclude $num > 1$ drivers, there will be num break points in the range $[1, N]$. Each segment has its own minimum set of prefixes and the disjunction of all sets produces $\mathcal{S}$.)
- R_i concatenates each prefix in $\mathcal{S}_i$ to each code in $\mathcal{Q}_i$ and forms a new set $\mathcal{Q}'_i$. For example, if $n = 9$ and $N = 16$, then $\mathcal{S}_i = \{00**, 01000, 0101*, 011**, 10000\}$ and $\mathcal{Q}'_i = \{00||str_{i1}||00**, 01||str_{i2}||00**, 10||str_{i3}||00**, 11||str_{i4}||00**, \cdots, 11||str_{i4}||10000\}$. In this way, R_i successfully mixes an acceptable drivers' identity range with the pick-up location.
- R_i embeds each code c_i in $\mathcal{Q}'_i$ and a random number r_i into $\mathcal{B}_i$ by setting for all $i \in [1, |\mathcal{Q}'_i|]$ and $j \in [1, u]$:

$$\mathcal{B}_i[H(H_{k_{u+1}}(H_j(c_i)) \oplus r_n)][H_j(c_i)] = 1, \tag{7}$$

$$\mathcal{B}_i[1 - H(H_{k_{u+1}}(H_j(c_i)) \oplus r_n)][H_j(c_i)] = 0. \tag{8}$$

Finally, R_i encrypts the secure mix index $\mathcal{RQ}_i = (\mathcal{B}_i, r_n)$ by using asymmetric encryption Enc and pk. R_i sends the ciphertext C_i with a signature σ_i, and a one-time certificate ct_i selected from $\mathcal{CT}_i$ to the RHSP.

D. Ride Response

An available driver D_j is near a current location CL_j and generates a ride response, i.e., mix token $\mathcal{RP}_j$, as follows.

- D_j computes a similar feasible area fa_j to transform her/his current location.
- D_j converts each feasible location into a string and computes a similar string set $\mathcal{Q}_j$.
- D_j converts her/his identity d into a prefix family $\mathcal{F}_j$.
- D_j concatenates each prefix in $\mathcal{F}_j$ to each code in $\mathcal{Q}_j$ and forms a new set $\mathcal{Q}'_j$. By doing so, D_j successfully mixes the current location with identity.
- D_j computes a mix token $\mathcal{RP}_j = \{H_{k_{u+1}}(H_j(c_i)), H_j(c_i)\}$ for all $i \in [1, |\mathcal{Q}'_j|]$ and $j \in [1, u]$:

Finally, D_j encrypts $\mathcal{RP}_j$ by using asymmetric encryption and pk. D_j sends the ciphertext C_j with a signature σ_i, and one-time certificate ct_j to the RHSP. We draw the process of ride request and ride response in Fig. 3.

On receiving the secure mix index $\mathcal{RQ}_i$ from R_i and mix token $\mathcal{RP}_j$ from D_j, the RHSP checks their validity and performs user matching as follows.

- RHSP authenticates the received index and token by checking the signature. $RHSP$ verifies their integrity by recomputing the hash value. If either one of the two verification fails, the RHSP drops the corresponding messages. Otherwise, RHSP continues to match users.
- RHSP decrypts the verified index and token. Let $pa_{c_i}[j]$ be the jth ordered pair in the token, i.e.,

$$\begin{aligned} pa_{c_i}[j] &= (pa_{c_i}[j].f, pa_{c_i}[j].s) \\ &= (H_{k_{u+1}}(H_j(c_i)), H_j(c_i)). \end{aligned} \tag{9}$$

- RHSP checks if there exists one $j \in [1, u]$ satisfying

$$\mathcal{B}[H(pa_{c_i}[j].f \oplus r_i)][pa_{c_i}[j].s] = 1. \tag{10}$$

  If the membership checking passes, RHSP choose a random authentication code au_{ij} and computes two matching results:

$$\mathcal{MR}_i = \{Enc(pk_{R_i}, ct_j||au_{ij}), \sigma_{ij}\}, \tag{11}$$

$$\mathcal{MR}_j = \{Enc(pk_{D_j}, ct_i||au_{ij}), \sigma_{ji}\}, \tag{12}$$

  where σ_{ij} and σ_{ji} are two signatures on their preceding ciphertexts.
- RHSP returns $\mathcal{MR}_i$ and $\mathcal{MR}_j$ to R_i and D_j, respectively.

E. Ride Initiation and Completion

After receiving $\mathcal{MR}_j$ from the RHSP, D_j decrypts it and contacts R_i through a new connection channel, where we assume that they can build a secure channel to further exchange the ride information [51], [52]. D_j sends au_{ij} to R_i. If the code is the same as the one received from the RHSP, the ride authentication is complete. D_j takes R_i toward a drop-off location DL_i. Upon arrival, D_i can pay a ride fare to D_j via e-cash, mobile payment, or cash.

