Lightweight Privacy-Preserving Ride-Sharing Protocols for Autonomous Cars

Sara Ramezanian
sara.ramezanian@helsinki.fi
University of Helsinki
Helsinki Institute for Information Technology (HIIT)
Helsinki, Finland

Gizem Akman
gizem.akman@helsinki.fi
University of Helsinki
Helsinki Institute for Information Technology (HIIT)
Helsinki, Finland

Mohamed Taoufiq Damir
mohamed.damir@helsinki.fi
University of Helsinki
Helsinki Institute for Information Technology (HIIT)
Helsinki, Finland

Valtteri Niemi
valtteri.niemi@helsinki.fi
University of Helsinki
Helsinki Institute for Information Technology (HIIT)
Helsinki, Finland

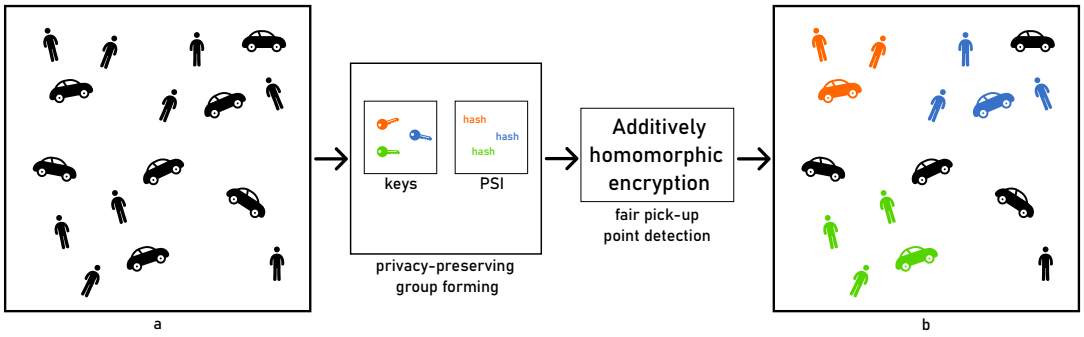

Figure 1: The figure illustrates a summary of our privacy-preserving ride-sharing protocol. Initially, the cars and passengers are not grouped (Fig. 1.a). After applying our protocol, the cars are assigned to groups of passengers with similarity in their journeys (Fig. 1.b).

ABSTRACT

Ride-sharing is a popular way of transportation that reduces traffic and the costs of the trip. Emerge of autonomous vehicles makes ride-sharing more popular because these vehicles do not require a driver's effort. Therefore, in order to find a suitable ride-share, the service provider is not restricted to the driver's trip. Thus, the autonomous cars are more flexible with matching the passengers. Passengers who want to participate in car-sharing send their trip data to a ride-sharing service provider. However, the passenger's trip data contains sensitive information about the passenger's locations. Multiple studies show that a person's location data can reveal personal information about them, e.g., their health condition, home, work, hobbies, and financial situation. In this paper, we propose a lightweight privacy-preserving ride-sharing protocol for autonomous cars. Contrary to previous works on this topic, our protocol does not rely on any extra party to guarantee privacy and security. Our protocol consists of two main phases: i) privacy-preserving group forming, and ii) privacy-preserving fair pick-up point selection. In addition to ride-sharing, the two phases of our protocol can also be applied to other use cases. We have implemented our protocol for a realistic ride-sharing scenario, where 1000 passengers simultaneously request a ride-share. Our evaluation results show that the time and communication costs of our protocol are such that it is feasible for real-world applications.

CSCS '22, December 8, 2022, Ingolstadt, Germany

ACM ISBN 978-1-4503-9786-5/22/12.
https://doi.org/10.1145/3568160.3570234

CCS CONCEPTS

• **Security and privacy → Privacy-preserving protocols**.

KEYWORDS

Autonomous Cars, Ride-Sharing, Privacy-Enhancing Technologies, Location Privacy, Private Set Intersection, Lightweight Cryptography.

ACM Reference Format:
Sara Ramezanian, Gizem Akman, Mohamed Taoufiq Damir, and Valtteri Niemi. 2022. Lightweight Privacy-Preserving Ride-Sharing Protocols for Autonomous Cars. In *Computer Science in Cars Symposium (CSCS '22), December 8, 2022, Ingolstadt, Germany*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3568160.3570234

1 INTRODUCTION

Ride-Sharing Services (RSSs) have gradually attracted more users in the past years. RRSs offer various advantages compared to classical means of transportation, e.g., reduced traffic congestion, alleviated parking shortages, and minimized transportation cost. Face to the tremendous demand for such an alternative, ride-sharing companies have been steadily growing across the globe, e.g., Bolt, Blablacar, UberPool, Flinc, and LyftLineT. The global ride-sharing market size was estimated at 84.30 billion USD in 2020, and it is expected to reach 242.73 billion USD in 2028 [19]. In this work, we present a ride-sharing scheme via autonomous cars that preserves the user's privacy. In this section, we first explain why preserving the user's privacy in the context of ride-sharing is important. Then, we explain how autonomous vehicles change the existing ride-sharing schemes. We then explain a broader related topic that is called *location privacy*. At the end of the section, we give the list of our contributions.

1.1 Privacy-Preserving Ride-Sharing

Usually, a RSS operates as a middleman who connects drivers to passengers via a mobile app. The service often requests the passenger/driver to provide their trip data, i.e., the ride time, the pick-up and drop-off locations. Accessing such information by a malicious entity is a serious threat to the user's privacy. Moreover, storing such (valuable) data by RSSs in their centralized servers made RSSs an attractive target for attackers. For example, in 2016, an attack on Uber servers resulted in stealing the records of 57 million users and drivers. More recently [37], the same company was a target of a cyber-attack. The attack's impact is still under investigation at the time of writing. It is worth mentioning that such a type of attack does not only affect the user's privacy but might also cause financial consequences for the company. For example, after the 2016 breach, Uber admitted to hiding the attack and paid 148 million USD in settlement with multiple U.S states [7]. Therefore, organizing a privacy-preserving ride-sharing (PPRS) is beneficial for both the service provider and its users.

1.2 Autonomous Cars and Ride-Sharing

Autonomous vehicles can positively impact our everyday lives in many ways, such as less car accidents, safe mobility option for non-ambulatory and elderlies, and helps fighting global warming [2, 23]. An autonomous vehicle does not require a driver; therefore, organizing a ride-share can be made regardless of the drop-off point. Please note that with a regular car, the passengers should leave the car before/at the same time as the driver. Therefore, compared with the "classical" passenger/driver RSSs, organizing a ride-share with an autonomous car is more flexible. Thus, we can expect even bigger requests for ride-sharing via autonomous cars [11]. However, designing a PPRS protocol for an autonomous car is more difficult than the classical case because of the elimination of the driver (who can play the role of a trusted third party in a PPRS protocol) and the ambiguity of the drop-off points (and consequently the whole trip route). In the present work, we focus on the users' privacy and propose a novel ride-sharing scheme for autonomous cars.

1.3 Location Privacy

The concept of privacy-preserving ride-sharing is a special case of a wider research area that is called *location privacy*. A user's location data can potentially reveal sensitive information about the user, such as their lifestyle, political and religious views, their (and their close ones') home/work locations, and their health and financial situation [17, 28, 38]. The general approaches to preserve location privacy are i) location obfuscation (hiding the exact location by adding noise), ii) anonymity, and iii) utilizing cryptographical techniques to encrypt the location [47]. To design a privacy-preserving ride-sharing, one or several of the above general location privacy approaches are used. In our protocol, we choose both anonymity and cryptography to ensure the user's privacy. Moreover, there has been a trend to utilize a third party in privacy-preserving location-related protocols [12]. However, utilizing a third party (such as a trusted third party or a cloud) can cause additional security and privacy challenges. For instance, the presence of a Trusted Third Party (TTP) in a scheme can introduce more costs [26] and make the scheme more prone to cyber-attacks [21, 26]. Although it is popular, cloud computing can cause a series of privacy and security challenges [34], such as insider attacks, denial of service attacks, and permanent data loss. Therefore, we avoid utilizing any third parties in our PPRS scheme.

1.4 Our Contributions

The list of our contributions is as follows:

- We modify the tree Diffie-Hellman (DH) group key exchange protocol [41] such that the parties can generate the secret key without the help of the server.
- We design a novel privacy-preserving group forming scheme. In our scheme, we use our modified tree DH group key exchange protocol, keyed hash function, and a private set intersection protocol. This scheme can be applied to other use cases than PPRS, e.g., potential partner suggestions on dating apps.
- We propose a privacy-preserving fair pick-up point selection protocol that utilizes additively homomorphic encryption. This protocol also can be applied to other use cases, e.g., organizing a social gathering in a location that is fair for all the participants.
- We propose a novel lightweight privacy-preserving ride-sharing protocol for autonomous cars. For security and privacy reasons, we do not use any additional party in our protocol. To the best of our knowledge, our protocol is the first ride-sharing scheme to achieve privacy without relying on any third party.

The rest of the paper is organized as follows: Section 2 gives the required preliminaries. In Section 3, we provide a closer look into our research problem, and in Section 4, we present the prior related work on the topic. Our novel protocol is presented in Section 5, and our evaluation results on the protocol are given in Section 6. We analyze the security and privacy of our protocol in Section 7. Finally, the future work and conclusion are given in Section 8.

2 PRELIMINARIES

This section provides the necessary preliminaries that are required for the rest of the paper.

2.1 Keyed Hash Function

A *keyed hash* function is an algorithm that uses a secret key and a one-way and collision-resistant hash function to create an output. The secret key is shared between all the parties that are computing the output [39].

2.2 Private Set Intersection (PSI)

Private Set Intersection (PSI) allows two parties (or more) to compute the intersection of their sets without revealing any information about the items that are not in the intersection. The efficiency of a given PSI protocol depends on its respective scenario, see for instance [30] and the references therein.

2.3 Dictionary Attack

A *dictionary attack* is a systematic brute-force approach of guessing a secret, e.g., decrypting ciphertext, by trying a restricted subset of possible solutions, e.g., keys.

2.4 Denial of Service (DoS)

A *Denial of Service* (Dos) attack is an attack that prevents users from accessing a service by overwhelming the service's physical resources or network connections. In general, this is done by flooding the service with user's traffic requests. To prevent DoS attacks, the service should be equipped with a mechanism that distinguishes between legit and malicious requests. One way to do so is to use *anonymous tokens*.

2.5 Anonymous Tokens

Anonymous tokens are lightweight, single-use anonymous credentials. Typically, a user receives a limited amount (less than the amount required to perform a DoS attack) of anonymous tokens when they prove to the service that they are legitimate user. Therefore, the user can access the service anonymously during their subsequent connections using these credentials [22].

2.6 Adversarial Model

A *semi-honest adversary*, a.k.a honest-but-curious, is a party that follows the protocol specification exactly. However, they may try to learn more information than allowed by investigating into the messages they received. A stronger adversary model is *malicious adversary*. A malicious party may deviate from the protocol specification and uses more efficient attack strategies, e.g., manipulating the protocol's input/output. The above two adversarial models are broadly accepted as the main adversary models in multi-party protocols [15].

2.7 Paillier Cryptosystem

Paillier Cryptosystem [29] is a widely adopted additively homomorphic probabilistic public-key encryption scheme. A probabilistic public-key encryption scheme (Gen, E, D), utilizes a key generation algorithm Gen to create a pair of public and secret keys (pk, sk). The probabilistic encryption function E takes three inputs: the public key pk, a random string r, and the plaintext m, and outputs a ciphertext $c = E_{pk}(m, r)$. A crucial property of E is the fact that encrypting the same message twice results in two different ciphertexts. The decryption function is denoted by $D_{sk}(c)$. For any plaintexts m_1, m_2, Paillier cryptosystem has the following homomorphic property[1]: $D_{sk}(E_{pk}(m_1, r_1) \times E_{pk}(m_2, r_2)) = m_1 + m_2$.

2.8 AES

The Advanced Encryption Standard (AES) [9] is a widely used symmetric key block cipher. AES encrypts a plaintext block of 128 bits using either a 128, 192, or 256 bits key.

2.9 Diffie–Hellman Key Exchange Protocol

The Diffie–Hellman exchange protocol (DH) [10] goes as follows: Let G be a finite cyclic group with order $|G|$ generated by g. In order to generate a shared secret key between two parties, A and B, the parties secretly choose integers s_A and s_B, respectively, at random from the interval $[0; |G| - 1]$. Then, they compute g^{s_A} and g^{s_B}, respectively, and exchange these group elements over the public channel. Finally, A and B compute the shared secret $K_A = (g^{s_B})^{s_A}$ and $K_B = (g^{s_A})^{s_B}$, respectively. Note that $K_A = K_B$.

2.10 Key Derivation Function

A Key Derivation Function (KDF) is an algorithm that takes a given secret as input, e.g., a secret password, and outputs a bitstring that is suitable for some cryptographic operation. For example, a KDF can be used to convert a Diffie-Hellman shared key to an AES key.

2.11 Tree Diffie-Hellman Group Key Exchange Protocol

A Tree Diffie-Hellman Group Key Exchange protocol is a generalization of the DH protocol to n parties. To establish a shared secret key between n parties $P_1, \ldots, P_n$, a binary tree is recursively constructed via the DH protocol on the multiplicative group modulo a prime p. One of the early examples of a tree-based group key exchanges is [33] and a more in-depth analysis figures in [41]. The protocol of [41] was designed in such a way that every user requires to establish a 2-party DH key with a server. Several prior works on the group key exchange protocol reduce/omit the role of the server, such as [14] and [13]. These protocols could be used in our protocol instead of our variant that is presented in Section 5.1. Various works [8, 43] later extended the earlier methods to cover further security and design properties, such as adding/removing parties, merging, managing, etc. The main idea behind these works is to construct a shared secret by a recursive use of the two parties Diffie-Hellman.

3 PROBLEM STATEMENT

In this section, we first discuss the privacy issues in ride-sharing via autonomous cars. Then, we formalize the research problem.

As we mentioned in Section 1, at the time of writing, the companies that offer ride-sharing services organize the trips without considering the privacy of their users. More precisely, a user reveals their identity, pick-up and drop-off locations, and time of travel

[1]The equalities are taken modulo a given integer, see [29] for more detail.

to the system. This information can reveal privacy-sensitive data about the user, such as their home, workplace, hobbies, their children's schools and hobbies, and the exact time of the day when they are not at home [24]. Revealing this private information in the online world can cause unwanted tracking [3], stalking [4], and the breach of the users' health data [17]. Therefore, we aim to design our privacy-preserving ride-sharing protocol in such a way that after executing the protocol, the user's identity, location, and time of the trip remain private.

Unlike regular cars, autonomous vehicles do not require a driver. Therefore, organizing ride-sharing requires different settings. With a regular car, the vehicle is rented or owned by a driver. If the driver wants to organize a ride-sharing, they should find other passengers that have similarities in their journey to the driver. Usually, the driver does not want to deviate from their route and prefers to pick up passengers that are traveling on the same path as the driver. Also, the drop-off points of the passengers should be similar or before the driver's destination. However, with an autonomous car, the passengers can agree on one or more pick-up points, and the car can go and pick them up. Moreover, the drop-off points of the passengers can be different. The only requirement for the drop-off points is that they should be in the same direction. Therefore, we aim to design privacy-preserving techniques that enable the server to group the passengers with similarity in their trips (initial location, direction of the trip, date, and time) together. We also want to enable the passengers to agree on a "fair" pick-up point in a private way.

Before going any further, let us explain what we mean by a "fair" pick-up point. A *fair* pick-up point is a location where its distance from all the parties is fair. Moreover, the pick-up point itself should be convenient, i.e., the point should be reachable easily by an autonomous car.