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

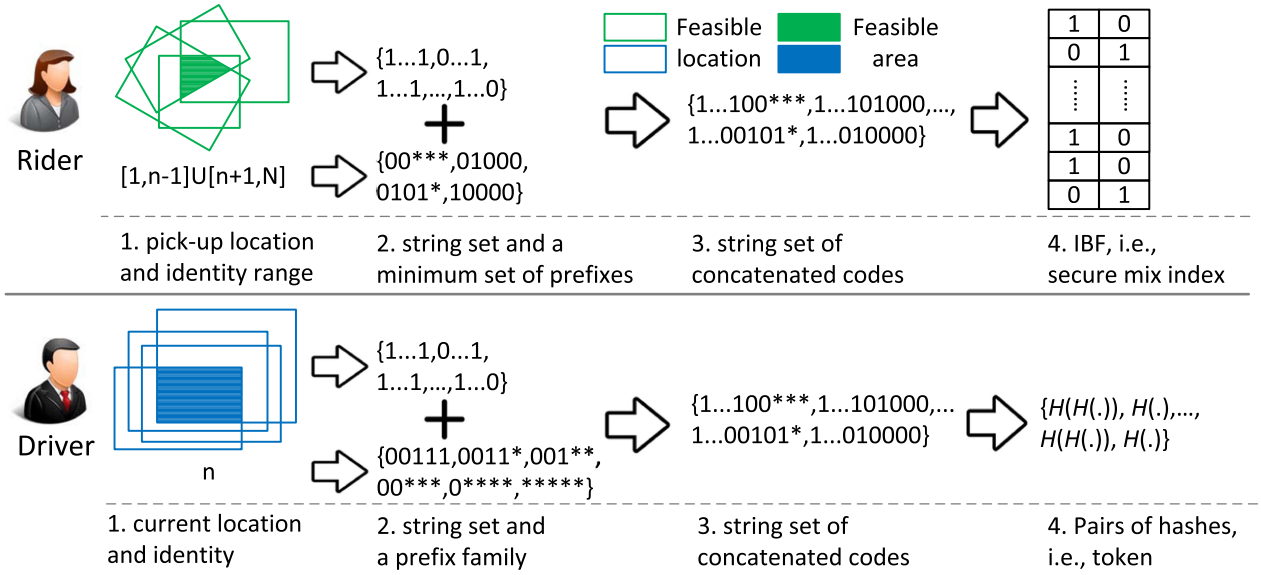


Fig. 3. Transformation of rider's request and driver's response.

VI. Privacy Analysis

In this section, we analyze the privacy properties of Mnemosyne, namely basic location privacy, strong location privacy, anonymity, and unlinkability.

A. Basic Location Privacy

We adopt the commonly acknowledged adaptive indistinguishability under chosen-keyword attack (IND-CKA) secure model [53]. IND-CKA indicates that the index is indistinguishable under chosen keyword attacks. Adaptive means that the adversary adaptively chooses queries based on the previously chosen queries, trapdoors, and query results [21]. In nature, to prove that a scheme is secure in the IND-CKA model against an adaptive adversary, we have to prove the existence of a Probabilistic Polynomial-Time (PPT) simulator that can simulate future unknown queries. An adaptive adversary cannot distinguish between the results of the real secure index and those of the simulator with a non-negligible probability. We next construct such a simulator for Mnemosyne to prove that Mnemosyne is IND-CKA secure against an adaptive adversary. We give two leakage functions as follows:

(1) $L_1(\mathcal{RQ}, \mathcal{LS})$: given a secure mix index $\mathcal{RQ}$ and a location set $\mathcal{LS}$, L_1 returns the size of the IBF and the number of locations.

(2) $L_2(\mathcal{RQ}, \mathcal{LS}, \mathcal{RP}, ts)$: given the secure mix index $\mathcal{RQ}$, the set of locations $\mathcal{LS}$, a token $\mathcal{RP}$, and a timestamp ts, it returns the search pattern and the access pattern. The search pattern is the information about whether $\mathcal{RP}$ was performed before ts. The access pattern is the information about which locations match $\mathcal{RP}$ at ts.

Theorem 1: Mnemosyne achieves basic location privacy, i.e., it is IND-CKA (L_1, L_2)-secure against an adaptive adversary.

Proof: We describe a simulator Sim that can simulate a view $V^* = (\mathcal{RQ}^*, \mathcal{RP}^*)$ by using L_1 and L_2. Next, we show that a PPT adversary $\mathcal{A}$ cannot distinguish between V^* and the real adversary view $V = (\mathcal{RQ}, \mathcal{RP})$. Note that since we do not have encrypted data items, we do not include the them in the view.

1) Simulate $\mathcal{RQ}$. Since Sim can learn the size if IBF from L_1, it builds an IBF $\mathcal{B}$ with the same structure as in the IBF in $\mathcal{RQ}$. In the ith twin of $\mathcal{B}$, Sim puts either 0 in $\mathcal{B}[0][i]$ and 1 in $\mathcal{B}[1][i]$, or 1 in $\mathcal{B}[0][i]$ and 0 in $\mathcal{B}[1][i]$. How to determine cell is decided by tossing a coin randomly. Sim chooses a random number to associate with $\mathcal{B}$. Sim sends $\mathcal{B}$ and r as the simulated secure mix index $\mathcal{RQ}^*$ to $\mathcal{A}$. The simulated $\mathcal{RQ}^*$ has exactly the same structure with the $\mathcal{RQ}$. Therefore, $\mathcal{A}$ cannot distinguish between the simulated secure mix index and the real one, achieving index privacy.
2) Assume that the Sim receives a query. From L_2, Sim knows whether it has been queried before. If so, Sim sends the previous token $\mathcal{RP}$ to $\mathcal{A}$. Otherwise, Sim generates a new token as follows. A token is a set of u-pair of hashes and locations. Since Sim can learn access pattern from L_2, it knows which IBF in the $\mathcal{RQ}$ matches or does not match the token $\mathcal{RP}$. Sim can write the bit output by using H to choose a u-pair and ensure that the chosen u-pair matches or does not match the IBF. By doing so, Sim outputs the generated u-pair as the simulated token $\mathcal{RP}^*$. Since the token is produced by the random hash functions, the simulated token is indistinguishable from the real one, achieving token privacy.

In summary, the simulated view V^* and the real view V are indistinguishable to $\mathcal{A}$. Therefore, our scheme is adaptive IND-CKA (L_1, L_2)-secure in the random oracle model. Therefore, Mnemosyne achieves basic location privacy. □

B. Strong Location Privacy

To achieve strong location privacy, we need to hide the rider's motivation to exclude a specific driver. This is done by integrating the location with identity in ride requesting and

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

ride responding. We prove it in two cases corresponding to the Definition 2 in Section III-C.

Case 1: distinguish between the rider not excluding a previous driver and the rider excluding a driver.

- Case 1.1: locations match. When the rider does not exclude a driver, we assume the rider is assigned the colluding driver. When the driver exclusion is executed, even though their location prefixes can be matched, the range prefix set and the identity prefix family do not have a common element such that after concatenation of the minimum set of prefixes $\mathcal{S}$ and the string set $\mathcal{Q}$, the elements in the rider's $\mathcal{Q}'$ are totally different from the driver's $\mathcal{Q}'$. Therefore, the driver and the RHSP do not know whether it is the location or the identity that lead to the mismatch.
- Case 1.2: locations unmatch. In this case, the rider's $\mathcal{S}$ and the driver's prefix family $\mathcal{F}$ do not match, which result in two different string sets for the rider and the driver, i.e., mismatch. Till here, we have proved that (1) holds, i.e., $|\Pr[\mathcal{A}(\pi(\emptyset;)) = 1] - \Pr[\mathcal{A}(\pi(D))] = 1| \leq \text{negl}$.