We formalize the problem of privacy-preserving ride-sharing arrangements for autonomous cars as follows: n users want to arrange ride-sharing at different times, locations, and for different routes. For security and privacy reasons, the users do not want to reveal any information about their locations and identities to each other or anybody else. A server of a company with autonomous cars wants to provide RSSs in a privacy-preserving way. The server wants to group the users who have similarities in their trips (and therefore can share a ride) together without learning any information about their trips. After the server performs the privacy-preserving group forming, the users of each group want to privately determine a fair pick-up point to start their trip together. The pick-up point selection should be made such that the users do not learn each other's locations. The server also remains oblivious to the users' locations. Moreover, for privacy and security reasons that are mentioned in Section 1, we want to avoid utilizing any other party (e.g., cloud, proxy, and TTP) than the server and the passengers. Lastly, we want to design our protocol in such a way that it can run in real-time, i.e., with negligible time and bandwidth usage.

4 RELATED WORK

The problem of designing a privacy-preserving ride-sharing protocol for both autonomous and regular cars has been studied previously. To organize a (privacy-preserving) ride-share, we require a (privacy-preserving) technique to match the passengers who have similar trip data, and a (privacy-preserving) scheme to determine the pick-up points. Therefore, in this section, we review some of the noteworthy articles on private group-forming techniques, privacy-preserving pick-up points selection schemes, and privacy-preserving ride-sharing protocols.

4.1 Privacy-Preserving Group Forming

The problem of finding nearby parties in a private manner is not restricted to ride-sharing. This research problem is studied so that the solutions can cover any use cases that require similarity tests with respect to location. Some use cases are ride-sharing [16], locating nearby friends on social media [44], and potential partner suggestions on dating apps [25]. One solution to the problem of finding nearby parties is to perform a privacy-preserving proximity test [20, 46]. However, proximity tests are usually performed between two parties; therefore extending the test to cover more than two parties requires many pre-computations and several rounds of communication. Moreover, the proximity tests may result in false positives/negatives. Thus, these tests are not suitable for matching passengers in a ride-sharing protocol.

He et al. proposed a three steps partner selection protocol for the purpose of privacy-preserving ride-sharing for a regular car [16]. Although their protocol does not introduce any false positive/negative and therefore is more practical than its predecessors, they require that the driver and the riders reveal some information about their pick-up and drop-off locations. The riders' pick-up point is determined by the driver, and therefore, the riders cannot pick a fair pick-up location. Moreover, the protocol requires multiple rounds of computation and communication between the server, the driver, and the riders, which makes the protocol costly.

4.2 Privacy-Preserving Fair Pick-up Point Selection

In most of the ride-sharing systems, the passengers' pick-up points are selected by the system or the driver, and the passengers cannot select a "fair" pick-up point for themselves [1]. The concept of fair pick-up selection is not limited to the ride-sharing organizations. A fair meeting point selection is relevant to any scenario where a group of entities (more than one) is required to be at a certain place at a given time. Therefore, privacy-preserving fair meeting point selection is studied independently from a specific use case (e.g., ride-sharing). However, in most of the previous works, the protocol relies on an additional party (than the server and the group of people), such as a cloud or a TTP [31, 35, 42], and the protocols' time complexities are not suitable for real-time location selection [5, 31, 42].

In Section 5, we propose a lightweight fair meeting point selection protocol between a group of passengers and a server that does not rely on any additional party and can perform in real time.

4.3 Privacy-Preserving Ride-Sharing

The following two approaches are common among prior works:

(1) Protocols involve additional party/parties than the passenger and the server. In the context of privacy-preserving ride-sharing, the encrypted trip data of several passengers should

be checked against each other to find a possible match. One way to preserve the privacy of the passengers against the server is to utilize another entity to handle the secret key that is used to encrypt the trip data, e.g., a third-party server [18, 27, 35, 45], a primary passenger (driver) [32], and a cloud [36].

(2) It is assumed that the server and all the passengers are semi-honest [1, 18, 32, 45]. In the context of privacy-preserving ride-sharing, it is realistic to assume that the server is semi-honest because the server does not gain anything by disrupting its reputation. However, we cannot safely assume that the passengers will not act maliciously.

In our protocol, we drop the assumption that the passengers are always semi-honest. Also, we do not add any entities other than the parties who are naturally part of the protocol, i.e., the server and the passenger. Because, utilizing an extra party (such as a TTP or a cloud) may introduce additional security and privacy challenges to the system. For example, Utilizing a TTP/cloud in a scheme can introduce more cyber-attacks [21, 26, 34]. Now, we take a closer look at a previous work on privacy-friendly RSSs for an autonomous car.

Sherif et al. proposed a privacy-preserving protocol to organize a ride-sharing for a primary passenger via autonomous vehicles [32]. Their protocol is between three different parties; a primary passenger, a set of secondary passengers, and a server. Sherif et al. assumed that the primary passenger rents/owns an autonomous car and tries to find other passengers to share the ride with. Their protocol is as follows: The primary passenger generates a secret key and a group signature credentials and distributes the secret key and the signature between all the other passengers that requested a ride-share (secondary passengers). Then, all the passengers (primary and secondary) encrypt their trip data with the primary user's secret key, send the data to the server, and the server performs a similarity test over the encrypted data. The server does not have the secret key, so it cannot learn any information about the trip from the encrypted data. The server sends the result of the similarity test to the primary passenger. Then, the primary passenger contacts the secondary passengers, who can share a ride. Sherif et al. adversarial model is semi-honest. However, this assumption is not always realistic for the passengers, especially since the primary passenger broadcasts the secret key and the group signature credentials to the secondary passengers (up to 500 passengers). Moreover, it is not clear how the primary passenger can securely deliver the key and signature to the secondary passengers.

5 THE PROTOCOL

In this section, we present our privacy-preserving ride-sharing protocol for autonomous cars. The protocol runs between the server of a company and the customers of that company. The company owns several autonomous cars. Among other services, the company provides ride-sharing services in several cities. Let us assume that the names of these cities are stored in a set C. The company's server divides the map of each city into subregions, e.g., squares with an area of 1 km^2. Each subregion has a number, and the map with its numbered subregions is available to all the users of the service.

The company has several parking spots in different locations that are distributed throughout the cities. When not in use, the autonomous cars will drive themselves to the nearest free parking spot, and if required, they recharge their batteries there as well. The server makes the list of potential parking locations visible to the users. Hereafter, if a customer of the company wants to utilize the ride-sharing service, we call that user a *passenger*.

In this section, we first give a brief overview of our protocol to provide an overall image of the different phases of the protocol. Then, we explain the steps of the main phases of our protocol in detail.

The protocol consists of a setup phase and two main phases. In the setup phase, the passengers register to the ride-sharing system by paying a fee. Then, for every ride, the passengers adopt a pseudo-random identity so that they can utilize the system anonymously. The passengers also obtain random-looking access tokens, which are used together with pseudo-random identities to prove that there is a valid customer behind the identity. An oblivious transfer is a well-established cryptographic protocol that guarantees fetching valid tokens without revealing any information to the server about who gets which tokens. Each token can only be used once, and its purpose is to reduce the potential for denial of service.

Still, in the setup phase, the server picks a non-deterministic and additively homomorphic asymmetric cryptosystem, e.g., the Paillier cryptosystem [29]. We assume that the public key for this cryptosystem is p_k, and s_k is the secret key. The encryption and decryption functions are E_{p_k} and D_{s_k}, respectively. The server distributes the public key p_k to all passengers.

The main phases of our protocol are the privacy-preserving group forming (Phase 1) and the privacy-preserving fair pick-up point selection (Phase 2). In Phase 1, the server groups the passengers based on the similarity of their intended journey without learning any information about the journey, i.e., in a privacy-preserving way. In this phase, the server categorizes the passengers into groups (a maximum) of m people, where m is the capacity of the autonomous cars (e.g., $m = 4$). In each group, the preferred pick-up locations of the passengers are likely to be close to each other, the passengers' journeys are roughly on the same path, and the date and time of traveling are similar.

In Phase 2, with the help of the server, the passengers of each group agree on a fair pick-up point in a private way. In other words, the passengers together find a fair location for an autonomous car to pick them up without revealing their initial locations to the server, to each other, or to the car.

In the following subsections, we explain the main phases of our protocol in detail.

5.1 Phase 1: Privacy-Preserving Group Forming

The server should group the passengers based on their preferred pick-up points, drop-off locations, and time of travel. However, the group forming should be done in such a way that the server does not learn any of the above information about the passengers' journeys. Moreover, the group-forming process should be fast enough to be used in real time. For these reasons, we propose a fast PSI protocol with the help of a secure keyed hash function with the secret key k. The key k is a one-time secret key that is created by a batch of

passengers for the purpose of performing one run of a PSI protocol. Therefore, Phase 1 of our protocol has two sub-phases: generating a shared secret key (Phase 1.a), and PSI (Phase 1.b).

5.1.1 Phase 1.a: Generating a Shared Secret Key. As mentioned before, each passenger has a pseudo-random identity and multiple anonymous tokens. Therefore, the ride-sharing system can grant access to valid passengers anonymously. We assume that the passengers let the server know about the city and date of their desired journeys anonymously. Then, the server inserts the passengers that want to travel in the same city (e.g., $c \in C$) and on the same day (e.g., d) in one set (e.g., $M_{c,d}$). For simplicity, let us assume that there are $n = 2^a$ passengers in $M_{c,d}$. The server proceeds by assigning a unique ordering number from 1 to n to each passenger in $M_{c,d}$, such that the passengers are ordered as P_1 to P_n.

In order to create a shared secret key k, the passengers P_1 to P_n enter a variant of the Diffie-Hellman key agreement protocol of Subsection 2.11 as follows:

(1) For $i = 1, ..., 2^{a-1}$, passenger P_{2i-1} and passenger P_{2i} engage in the 2-party Diffie-Hellman key exchange protocol and create a shared secret key $S_{2i-1,2i} = g^{S_{2i-1}S_{2i}}$ based on locally generated random values S_{2i-1} and S_{2i}. The pair of passengers P_{2i-1} and P_{2i} also compute a public key for the next round (i.e., $g^{S_{2i-1,2i}}$). Now, round 1 of the protocol is finalized.
(2) For $j = 1, ..., 2^{a-2}$, the passengers P_{4j-3} and P_{4j-1} exchange their respective public keys, and so do the passengers P_{4j-2} and P_{4j}. Then the four passengers P_{4j-3}, P_{4j-2}, P_{4j-1} and P_{4j} compute their shared secret key $S_{4j-3,...,4j}$ which is equal to $g^{S_{4j-3,4j-2}S_{4j-1,4j}}$. They also compute their public key $g^{S_{4j-3,...,4j}}$. Round 2 of the protocol is now finished.
(3) The protocol is continued in such a way that for any round $b < a$ and for $j = 1, ..., 2^{a-b}$, the passengers $P_{2^b \cdot j-(2^b-1)}$ and $P_{2^b \cdot j-(2^{b-1}+1)}$, the passengers $P_{2^b \cdot j-(2^b-2)}$ and $P_{2^b \cdot j-(2^{b-1}+2)}$, ..., and the passengers $P_{2^b \cdot j-2^{b-1}}$ and $P_{2^b \cdot j}$ exchange their corresponding public keys. Then, they all are able to generate a new shared secret key and a corresponding public key, and round b ends.
(4) The protocol is continued until round a, where a shared secret key $S_{1,\cdots,n}$ is computed by all passengers P_1 to P_n.

The shared secret key k that is required in Phase 1.b is equal to $S_{1,...,n}$. Figure 2 illustrates the above procedure for eight parties.

5.1.2 Phase 1.b: Private Set Intersection. The passengers in the set $M_{c,d}$ have created a shared secret key k that can be used with a keyed hash function H_k. Now each passenger finds the subregions corresponding to their desired pick-up points and chooses one of the eight directions, north, northeast, east, southeast, south, southwest, west, and northwest, as their trip direction. Then, the passenger computes H_k(subregion-number ‖ direction ‖ time) for all the acceptable combinations of regions, directions and times, and sends the hash values to the server.

After all the passengers have sent their sets of hash values to the server, the PSI protocol is performed by the server. If there are two or more passengers with the exact same hash digest(s), the server groups them together. The formation of the groups should be done in an optimized way. However, partitioning an arbitrary set

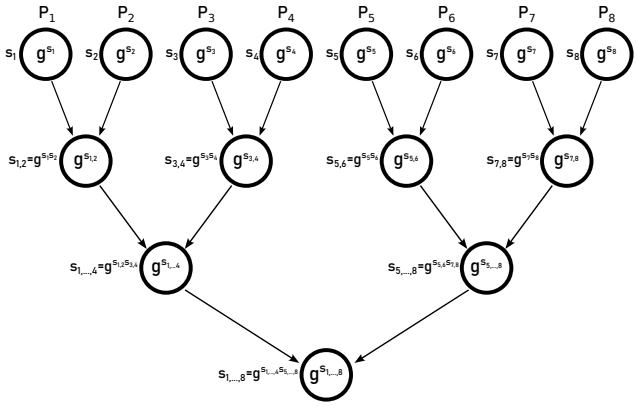

Figure 2: A Tree Diffie-Hellman Group Key Exchange for 8 Parties.

of passengers into groups of a fixed size based on their submitted list of preferences is a notoriously difficult problem. One way to approach this optimization problem is to reformulate it in a graph theoretic setting. We consider two sets of vertices, $\mathcal{P}$ and $\mathcal{H}$, where the set $\mathcal{P}$ represents the passengers and $\mathcal{H}$ represents the set of all submitted hashes. A Graph G is constructed such that there is an edge between $x \in \mathcal{P}$ and $y \in \mathcal{H}$ whenever y is a hash value submitted by the passenger x. Clearly, the graph G is bipartite, i.e., there is no edge between any two vertices in $\mathcal{P}$ (resp. $\mathcal{H}$).

To partition passengers to groups of size m, with a common submitted hash, it is enough to partition the vertices in $\mathcal{P}$ to disjoint subsets of size m, such that the vertices in every subset are all connected to at least one vertex in $\mathcal{H}$. However, partitioning bipartite graphs into sub-graphs consisting of m disjoint vertices all connected to one vertex is an NP-complete problem for $m \geq 3$, see for instance [40] and the references therein.

We detail the algorithm that we used for optimized grouping in Section 6 and leave further development on the topic of optimized group forming for future work.

Please note that the number of passengers in each group cannot be more than the capacity of the autonomous cars (m passenger). Moreover, the shared secret key is used only once to group the passengers. If grouping fails for a passenger, they re-enter Phase 1.a and compute a new shared secret key with different passengers. Please also note that if grouping succeeds for a passenger and they wish to arrange another ride-sharing, they require a new key.

The passengers that are successfully grouped together enter Phase 2 of our protocol as follows.

5.2 Phase 2: Privacy-Preserving Fair Pick-up Point Selection

In this subsection, we present a privacy-preserving fair pick-up point selection protocol for ride-sharing with autonomous cars.

Phase 2 of the protocol computes the geometric center (centroid) of the passengers' locations in a privacy-preserving manner. If the centroid is a convenient location for a vehicle to drive to, the passengers' pick-up point will be the centroid. Otherwise, the passengers

can find the closest parking spot to their centroid. That parking spot will be the passengers' pick-up point.

As we explained earlier, after executing the first phase of our protocol, the passengers who require similar ride services are grouped together. If the maximum capacity of a car is m passengers, then the number of passengers in each group is m', where $2 \leq m' \leq m$.