Case 2: distinguish between the rider excluding a driver and the rider excluding another (the same) driver.

- Case 2.1: locations match. When the colluding driver is one of the excluded driver, he and the RHSP cannot attribute the mismatch to the identity part because the identity prefixes and location codes are mixed together.
- Case 2.2: locations unmatch. For the same reason in case 1.2, the unmatched locations will result in excluding the two drivers regardless of their identities. Till here, we have proved that (2) holds, i.e., $|\Pr[\mathcal{A}(\pi(D)) = 1] - \Pr[\mathcal{A}(\pi(D'))] = 1| \leq \text{negl}$.

To sum up, if the driver colludes with the RHSP, they cannot tell the difference between the two cases above, thus achieving strong location privacy.

C. Anonymity

We use a set of one-time certificates to stand for the pseudonym of a user. The certificate is randomly generated and only known by the trusted CA in the entity registration phase. Therefore, the one-time certificates look random to the RHSP and even the previously matched drivers, which cannot be link the certificates to the identity of the user, i.e., the users stay anonymous in the ride matching process.

D. Unlinkability

Unlinkability resides in two respects: pseudonym and location. First, we use different certificates for a user to choose from during each ride request or ride response. The RHSP cannot see two ride requests attached with the same certificate. Second, we transform the location of a user to a feasible area that is a cloaking area hiding the real location. The identity range is converted and then integrated with the location string. For different requests, the rider will use different random numbers that lead to different IBF. Similar operations are conducted to the drivers' location and identity. Therefore, two ride requests/responses from a rider/honest driver cannot be linked by the RHSP.

TABLE II
HARDWARE SETTINGS

Server	
CA	Ubuntu Server 20.04 CPU, 2GB RAM, 40GB SSD
RHSP	CentOS 8.2 CPU, 1GB RAM, 25GB SSD
User	
Honor 10	Android 10.0.0, HUAWEI Kirin 970, 6GB RAM
Xiaomi 8	Android 8.1.0, Snapdragon 845 CPU, 6GB RAM
Android VM 1	Android 8.0.0, 2.8GHz*2 CPU, 2GB RAM
Android VM 2	Android 8.0.0, 2.8GHz*2 CPU, 2GB RAM

TABLE III
EXPERIMENTAL PARAMETERS AND VALUES

Notation	Value
H_i, H	100, SHA-256, SHA-384
t	$\{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$
m, u, v	10000, 5, 10
N_R, N_D	$\{20, 40, 60, 80, 100\}$
$\|k\|$, num	1024, $\{1, 2, 3, 4\}$

$t = 8$ by default.

VII. PERFORMANCE EVALUATION

In this section, we build a prototype of Mnemosyne and evaluate its performance regarding computational costs and communication overhead. Specifically, we analyze the time costs under different number of drivers to be excluded. We also compare Mnemosyne with existing work and record the theoretical and practical results.

A. Experimental Settings

We used the JPBC library [46] to implement cryptographic primitives. We chose users' locations from a real-world dataset Gowalla [47] which consists of more than 6 million check-in records of users. The parameters are hash function in IBF H_i and H, number of projection functions t, number of secret keys $u + 1$, number of twins in IBF m, the length of IBF $\mathcal{B}$, number of certificates of a user v, length of secret key $|k|$, driver's identity n, number of riders N_R, number of drivers N_D, and number of drivers to be excluded num.

We also instantiated the Mnemosyne using two servers as CA and RHSP, two android smartphones as rider and driver, and two Android virtual machine as testing devices. Specifically, we create the application based on Flutter [48] on the two Android smartphones and use Gradle to introduce the JPBC library and bcprov-jdk.jar from Bouncy Castle [50]. There are two server-side Spring Boot [49] projects at IDEA using Maven to invoke the downloaded JPBC library. We use websocket and http to communicate among two applications and two server-side projects. We store the data generated during communication in a database using MySQL. We list the hardware setting in Table II and record the main experimental parameters in Table III. We upload the source codes of Mnemosyne to https://github.com/UbiPLab/Mnemosyne.

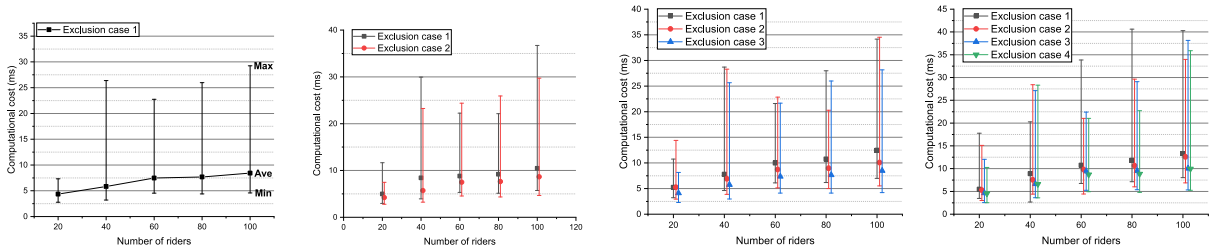
B. Computational Costs

Now we analyze the computational costs of rider, driver, and the RHSP through counting the total number of cryptographic