We recall from the setup phase of our protocol that the server distributed the public key p_k for the Paillier cryptosystem to all passengers. The passengers use the public key p_k to encrypt their initial location and consequently hide their locations from each other. However, if the passengers send their encrypted locations to each other via an insecure channel, the server can obtain the encrypted locations and decrypt them with its secret key.

Therefore, we require that the passengers in each group communicate with each other via an end-to-end encrypted channel. Thus, they require a shared secret key k', which is only known to them, to encrypt the channel. To create the shared secret key k', the server assigns another unique ordering number from 1 to m' to each passenger in each group. Then, the passengers enter in another Diffie-Hellman tree group key exchange protocol and together create the key k'. Then, the passengers utilize a key derivation function with the key k' to derive the AES key. Finally, AES encryption is used to encrypt the messages that the passengers send to each other in Phase 2.

The initial location of passenger P_i is denoted by (x_i, y_i). The passengers compute their centroid in a privacy-preserving way as follows:

(1) One of the passengers (e.g., P_1) picks two positive random numbers r_1 and r_2, creates an initial vector $W = (r_1, r_2)$, and sends this vector to other passengers P_2,...,$P_{m'}$ in the group.
(2) Each passenger P_i encrypts their initial location (x_i, y_i) with the server's public key p_k, obtaining $(E_{p_k}(x_i), E_{p_k}(y_i))$. The passenger first sends hash values computed from these two encrypted coordinates to other passengers. Once the passenger P_i has received such hash values from all other passengers, P_i sends the actual encrypted coordinates $E_{p_k}(x_i)$ and $E_{p_k}(y_i)$ to all other passengers.
(3) Each passenger computes

$$E_{p_k}(x_1)E_{p_k}(x_2)...E_{p_k}(x_{m'})E_{p_k}(r_1) = R_1$$
$$E_{p_k}(y_1)E_{p_k}(y_2)...E_{p_k}(y_{m'})E_{p_k}(r_2) = R_2.$$

(4) Passengers send (R_1, R_2) to the server.
(5) The server decrypts (R_1, R_2), checks that all the values sent by the passengers decrypt to same plain texts, and sends the results back to the passengers. The decryptions of R_1 ans R_2 are:

$$D_{s_k}(R_1) = x_1 + x_2 + ... + x_{m'} + r_1$$
$$D_{s_k}(R_2) = y_1 + y_2 + ... + y_{m'} + r_2$$

(6) The passengers compute

$$(D_{s_k}(R_1) - r_1)/m' = x_{\text{centroid}}$$
$$(D_{s_k}(R_2) - r_2)/m' = y_{\text{centroid}}$$

(7) The passengers can find the closest parking location to their centroid together, or accept the centroid as a convenient pick-up point.

Our protocol can be easily used for the case where an owner of an autonomous car wants to share a ride with other passengers. In this case, the owner registers their car in the server's ride-sharing system and enters the protocol as a regular passenger. After executing the protocol, when the time of the trip arrives, the car drives its owner to the selected pick-up point and collects other passengers.

A summary of our protocol is illustrated by Figure 3.

6 PERFORMANCE EVALUATION

In this section, we evaluate the time and bandwidth usage of our protocol. In our performance evaluation, we adopt the following parameters: The size of the modulo p in DH is 2048 bits, and there are $n = 2^{10} = 1024$ passengers participating in the DH-tree. Without loss of generality, we assume that the area of the city is 300 km^2. Therefore, there are 300 subregions in each city. Please note that the number of subregions does not affect the performance of our protocol. In our protocol, we group the passengers based on their pick-up points and the similarity of their destinations, which is independent of the size of the city. We utilize SHA-256 for the keyed hash function. The server groups the passengers into groups of a maximum of 4. The size of modulo N^2 in Paillier is 4096 bits. The time complexity of the protocol depends on the processor that a party (a passenger or a server) utilizes. For a realistic evaluation, we execute the protocol on a PC (laptop). The operating system (OS) is Windows 10, and it runs on an x86-64 Intel Core i5 processor clocked at 2.7 GHz with a 4 MB L3 cache. The passenger and the server use a single thread to perform the required computations.

In the setup phase, the server needs to compute several anonymous tokens for each passenger. The process of creating, distributing, and verifying the anonymous tokens is quick (in the order of μs for each passenger) [22]. Therefore, in this section, we focus on the cryptographic and optimization parts of our protocol, which are the most time-consuming parts.

6.1 Phase 1

In Phase 1, in order to create a shared secret key k, each passenger sends $\log n$ DH public keys to other passengers that are participating in the DH-tree. Therefore, each passengers sends $\log 2^{10} = 10$ public keys, which is 2.5 KB. To calculate the key k, each passenger computes $2 \log n$ modular exponentiation, and $\log n$ modular multiplications. The computation of modular multiplication is fast (less than 1 μs), thus, the costs of them is negligible. Each exponentiation takes 0.002 seconds, therefore, each passengers require 20(0.002) = 0.04 s to compute k. Computing the outcome of a keyed hash function is extremely fast (on average, 0.5 μs). In our protocol, each passenger only requires to compute a small number of hash values; therefore, the required time to compute the hashes is negligible. Each hash value is of size 256 bits. Therefore, the size of a passenger's set is less than 0.5 KB. The server performs PSI and forms the groups in an optimized way.

6.1.1 Optimized Grouping Algorithm. Our algorithm to group passengers based on their sets' intersection in an optimized way is presented in the following. The passengers' anonymous ids are denoted by P_1,...,P_n, and the set of their hash values denoted by $\{h_1, h_2, \cdots, h_e\}$, where each $h_i \in \{0,1\}^{256}$. Note that e is the number of hash values in a passenger's set, and therefore, it depends

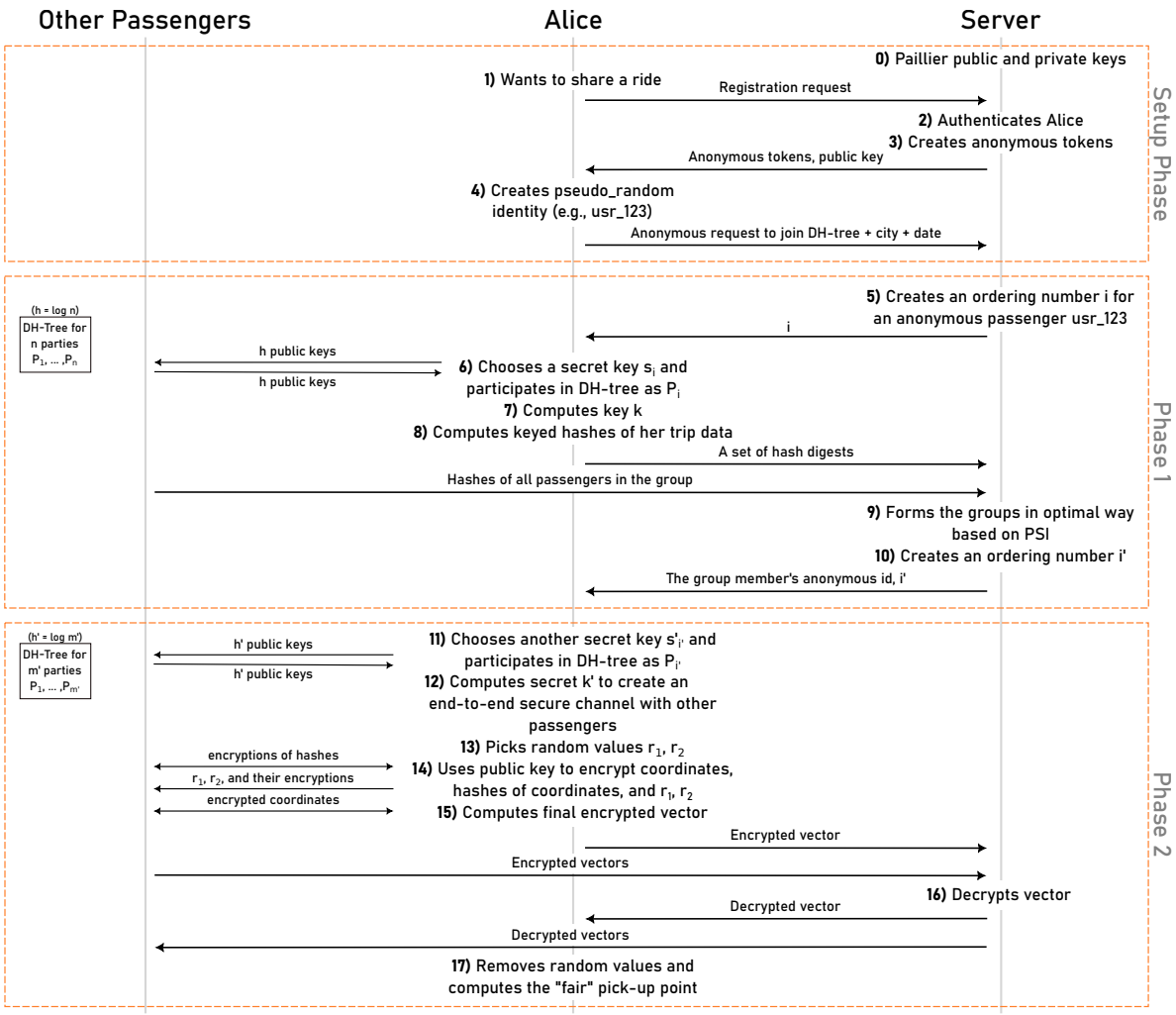

Figure 3: The figure shows a summary of the steps that a certain passenger (Alice) and a server take in our privacy-preserving ride-sharing protocol for autonomous cars.

on the flexibility of the passenger's request. For our setting, we assume that $1 \leq e \leq 4$.

(1) The server receives the passengers' hash values. Then, it creates one bucket for each unique hash and maps the buckets to the passenger. For instance, if hash value h_{11} is in the sets of the passengers P_{102}, P_{54}, and P_9, they will be mapped to bucket h_{11}.

After the above step, the passengers are categorized based on whether they have a certain hash value. However, these categories may have more than m passengers, and/or there are passengers that are in more that one category. To resolve these issues, the server performs the following steps.

(2) The server removes the buckets containing only one passenger, then computes the union of all passengers that are in the remaining buckets.
(3) If a passenger is not in the above union, they cannot be in any group. The server informs them that there is no possible ride-share for them. If they wish, they can re-enter Phase 1 later.

If possible, the server creates groups of maximum capacity (m). The server also prioritizes passengers that have more hash values. Please recall that more hash values in a set mean that the passenger is more interested in finding a ride-share. in their sets.

(5) The server computes the number of times that each passenger appears in different buckets. Let us call this number the *degree of a passenger*.
(6) The server starts grouping from buckets with the maximum number of passengers. If a bucket contains more than m passengers (e.g., m''), the server picks m passengers with the least amount of degree, and forms a group. Then the server subtracts 1 from the degree of the remaining $m'' - n$ passengers. The server also removes the m passengers that are grouped from all the other buckets.

(7) The server repeats the above step until the grouping is not possible anymore. Then, the algorithm ends, and if there are any passengers that are not in any group will be informed. They can try the protocol again later.

For executing the above algorithm, the server requires 0.25 seconds to group 1024 passengers. Thus, in Phase 1, a passenger's time complexity is 0.04 s, and the communication complexity is 3 KB. For the server, the required time to perform Phase 1 is 0.25 seconds, and the bandwidth usage is less than 0.25 KB as the server only sends two ordering numbers, i and i', and a maximum of 3 ids.

6.2 Phase 2

In Phase 2, the passengers of each group require to compute a key k'. If there are 4 passengers in each group, each passenger sends 4 public keys (in total 1 KB) and performs 8 modular exp., which takes 0.016 s. Computing the end-to-end encryption with AES is extremely fast, and therefore, the time complexity of it is negligible. Each passenger performs 2 Paillier encryptions that take 0.112 s and sends 2 ciphertexts of size 4096 bits to other passengers. One of the passengers requires performing an extra 2 encryptions to encrypt r_1 and r_2, that takes an extra 0.112 s. This passenger sends to all the other passengers r_1, r_2, and their encryptions, which are in total 1.5 KB. Each passenger performs 8 modular multiplications that take 0.0004 s and sends the final result of 4096 bits to the server. Therefore, the communication and computation complexities for a passenger in the worth case are 5 KB and 0.24 s, respectively. The server decrypts two ciphertexts in 0.33 s and sends 0.5 KB to each passenger (2 KB in total).

The total time and communication complexities of our protocol for the server (for a passenger) are 0.58 seconds (0.28 seconds) and 2.25 KB (8 KB), respectively. Table 1 compares the security, privacy, fairness, usability, and costs of our protocol with the prior art.

7 SECURITY AND PRIVACY ANALYSIS

We analyze the security and privacy of our privacy-preserving ride-sharing protocol in this section. We first examine the security and privacy of each part of the protocol against a semi-honest server. Then, we change the point of view and analyze each part of the protocol against semi-honest and malicious passengers.

Our model is based on the assumption that the server of the ride-sharing company is honest-but-curious (i.e., it is semi-honest). The company's business model is to provide ride-sharing services with autonomous cars while, at the same time, protecting the privacy of passengers. Moreover, as mentioned in Section 1, in order to avoid certain cyber-attacks, the company may not even want to store users' personal information. Hence, the company advertises its service as a privacy-preserving ride-sharing system. If at any point of operation, it turns out that the company does not follow the protocol or diverges from it (e.g., due to a successful cyber-attack), the company can harm its prestige and may lose its customers. Therefore, considering the server as a semi-honest party is a sensible assumption. However, we cannot similarly assume that all the passengers are honest or even semi-honest. Thus, in our analyses, we consider both semi-honest and malicious passengers.

In the setup phase, each passenger reveals the city and the date of their trip to the server while keeping their identity anonymous. Because the information is coarse-grained and is revealed anonymously, the privacy or security of the passenger is not violated.

Theorem 7.1. *After executing Phase 1.a, i) the semi-honest server does not learn the shared secret key k of n passengers nor any of the passengers' secret keys, ii) a semi-honest/malicious passenger cannot learn the secret key of other passengers.*

Proof. The security of Phase 1.a lies in the fact that the Diffie-Hellman tree group key exchange protocol is secure [6]. Therefore, the semi-honest server cannot learn any of the passengers' secret keys nor the final key k. Moreover, the passengers, semi-honest or malicious, cannot learn each other's secret keys. □

The passengers do not reveal their identity either to each other or to the server. Anonymity typically helps in launching Denial of Service attacks. However, a malicious passenger cannot perform an effective DoS attack against the server because the number of their anonymous tokens is not big enough to perform such an attack.

Theorem 7.2. *After executing Phase 1.b, i) the semi-honest server does not learn any information about the passengers' trip, ii) a semi-honest/malicious passenger cannot learn any information about other passengers' trips.*

Proof. The security of Phase 1.b relies on the security properties of keyed hash functions. The one-wayness of hash functions makes it impossible for the server to learn the function's input by investigating its output. Please note that because the server does not have the key k, it cannot perform a dictionary attack on the hash values. Therefore, the server does not learn any information about a trip from its corresponding hash value. As the communication between the server and an individual passenger is through a secure channel, the passengers cannot obtain each other's hash values. Note that they cannot obtain these with the help of the server because the server is assumed to be semi-honest. □

Theorems 7.1 and 7.2 show that by executing Phase 1, the server groups the passengers without learning the passengers' exact or approximate locations or times of the planned trips. Moreover, the passengers do not learn any information about each other's journeys. Therefore, running Phase 1 can successfully group the passengers in a secure and private manner.