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
PERFORMANCE ANALYSIS AND COMPARISON WITH EXISTING WORK

Scheme		Computational Costs (Millisecond)			Communication Cost (KBytes)		
		Rider	Driver	RHSP/RSU	Rider	Driver	RHSP/RSU
ORide [1]	Theory	$2T_{Enc} + T_{Dec}$	$2T_{Enc}$	T_{dist}	$O_z + O_{dt} + O_{c_{x_R}} + O_{c_{y_R}} + O_{k_p}$	$O_{c_{x_D}} + O_{c_{y_D}}$	$O_{c_{dist}} + O_{k_p} + O_i$
	Practice	14000	4	100000	372	248	310
FICA [2]	Theory	$2u\|\{p\}\|T_H + 9T_{mul} + T_{Add} + 2T_{Div} + 2T_{bp} + 2T_H$	$u\|\{p\}\|T_H + 9T_{mul} + T_{Add} + 2T_{Div} + 2T_{bp} + 2T_H$	$2(9T_{Mul} + 3T_{Add} + 4T_{Div} + 4T_{bp} + 2T_H) + T_{MA_1}$	$O_{REQ} + O_{Cert} + O_{sig}$	$O_{RES} + O_{Cert} + O_{sig}$	$O_{SYN} + O_{\{res\}}$
	Practice	398	394	143	0.68	1	1
CoRide [3]	Theory	$(4+7u)T_H + T_{E_1} + T_{D_2} + T_{E_2} + 9T_{Mul} + 2T_{Add} + 2T_{Div} + 2T_{bp}$	$(4+5u)T_H + T_{E_1} + T_{D_2} + T_{E_2} + 9T_{Mul} + 2T_{bp}$	$3T_H + 10T_{Mul} + 3T_{bp} + 2T_{Div} + 3T_{Div} + T_{ver} + oT_H + T_{sig}$	O_{Req}	O_{Res}	$O_{mes} + O_{tx}^{Han}$
	Practice	56	55	86	3.63	1.13	0.87
Mnemosyne	Theory	$4(num+1)* uQT_H + T_{sig}$	$3uQT_H + T_{sig}$	$2T_{ver} + T_{MA_2}$	$O_{\mathcal{B}} + O_{ct} + O_{\sigma}$	$O_{\mathcal{RP}} + O_{ct} + O_{\sigma}$	$2(O_{ct} + O_{au})$
	Practice	68	6.3	75	3.82	8.88	0.77



(a) (b) (c) (d)

Fig. 4. Computational Cost of A Rider in Generating an IBF (a) $num = 1$, 1 exclusion case (b) $num = 2$, 2 exclusion cases (c) $num = 3$, 3 exclusion cases (d) $num = 4$, 4 exclusion cases.

operations. T and O denote the computational cost and communication overhead. Mul, Add, Div denote the multiplication, addition, division in the multiplicative cyclic group [2]. bp denote the bilinear paring. H is the hash operation. MA_1 and MA_2 is the match operation on BF and IBF. sig and ver are the signature generation and signature verification. Enc and Dec are the somewhat-homomorphic encryption and decryption [1]. E_1 and D_1 (E_2 and D_2) are public-key (private-key) encryption and decryption [3]. We record the theoretical and practical results in Table IV.

In ride requesting, a rider computes four feasible locations and one feasible area fa, converts the feasible area into a string $\mathcal{Q}$, converts an identity range into a minimum set of prefixes $\mathcal{S}$ (if one driver $n \in (1, N)$ is to be excluded), completes the concatenation, and calculates an IBF. To see how the number of drivers to be excluded num affects the time cost of the rider, we design four sets of experiments where N riders exclude num drivers among N drivers, respectively. Here, $N \in \{20, 40, 60, 80, 100\}$ and $num \in \{1, 2, 3, 4\}$. In the first set where $num = 1$, we ask the 20 riders to randomly choose the identity of the driver to be excluded given 20 drivers. There are several cases to be considered in the last three sets. In the third set where $num = 3$, the identity set of the drivers to be excluded has three possible combination: *all separate*, *one separate*, and *sequential*. For example, the three identity sets can be $\{\underline{1}, \underline{4}, \underline{9}\}$, $\{1, \underline{4}, \underline{5}\}$, $\{\underline{2}, \underline{3}, \underline{4}\}$. In each of the four subfigures in Fig. 4, we mark the maximum, average, and minimum of the time cost at each coordinate point $X = n$ after running n experiments. We can see that the average time cost increases with num because the driver identity range becomes wider (from [1,20] to [1,100]) which leads to more prefixes to be concatenated with the location string set $\mathcal{Q}$ and then more hash operations in the IBF. In the last three subfigures, the average time cost decreases at each X coordinate point from case 1 to case 2, case 3, and case 4. This is because the rider will be faced with more driver identity ranges which mean more prefixes and hash operations.

In ride responding, a driver computes a similar feasible area fa, converts the feasible area into a string set $\mathcal{Q}$, converts the identity into a prefix family $\mathcal{F}$, completes the concatenation, and computes a mix token $\mathcal{RP}$. We conduct three sets of experiments, and we run the ride response phase 60 times in each set. Specifically, the identity of driver is randomly chosen from [2,1023]. The driver only consumes less than 10 ms in generating a ride response. Since the drivers do not have the exclusion step, the time cost of any driver stays stable.

In ride matching, when matching a rider (who excludes one driver) with one driver, the RHSP spends 75 ms. In the four sets of experiments mentioned above, we also record the time cost of the RHSP. From Fig. 5, we can see that the average matching

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

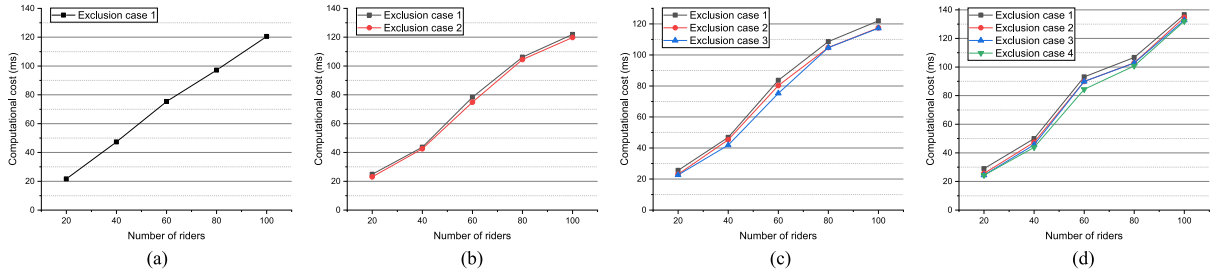


Fig. 5. Computational Cost of The RHSP in Querying the IBFs (a) $num = 1$, 1 exclusion case (b) $num = 2$, 2 exclusion cases (c) $num = 3$, 3 exclusion cases (d) $num = 4$, 4 exclusion cases.