Theorem 7.3. *After executing Phase 2, i) the server neither learns the passengers' initial locations nor their centroid, ii) a semi-honest/malicious passenger cannot learn the other passengers' locations.*

Proof. The security of Phase 2 relies on the security of the Paillier cryptosystem. The random values r_1 and r_2 in the vector W hide the locations of the passengers and the centroid from the server. Moreover, using the secure channel prevents the server from learning anything about the process of computing the components of vector W. It is clear that the semi-honest passengers do not learn each others' locations because their coordinates are encrypted, and they do not have the Paillier private key.

Because hash values of the encrypted coordinates have to be sent and received before the actual encrypted coordinates are shared, malicious passenger has to commit to the values they intend to send before they see anybody else's encrypted values. This implies

Table 1: Comparison of our protocol with the prior art. The order of time complexity is the total costs of pre-computation, off-line and on-line phases of the protocols. Each column presents a different property. The best performance in each column is remarked with bold font.

Properties / Protocols	Requires Extra Party	Works for Autonomous Cars	Anonymity	False Positive/Negative	The Order of Time Complexities	The Order of Communication Complexities	Secure Against A Malicious Passenger	Fair Pick-up Point
[32]	Yes	**Yes**	**Yes**	**No**	Minute	MB	No	No
[45]	Yes	No	No	**No**	Hour	MB	No	**Yes**
[18]	Yes	No	No	Yes	Minute	MB	No	No
[35]	Yes	No	**Yes**	Yes	Minute	MB	**Yes**	**Yes**
Ours	**No**	**Yes**	**Yes**	**No**	**Second**	**KB**	**Yes**	**Yes**

a malicious party cannot deviate too much from what an honest party would do in the protocol. They can still send something else instead of encryptions of their real coordinates, but this attack is just equivalent to lying about one's location in the first place. However, as a result, the malicious party only learns the centroid computed as if they would be in another location than where they actually is. □

Theorem 7.3 shows that by executing Phase 2, the passengers can arrange a fair pick-up point without learning each other's exact or approximate locations and without revealing any information about their trip to the server. Therefore, by running Phase 2, the passengers can successfully arrange the pick-up point in a secure and private manner.

In Phase 1, if the server starts to act maliciously, it can learn the key k either by colluding with one of the malicious passengers, or by participating in the key agreement protocol as a passenger. A malicious server who obtained the key k can compute all the possible combinations of the hash values $H_k(k \parallel \text{subregion-number} \parallel \text{direction} \parallel \text{time})$. Then, the malicious server can perform the dictionary attack and learn the input values. However, the passengers utilize the system anonymously in this phase. Thus, the information that is revealed to the server by a successful dictionary attack is still anonymous, so that it would not jeopardize the privacy of the passengers too much.

In Phase 2, if there are m' passengers in a group and $m' - 1$ passengers collude with each other and reveal their initial locations to each other, they can learn the location of the remaining passenger based on information about the centroid. Similar situation may occur when a malicious passenger registers to the system multiple times and obtains several anonymous identities. However, the passengers are grouped anonymously and based on an optimized matching algorithm. Therefore, the conspiring passengers have only moderate chances to be grouped together. In addition, they have no control on who would be the remaining anonymous passenger whose initial location they would learn. Because the attack requires using tokens, it has a notable cost, and the cost-benefit ratio for the attacker does not seem to be good.

In Phase 2, there is a special scenario where a malicious passenger may be able to cheat and move the centroid closer to their initial location. In this scenario, the malicious passenger knows that there cannot be any passengers in a certain area A that is close to the malicious passenger. For instance, the passenger is close to sea shore. Then, the malicious passenger enters Phase 2 with false coordinates that represent a point inside A. Thus, the malicious party manipulates the computation of the centroid such that the pick-up point would likely be closer to them[2]. However, if the other passengers are close to the malicious passenger, the pick-up point might fall in the area A (inside the sea!), and the passengers will realize that there is a malicious party in the group. In this case, the passengers send the encryptions of the coordinates of other passengers to the server. The server decrypts the coordinates and finds the malicious party. Please note that in the context of ride-sharing, a passenger books their prospective trip beforehand, hence uses an expected future location as an initial location. Therefore, in general, it does not make much sense to even try to verify whether the input location in Phase 2 is correct.

8 CONCLUSION

The paper presented our novel lightweight privacy-preserving ride-sharing protocol for autonomous cars. Our protocol is between a set of passengers and a server, and contrary to most of the previous related works, it does not require an extra party to provide privacy and security. Moreover, we have presented two novel privacy-preserving protocols for group forming (in Phase 1) and fair meet-up point detection (in Phase 2). In addition to ride-share organizations, the protocols of Phases 1 and 2 can be used in other use cases.

We suggested a modification to the DH-tree group key protocol that does not require a server. We also used an optimized group-forming algorithm in our protocol. Future work could try to investigate further into the topics of group key agreements and optimized group forming, as both of these topics have several use cases for autonomous cars.

We assume that the server is honest-but-curious. However, as we explained, it is not realistic to assume that all the passengers follow the protocol honestly. We analyzed the privacy and security of our protocol and proved that our protocol provides protection against malicious passengers.

We have implemented our protocol and evaluated its performance for a realistic scenario where 1000 passengers simultaneously request a ride-share in a city with an area of 300 km^2. The results of our experiments show that our protocol can be used in real life to organize ride-shares with autonomous cars in a privacy-friendly manner. In other words, the cost of our protocol is feasible in practice. Moreover, we compared several properties of our protocol with the prior art. Our comparison shows that despite the fact that we do not use any extra party in our protocol and we cover malicious passengers in addition to semi-honest ones, our protocol outperforms the prior art.

[2]Please note that the malicious party still cannot learn other passengers' initial locations.

REFERENCES

[1] Ulrich Matchi Aïvodji, Sébastien Gambs, Marie-José Huguet, and Marc-Olivier Killijian. 2016. Meeting points in ridesharing: A privacy-preserving approach. *Transportation Research Part C: Emerging Technologies* 72 (2016), 239–253.

[2] Saeed Asadi Bagloee, Madjid Tavana, Mohsen Asadi, and Tracey Oliver. 2016. Autonomous vehicles: challenges, opportunities, and future implications for transportation policies. *Journal of modern transportation* 24, 4 (2016), 284–303.

[3] Syagnik Banerjee. 2019. Geosurveillance, location privacy, and personalization. *Journal of Public Policy & Marketing* 38, 4 (2019), 484–499.

[4] Leila Benarous and Benamar Kadri. 2022. Obfuscation-based location privacy-preserving scheme in cloud-enabled internet of vehicles. *Peer-to-Peer Networking and Applications* 15, 1 (2022), 461–472.

[5] Igor Bilogrevic, Murtuza Jadliwala, Vishal Joneja, Kübra Kalkan, Jean-Pierre Hubaux, and Imad Aad. 2014. Privacy-preserving optimal meeting location determination on mobile devices. *IEEE transactions on information forensics and security* 9, 7 (2014), 1141–1156.

[6] Timo Brecher, Emmanuel Bresson, and Mark Manulis. 2009. Fully robust tree-Diffie-Hellman group key exchange. In *International Conference on Cryptology and Network Security*. Springer, 478–497.

[7] Ryan Browne. 2022. Uber investigates cybersecurity incident after reports of a hack on the company. https://www.cnbc.com/2022/09/16/uber-investigates-cybersecurity-incident-after-reports-of-a-hack.html

[8] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. 2018. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1802–1819.

[9] Joan Daemen and Vincent Rijmen. 2013. The Design of Rijndael AES – The Advanced Encryption Standard. (2013).

[10] Whitfield Diffie and Martin E Hellman. 2022. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*. 365–390.

[11] Fábio Duarte and Carlo Ratti. 2018. The impact of autonomous vehicles on cities: A review. *Journal of Urban Technology* 25, 4 (2018), 3–18.

[12] Per Hallgren, Martin Ochoa, and Andrei Sabelfeld. 2015. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 1–6.