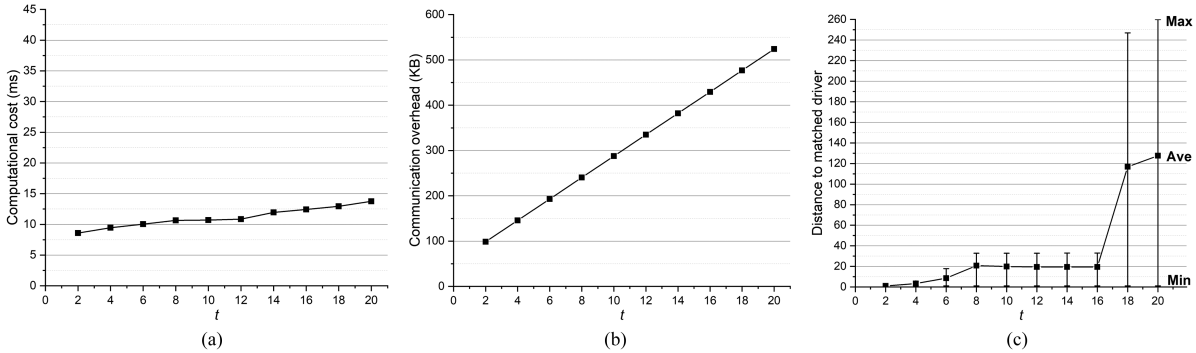


Fig. 6. User's Performance By Varying t (a) Computational Cost (b) Communication Overhead (c) Distance to Matched Driverss.

time of the RHSP increases with n. Corresponding to the second observation in Fig. 4, the RHSP also spends less time with num when n is fixed.

C. Communication Overhead

We now analyze the communication overhead of rider and driver. The the rider only sends an IBF $\mathcal{B}$, a certificate ct, and a signature σ to the RHSP which is $m + |ct| + |\sigma| = 2 * 10000 + 392 * 8 + 1023 * 8$ bits $= 3.82$ KB. A driver sends a mix token $\mathcal{RP}$, a certificate ct, and a signature σ to the RHSP which is $\mathcal{RP} + |ct| + |\sigma| = 256 * u * |\mathcal{Q}| + 392 * 8 + 1023 * 8 = 256 * 5 * 48 + 1544$ bits $= 8.88$ KB. For the rider, the communication overhead does not change with the number of driver to be excluded, because all the codes are inserted into the IBF of which length stays the same. For each pair of rider and driver, the RHSP sends a certificate ct and an authentication code au to the rider/driver. Its length is $2 * (392 * 8 + 10) =$ 0.77 KB.

D. Performance by Varying t

Now we evaluate how t affects the performance of Mnemosyne, namely the computational cost, communication overhead, and distance to the matched driver for rider, and computational cost and communication overhead for driver and the RHSP. Recall that two projection functions determine a feasible location and two feasible locations determine a feasible area. We select t from [2,4,6,8,10,12,14,16,18,20] and each case of t $(t > 2)$ requires an additional OR than the case of $t - 2$. For the rider, with the increase of t, she has to compute more feasible locations fl and feasible areas fa, resulting in a large string set $\mathcal{Q}$ and a larger new string set $\mathcal{Q}'$. Therefore, the rider has to compute more hashings in a larger IBF, i.e., computational cost and communication overhead will increase accordingly. As shown in Fig. 6(a) and (b), the cost echoes this pattern. We also evaluate how t impacts the distance from the rider to the matched driver. In Fig. 6(c), the larger the t is, the longer the maximum distance is. This is because a larger t creates a large feasible area that covers more drivers to be matched. The sudden increment of distance from $t = 16$ to $t = 18$ is due to the big difference between the numbers of matched drivers, where it is 20 in the former case and 37 in the latter. For the driver, with the increase of t, the driver's cost also grows for computing a larger mix token, as shown in Fig. 7(a) and (b). For the RHSP, we test three types of time costs when matching one rider with drivers within her submitted area: (1) min, the minimum time of finding the first matched driver; (2), max: the time of matching with all drivers; and (3) ave, the average time of matching. From Fig. 7(c) and (d), the time cost of the RHSP increases with t while the communication overhead stays the same because the RHSP only returns one driver's information to the rider.

E. Comparison With Existing Work

We compare Mnemosyne with existing work ORide [1], FICA [2], and CoRide [3] in terms of computational costs and communication overhead. The recorded theoretical and practical results in Table IV show that Mnemosyne outperforms other schemes regarding computational efficiency. The driver has to send longer messages to the RHSP due to the adoption of

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

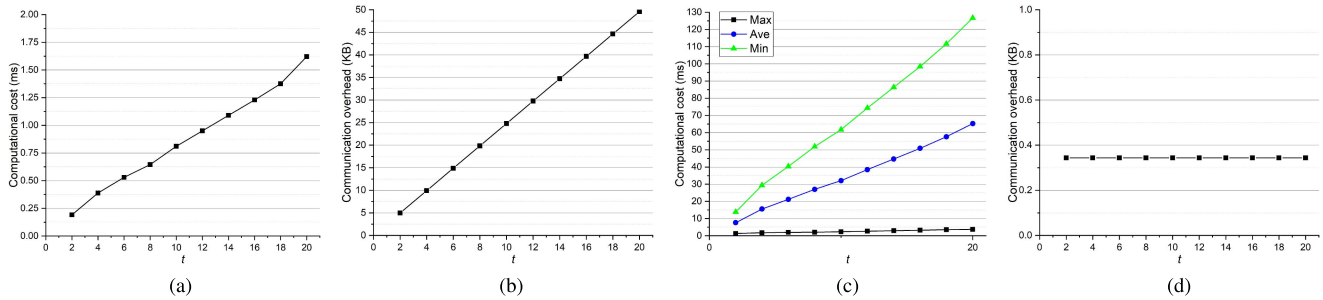


Fig. 7. Driver and RHSP's Performance By Varying t (a) Driver's Comp. Cost (b) Driver's Comm. Overhead (c) RHSP's Comp. Cost (d) RHSP's Comm. Overhead.

hash pairs of $\mathcal{Q}$. We can reduce the communication burden by choosing a small number of secret keys while guaranteeing a satisfying level of security.

VIII. Discussions

A. Assumption of Repetitive Matching

To validate our observation, we initiated a ballot on the Mechanic Turk including two questions: "Do you think it is common for a user to hail a ride frequently from location A to location B?" and "Do you think it is possible for the rider to be matched with the same Uber driver sometimes since the driver may also live near the rider?" The three answers for both questions are "Yes," "No," and "Not sure". We provided $0.02 for answering the two questions as an incentive. From January 25, 2021 to March 17, 2021, we collected 626 answers from 626 different workers. The results show that 432 workers (69%) and 348 workers (67%) chose "Yes" in question 1 and question 2, respectively. The ballot results indicate that most workers agree with our assumption of repetitive matching, which lays a solid foundation for the validity of our assumption. Further, it also calls for location privacy under this scenario.

B. An Extreme Case

We also notice an extreme case where there is only one driver near a rider and the rider requests to exclude this driver. When it happens, the rider will not be matched to this driver and she/he will not enjoy a service immediately. To solve this problem, there are two possible solutions. First, the rider can extend her/his pick-up area to match more drivers a little bit far away. Second, the RHSP can collect feedback from anonymous riders on their platform to know which areas lack drivers and then adjust their driver dispatching strategies.

C. Malicious Rider

In this work, we mainly focus on malicious and colluding RHSP (drivers). As for the rider, it is possible (but not quite likely) that a rider is mischievous and pries into drivers' location privacy under repetitive matching. In this case, we can design a two-way matching method such that both of the riders and driver generate a mix index and a mix token for the matching.

IX. Conclusion

We have identified a new location privacy problem in excluding a previously matched driver in RHS. We consider collusion attack to extend our previous work. We propose Mnemosyne to address the privacy problem by designing a mix index and a mix token. The privacy analysis shows that Mnemosyne achieves strong location privacy. The experimental results demonstrate that Mnemosyne have a low cost under different parameters and outperforms existing RHS schemes regarding computational costs and communication overhead.

References

[1] A. Pham, I. Dacosta, G. Endignoux, J. Troncoso-Pastoriza, K. Huguenin, and J.-P. Hubaux, "ORide: A privacy-preserving yet accountable ride-hailing service," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 1235–1252.
[2] M. Li, L. Zhu, and X. Lin, "Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4573–4584, Jun. 2019, doi: 10.1109/JIOT.2018.2868076.
[3] M. Li, L. Zhu, and X. Lin, "CoRide: A privacy-preserving collaborative-ride hailing service using blockchain assisted vehicular fog computing," in *Proc. 15th EAI Int. Conf. Secur. Privacy Commun. Syst.*, 2019, pp. 408–422.
[4] "DiDi," 2022. [Online]. Available: https://www.didiglobal.com
[5] "Lyft," 2022. [Online]. Available: https://www.lyft.com
[6] "Uber," 2022. [Online]. Available: https://www.uber.com
[7] 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. 17, no. 4, pp. 703–715, Jul.-Aug. 2020, doi: 10.1109/TDSC.2018.2850780.
[8] Y. Luo, X. Jia, S. Fu, and M. Xu, "pRide: Privacy-preserving ride matching over road networks for online ride-hailing service," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 7, pp. 1791–1802, Jul. 2019.
[9] M. Li, L. Zhu, and X. Lin, "Privacy-preserving traffic monitoring with false report filtering via fog-assisted vehicular crowdsensing," *IEEE Trans. Serv. Comput.*, vol. 14, no. 6, pp. 1902–1913, Nov./Dec. 2021, doi: 10.1109/TSC.2019.2903060.
[10] E. Mazareanu, "Monthly number of Uber's active users worldwide from 2017 to 2020, by quarter (in millions)," 2022. [Online]. Available: https://www.statista.com/statistics/833743/us-users-ride-sharing-services
[11] M. Li, D. Hu, C. Lal, M. Conti, and Z. Zhang, "Blockchain-enabled secure energy trading with verifiable fairness in industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6564–6574, Oct. 2020, doi: 10.1109/TII.2020.2974537.
[12] M. Li, L. Zhu, Z. Zhang, C. Lal, M. Conti, and M. Alazab, "User-defined privacy-preserving traffic monitoring against n-by-1 jamming attack," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2060–2073, Oct. 2022, doi: 10.1109/TNET.2022.3157654.
[13] 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. Inform.*, vol. 15, no. 6, pp. 3527–3537, Jun. 2019.
[14] D. Liu, C. Huang, J. Ni, X. Lin, and X. S. Shen, "Blockchain-cloud transparent data marketing: Consortium management and fairness," *IEEE Trans. Comput.*, vol. 71, no. 12, pp. 3322–3335, Dec. 2022.

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

[15] M. Tariq, F. Naeem, M. Ali, and H. V. Poor, "Vulnerability assessment of 6G enabled smart grid cyber–physical systems," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5468–5475, Apr. 2021.

[16] M. Kamal, G. Srivastava, and M. Tariq, "Blockchain-based lightweight and secured V2V communication in internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 3997–4004, Jul. 2021.

[17] F. Naeem, Z. Zhou, S. Siefullahi, and M. Tariq, "A generative adversarial network enabled deep distributional reinforcement learning for transmission scheduling in the internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4550–4559, Jul. 2021.

[18] M. Li et al., "Astraea: Anonymous and secure auditing based on private smart contracts for donation systems," *IEEE Trans. Dependable Secure Comput.*, early access, Sep. 05, 2022, doi: 10.1109/TDSC.2022.3204287.

[19] W. K. Wong, D.W.I. Cheung, B. Kao, and N. Mamoulis, "Secure KNN computation on encrypted databases," in *Proc. 35th ACM SIGMOD Int. Conf. Manage. Data*, 2019, pp. 139–152.

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

[21] R. Li and A. X. Liu, "Adaptively secure conjunctive query processing over encrypted data for cloud computing," in *Proc. IEEE 33rd Int. Conf. Data Eng.*, 2017, pp. 697–708.