[13] Lein Harn, Ching-Fang Hsu, and Bohan Li. 2018. Centralized group key establishment protocol without a mutually trusted third party. *Mobile Networks and Applications* 23, 5 (2018), 1132–1140.

[14] Lein Harn and Changlu Lin. 2014. Efficient group Diffie–Hellman key agreement protocols. *Computers & Electrical Engineering* 40, 6 (2014), 1972–1980.

[15] Carmit Hazay and Yehuda Lindell. 2010. A note on the relation between the definitions of security for semi-honest and malicious adversaries. *Cryptology ePrint Archive* (2010).

[16] Yuanyuan He, Jianbing Ni, Xinyu Wang, Ben Niu, Fenghua Li, and Xuemin Shen. 2018. Privacy-preserving partner selection for ride-sharing services. *IEEE Transactions on Vehicular Technology* 67, 7 (2018), 5994–6005.

[17] Jianbo Huang, Liang Chang, Long Li, and Xuguang Bao. 2020. An Adaptive Dummy-based Mechanism to Protect Location Privacy in Smart Health Care System. In *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE, 92–97.

[18] Junxin Huang, Yuchuan Luo, Ming Xu, Bowen Hu, and Jian Long. 2022. pShare: Privacy-Preserving Ride-Sharing System with Minimum-Detouring Route. *Applied Sciences* 12, 2 (2022), 842.

[19] Fortune Business Insights. 2022. *Ride sharing market size, share and covid-19 impact analysis.* https://www.fortunebusinessinsights.com/ride-sharing-market-103336

[20] Kimmo Järvinen, Ágnes Kiss, Thomas Schneider, Oleksandr Tkachenko, and Zheng Yang. 2018. Faster privacy-preserving location proximity schemes. In *International Conference on Cryptology and Network Security*. Springer, 3–22.

[21] Tobias Jeske. 2011. Privacy-preserving smart metering without a trusted-third-party. In *Proceedings of the International Conference on Security and Cryptography*. IEEE, 114–123.

[22] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. 2020. Anonymous tokens with private metadata bit. In *Annual International Cryptology Conference*. Springer, 308–336.

[23] Miltos Kyriakidis, Riender Happee, and Joost CF de Winter. 2015. Public opinion on automated driving: Results of an international questionnaire among 5000 respondents. *Transportation research part F: traffic psychology and behaviour* 32 (2015), 127–140.

[24] Donghe Li, Qingyu Yang, Dou An, Wei Yu, Xinyu Yang, and Xinwen Fu. 2018. On location privacy-preserving online double auction for electric vehicles in microgrids. *IEEE Internet of Things Journal* 6, 4 (2018), 5902–5915.

[25] Marco Maier, Lorenz Schauer, and Florian Dorfmeister. 2015. Probetags: Privacy-preserving proximity detection using wi-fi management frames. In *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 756–763.

[26] Macià Mut-Puigserver, Miquel A Cabot-Nadal, and M Magdalena Payeras-Capellà. 2020. Removing the trusted third party in a confidential multiparty registered eDelivery protocol using blockchain. *IEEE Access* 8 (2020), 106855–106871.

[27] Mahmoud Nabil, Ahmed Sherif, Mohamed Mahmoud, Ahmad Alsharif, and Mohamed Abdallah. 2019. Efficient and privacy-preserving ridesharing organization for transferable and non-transferable services. *IEEE Transactions on Dependable and Secure Computing* 18, 3 (2019), 1291–1306.

[28] Iynkaran Natgunanathan, Abid Mehmood, Yong Xiang, Longxiang Gao, and Shui Yu. 2018. Location privacy protection in smart health care system. *IEEE Internet of Things Journal* 6, 2 (2018), 3055–3069.

[29] Pascal Paillier and David Pointcheval. 1999. Efficient public-key cryptosystems provably secure against active adversaries. In *International conference on the theory and application of cryptology and information security*. Springer, 165–179.

[30] Benny Pinkas, Thomas Schneider, and Michael Zohner. 2018. Scalable private set intersection based on OT extension. *ACM Transactions on Privacy and Security (TOPS)* 21, 2 (2018), 1–35.

[31] Hua Shen, Mingwu Zhang, Hao Wang, Fuchun Guo, and Willy Susilo. 2020. A lightweight privacy-preserving fair meeting location determination scheme. *IEEE Internet of Things Journal* 7, 4 (2020), 3083–3093.

[32] Ahmed BT Sherif, Khaled Rabieh, Mohamed MEA Mahmoud, and Xiaohui Liang. 2016. Privacy-preserving ride sharing scheme for autonomous vehicles in big data era. *IEEE Internet of Things Journal* 4, 2 (2016), 611–618.

[33] David G Steer, Leo Strawczynski, Whitfield Diffie, and M Wiener. 1988. A secure audio teleconference system. In *Conference on the Theory and Application of Cryptography*. Springer, 520–528.

[34] Nalini Subramanian and Andrews Jeyaraj. 2018. Recent security challenges in cloud computing. *Computers & Electrical Engineering* 71 (2018), 28–42.

[35] Hongliang Sun, Linfeng Wei, Libo Wang, Juli Yin, and Wenxuan Ma. 2022. A Trusted and Privacy-Preserving Carpooling Matching Scheme in Vehicular Networks. *Journal of Information Security* 13, 1 (2022), 1–22.

[36] Iraklis Symeonidis, Dragos Rotaru, Mustafa A Mustafa, Bart Mennink, Bart Preneel, and Panos Papadimitratos. 2021. HERMES: Scalable, Secure, and Privacy-Enhancing Vehicular Sharing-Access System. *IEEE Internet of Things Journal* 9, 1 (2021), 129–151.

[37] Uber team. 2022. *Security update.* https://www.uber.com/newsroom/security-update/

[38] Hien To, Gabriel Ghinita, and Cyrus Shahabi. 2014. A framework for protecting worker location privacy in spatial crowdsourcing. *Proceedings of the VLDB Endowment* 7, 10 (2014), 919–930.

[39] James M Turner. 2008. The keyed-hash message authentication code (hmac). *Federal Information Processing Standards Publication* 198, 1 (2008), 1–13.

[40] René Van Bevern, Robert Bredereck, Laurent Bulteau, Jiehua Chen, Vincent Froese, Rolf Niedermeier, and Gerhard J Woeginger. 2017. Partitioning perfect graphs into stars. *Journal of Graph Theory* 85, 2 (2017), 297–335.

[41] Debby Wallner, Eric Harder, and Ryan Agee. 1999. *Key management for multicast: Issues and architectures.* Technical Report.

[42] Xiaofen Wang, Yi Mu, and Rongmao Chen. 2016. One-round privacy-preserving meeting location determination for smartphone applications. *IEEE Transactions on Information Forensics and Security* 11, 8 (2016), 1712–1721.

[43] Chung Kei Wong, Mohamed Gouda, and Simon S Lam. 2000. Secure group communications using key graphs. *IEEE/ACM transactions on networking* 8, 1 (2000), 16–30.

[44] Xi Xiao, Chunhui Chen, Arun Kumar Sangaiah, Guangwu Hu, Runguo Ye, and Yong Jiang. 2018. CenLocShare: A centralized privacy-preserving location-sharing system for mobile online social networks. *Future Generation Computer Systems* 86 (2018), 863–872.

[45] Haining Yu, Hongli Zhang, Xiangzhan Yu, Xiaojiang Du, and Mohsen Guizani. 2020. PGRide: Privacy-preserving group ridesharing matching in online ride hailing services. *IEEE Internet of Things Journal* 8, 7 (2020), 5722–5735.

[46] Yao Zheng, Ming Li, Wenjing Lou, and Y Thomas Hou. 2015. Location based handshake and private proximity test with location tags. *IEEE Transactions on Dependable and Secure Computing* 14, 4 (2015), 406–419.

[47] Xiaoyan Zhu, Haotian Chi, Ben Niu, Weidong Zhang, Zan Li, and Hui Li. 2013. Mobicache: When k-anonymity meets cache. In *2013 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 820–825.