[22] X. Lei, A. X. Liu, R. Li, and G. Tu, "SecEQP: A secure and efficient scheme for SkNN query problem over encrypted geodata on cloud," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 662–673.

[23] Y. Chen, M. Li, S. Zheng, D. Hu, C. Lal, and M. Conti, "One-time, oblivious, and unlinkable query processing over encrypted data on cloud," in *Proc. 22nd Int. Conf. Inf. Commun. Secur.*, 2020, pp. 350–365.

[24] A. X. Liu and F. Chen, "Collaborative enforcement of firewall policies in virtual private networks," in *Proc. 27th ACM Symp. Princ. Distrib. Comput.*, 2008, pp. 95–104.

[25] B. H. Bloom, "Space/time tradeoffs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[26] H. Yu, J. Shu, X. Jia, H. Zhang, and X. Yu, "lpRide: Lightweight and privacy-preserving ride matching over road networks in online ride hailing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 10418–10428, Nov. 2019.

[27] H. Xie, Y. Guo, and X. Jia, "A privacy-preserving online ride-hailing system without involving a third trusted server," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 3068–3081, 2021.

[28] M. Li, Y. Chen, C. Lal, M. Conti, F. Martinelli, and M. Alazab, "Nereus: Anonymous and secure ride-hailing service based on private smart contracts," *IEEE Trans. Dependable Secure Comput.*, early access, Jul. 19, 2022, doi: 10.1109/TDSC.2022.3192367.

[29] Y. Cao, Y. Xiao, L. Xiong, L. Bai, and M. Yoshikawa, "Protecting spatiotemporal event privacy in continuous location-based services," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 8, pp. 3141–3154, Aug. 2021.

[30] G. Sun, Y. Li, H. Yu, A. Vasilakos, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," *Future Gener. Comput. Syst.*, vol. 1, no. 91, pp. 347–360, 2019.

[31] X. Du and F. Lin, "Improving routing in sensor networks with heterogeneous sensor nodes," in *Proc. IEEE 61st Veh. Technol. Conf.*, 2005, pp. 2528–2532.

[32] Z. Guan, X. Lu, N. Wang, J. Wu, X. Du, and M. Guizani, "Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks," *Future Gener. Comput. Syst.*, vol. 1, no. 110, pp. 686–695, 2020.

[33] Y. Xiao, H. Chen, X. Du, and M. Guizani, "Stream-based cipher feedback mode in wireless error channel," *IEEE Trans. Wireless Commun.*, vol. 8, no. 2, pp. 622–626, Feb. 2009.

[34] M. Li, Y. Chen, S. Zheng, D. Hu, C. Lal, and M. Conti, "Privacy-preserving navigation supporting similar queries in vehicular networks," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1133–1148, Mar./Apr. 2022, doi: 10.1109/TDSC.2020.3017534.

[35] M. Li, Y. Chen, C. Lal, M. Conti, M. Alazab, and D. Hu, "Eunomia: Anonymous and secure vehicular digital forensics based on blockchain," *IEEE Trans. Dependable Secure Comput.*, early access, Nov. 25, 2021, doi: 10.1109/TDSC.2021.3130583.

[36] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4585–4600, Jun. 2019.

[37] J. Kang, Z. Xiong, D. Niyato, P. Wang, D. Ye, and D. I. Kim, "Incentivizing consensus propagation in proof-of-stake based consortium blockchain networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 1, pp. 157–160, Feb. 2019.

[38] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[39] Z. Xiong, Y. Zhang, N. C. Luong, D. Niyato, P. Wang, and N. Guizani, "The best of both worlds: A general architecture for data management in blockchain-enabled Internet-of-Things," *IEEE Netw.*, vol. 34, no. 1, pp. 166–173, Jan./Feb. 2020.

[40] C. Zhang, L. Zhu, C. Xu, J. Ni, C. Huang, and X. Shen, "Location privacy-preserving task recommendation with geometric range query in mobile crowdsensing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4410–4425, Dec. 2022.

[41] M. Li, L. Zhu, Z. Zhang, C. Lal, M. Conti, and M. Alazab, "Anonymous and verifiable reputation system for E-commerce platforms based on blockchain," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4434–4449, Dec. 2021, doi: 10.1109/TNSM.2021.3098439.

[42] M. Li, Y. Chen, N. Kumar, C. Lal, M. Conti, and M. Alazab, "Quantifying location privacy for services in sustainable vehicular networks," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1267–1275, Sep. 2022, doi: 10.1109/TGCN.2022.3144641.

[43] 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.*, vol. 18, no. 3, pp. 1291–1306, May/Jun. 2021.

[44] H. Yu, J. Shu, X. Jia, H. Zhang, and X. Yu, "lpRide: Lightweight and privacy-preserving ride matching over road networks in online ride hailing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 10418–10428, Nov. 2019.

[45] P. Zhao et al., "Synthesizing privacy preserving traces: Enhancing plausibility with social networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2391–2404, Dec. 2019.

[46] A. De Caro and V. Iovino, "The java pairing based cryptography library (JPBC)," in *Proc. IEEE 16th Symp. Comput. Commun.*, 2011, pp. 850–855. [Online]. Available: http://gas.dia.unisa.it/projects/jpbc

[47] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2011, pp. 1082–1090.

[48] "Flutter-build apps for any screen," 2022. [Online]. Available: https://flutter.dev

[49] "Spring Boot," 2022. [Online]. Available: https://spring.io/projects/spring-boot

[50] "The legion of the bouncy castle," 2022. [Online]. Available: https://www.bouncycastle.org/java.html

[51] H. Yu, X. Jia, H. Zhang, X. Yu, and J. Shu, "PSRide: Privacy-preserving shared ride matching for online ride hailing systems," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1425–1440, May/Jun. 2021.

[52] J. Huang, Y. Luo, S. Fu, M. Xu, and B. Hu, "pRide: Privacy-preserving online ride hailing matching system with prediction," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7413–7425, Aug. 2021.

[53] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. 13th ACM SIGSAC Conf. Comput. Commun. Secur.*, 2006, pp. 79–88.

Meng Li (Senior Member, IEEE) received the Ph.D. degree in computer science and technology from the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China, in 2019. He is currently an Associate Researcher and Dean Assistant with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China. He is also a Postdoctoral Fellow with the Department of Mathematics and HIT Center, University of Padua, Padua, Italy, where he is with the Security and Privacy Through Zeal (SPRITZ) Research Group led by Prof. Mauro Conti. He was sponsored by ERCIM 'Alain Bensoussan' Fellowship Programme (2020.10.1-2021.3.31) to conduct post-doctoral research supervised by Prof. Fabio Martinelli with CNR, Italy. He was sponsored by China Scholarship Council (2017.9.1-2018.8.31) for joint Ph.D. study supervised by Prof. Xiaodong Lin with the Broadband Communications Research (BBCR) Lab, University of Waterloo, Waterloo, ON, Canada, and Wilfrid Laurier University, Waterloo. He has authored or coauthored more than 50 papers in international peer-reviewed venues, including TDSC, ToN, TVT, TSC, TNSM, TII, TNSE, TGCN, *IEEE Communications Magazine*, IEEE WIRELESS COMMUNICATIONS, MobiCom, ICICS, SecureComm, and TrustCom. His research interests include security, privacy, fairness, vehicular networks, applied cryptography, privacy computing, cloud computing, edge computing, and blockchain.

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.

Jianbo Gao (Student Member, IEEE) received the B.E. degree from Anhui Medical University, Hefei, China, in 2020. He is currently working toward the M.S. degree with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China. His research interests include security and privacy, applied cryptography, and vehicular networks.

Zijian Zhang (Member, IEEE) received the Ph.D. degree from the School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China. He is currently a Research Fellow with the School of Computer Science, University of Auckland, Auckland, New Zealand. In 2015, he was a Visiting Scholar with the Computer Science and Engineering Department, State University of New York at Buffalo, Buffalo, NY, USA. His research interests include design of authentication and key agreement protocol and analysis of entity behavior and preference.

Liehuang Zhu (Senior Member, IEEE) is currently a Full Professor with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China. He is selected into the Program for New Century Excellent Talents in University from Ministry of Education, China. He has authored or coauthored more than 100 SCI-indexed research papers in his research field, which include Internet of Things, cloud computing security, internet and mobile security, and a book published by Springer. He is on the Editorial Boards of three international journals, including IEEE INTERNET-OF-THINGS JOURNAL, IEEE NETWORK, and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. He was the recipient of the Best Paper Award at IEEE/ACM IWQoS 2017 and IEEE TrustCom 2018.

Chhagan Lal received the Ph.D. degree in computer science and engineering from the Malaviya National Institute of Technology, Jaipur, India, in 2014. He is currently a Postdoctoral Research Fellow with the Delft University of Technology, Delft, The Netherland. He was a Postdoctoral Fellow with the Department of Mathematics, University of Padua, Padua, Italy, where he was part of the SPRITZ Research Group. He was a Postdoctoral Research Fellow with Simula Research Laboratory, Norway. His research interests include applications of blockchain technologies, security in software-defined networking, and Internet of Things networks. During his Ph.D., he was awarded with the Canadian Commonwealth Scholarship under the Canadian Commonwealth Scholarship Program to work with the University of Saskatchewan, Saskatoon, SK, Canada.

Mauro Conti (Fellow, IEEE) received the Ph.D. degree from the Sapienza University of Rome, Rome, Italy, in 2009. He is currently a Full Professor with the University of Padua, Padua, Italy. He is also affiliated with the Delft University of Technology (TU Delft), Delft, The Netherland, and University of Washington, Seattle, WA, USA. After his Ph.D., he was a Postdoc Researcher with Vrije Universiteit Amsterdam, Amsterdam, The Netherlands. In 2011, he joined as an Assistant Professor with the University of Padua, where he became an Associate Professor in 2015, and a Full Professor in 2018. He has been Visiting Researcher with George Mason University (GMU): Home, University of California, Los Angeles (UCLA), University of California, Irvine (UCI), Technical University of Darmstadt (TU Darmstadt), University of Florida (UF), and Florida International University (FIU). He was awarded with a Marie Curie Fellowship in 2012 by the European Commission, and with a Fellowship by the German DAAD in 2013. His research is also funded by companies, including Cisco, Intel, and Huawei. He has authored or coauthored more than 400 papers in topmost international peer-reviewed journals and conferences, in his research field, which include the area of security and privacy. He is the Editor-in-Chief of IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, Area Editor-in-Chief of IEEE COMMUNICATIONS SURVEYS & TUTORIALS, and has been an Associate Editor for several journals, including IEEE COMMUNICATIONS SURVEYS & TUTORIALS, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, and IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. He was the Program Chair for TRUST 2015, ICISS 2016, WiSec 2017, ACNS 2020, CANS 2021, and General Chair for SecureComm 2012, SACMAT 2013, NSS 2021, and ACNS 2022. He is a Senior Member of the ACM, and Fellow of the Young Academy of Europe.

Mamoun Alazab (Senior Member, IEEE) received the Ph.D. degree in computer science from the School of Science, Information Technology and Engineering, Federation University of Australia, Ballarat, VIC, Australia. He is currenlty an Associate Professor with the College of Engineering, IT and Environment, Charles Darwin University, Darwin, Nt, Australia. He is currently a Cybersecurity Researcher and Practitioner with industry and academic experience. His research focuses on multidisciplinary that focuses on cybersecurity and digital forensics of computer systems with a focus on cybercrime detection and prevention including cyber terrorism and cyber warfare. He has more than 150 research papers. He delivered many invited and keynote speeches, 24 events in 2019 alone. He convened and chaired more than 50 conferences and workshops. He works closely with government and industry on many projects, including the Northern Territory Department of Information and Corporate Services, IBM, Trend Micro, Australian Federal Police, Australian Communications and Media Authority, Westpac, United Nations Office on Drugs and Crime, and the Attorney Generals Department.He is the Founding Chair of the IEEE Northern Territory Subsection.

Authorized licensed use limited to: TU Delft Library. Downloaded on May 17,2023 at 06:39:29 UTC from IEEE Xplore. Restrictions apply.