DFPS: A Distributed Mobile System for Free Parking Assignment

Abeer Hakeem, Reza Curtmola, Xiaoning Ding, Cristian Borcea

Abstract—Cruising for vacant curbside parking spaces causes waste of time, frustration, waste of fuel, and pollution. This problem has been addressed by centralized solutions that perform parking assignments and communicate them to drivers' smart phones. These solutions suffer, however, from two intrinsic problems: scalability, as the server has to perform intensive computation and communication with the drivers; and privacy, as the drivers have to disclose their destinations to the server. This article proposes DFPS, a distributed mobile system for free parking assignment. DFPS solves the scalability problem by using the drivers' smart phones to cooperatively compute the parking assignments, and a centralized dispatcher to receive and distribute parking requests to the network of smart phones. The phones of the parked drivers in DFPS are structured in a K-D tree to serve parking requests in a distributed fashion. DFPS removes the computation from the dispatcher and substantially reduces its communication load. DFPS solves the privacy problem through an entropy-based cloaking technique that runs on drivers' smart phones and conceals drivers' destinations from the dispatcher. The evaluation demonstrates that DFPS is scalable and obtains better travel time than a centralized system, while protecting the privacy of drivers' destinations.

Index Terms—Parking assignment, cooperative system, mobile app, destination privacy.

F

1 Introduction

Finding vacant curbside parking spaces in congested areas of big cities (i.e., downtown) leads to traffic congestion, waste of time, driver frustration, waste of fuel, and pollution. This situation is particularly serious in developing countries where the increase in the number of vehicles outpaces the investments in parking facilities. Some governments try to mitigate this problem by deploying roadside sensors and parking guidance systems [2], [3], [4]. Unfortunately, the high initial and maintenance costs inhibit a widespread deployment of such solutions. Furthermore, because they are not designed to provide individual guidance to a specific parking space for each driver and can only show the same map of parking availability to all drivers, these solutions may lead to parking space contention and traffic congestion.

Targeting these problems, our prior work proposed a centralized parking assignment system [5] that relies on drivers to cooperatively maintain parking availability information and a centralized server to assign parking spaces. The system is cost-effective, as it does not rely on any sensing infrastructure. It reduces parking space contention and traffic congestion, because it uses a novel parking assignment algorithm to assign each driver to an available parking space close to her destination in a way that reduces the total travel time (i.e., the sum of the driving time to the parking space and the walking time from the parking space to the destination).

However, the system suffers from several limitations. First, the centralized server requires substantial computation (to manage and assign free parking spaces) and communication with the vehicles (to receive requests and send responses) in real-time. Thus, the sever can become a bottleneck for urban areas with many parking requests. Second, the parking assignment procedure is under risk of privacy violation. Specifically, the system requires drivers to disclose their destinations to the central server to ensure that parking spaces close to the destinations are allocated to them. Such information may be utilized to infer private information, such as the type and the frequency of visited places for each driver.

A few recent works [6], [7] have been presented solutions for privacy-preserving parking. The work in [6] mainly preserves the drivers' privacy by using anonymous credentials. However, hiding the drivers' real identities is not enough because the drivers still reveal sensitive information, such as current locations, destinations, and arrival times to the cloud server. Thus, the cloud server can identify the drivers based on their locations and destinations. PrivAV [7] extends anonymous authentication to support two-factor authentication to reduce the risks of vehicle theft while protecting the privacy. While location privacy is protected by the parking assistant, the driver's identity and destination may still be exposed.

To address the limitations of both these works and our prior centralized system, this article proposes DFPS, a distributed mobile system for free parking assignment. DFPS has two components: a mobile app running on drivers' smart phones and a dispatcher running at a server that enables cooperation among phones. The mobile apps on the

A preliminary version of this paper appeared in IEEE VNC 2017 [1].

A. Hakeem is with the Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia; and also with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102-1982 USA.
E-mail: ahakim@kau.edu.sa

R. Curtmola, X. Ding, and C. Borcea are with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102-1982 USA.
E-mail: {reza.curtmola, xiaoning.ding, borcea}@njit.edu

phones form a distributed system that manages and assigns free curbside parking spaces. This substantially reduces the communication and computation overhead on the central server. The parked drivers in DFPS are structured in a K-D tree [8] based on their locations. This structure allows high efficiency in serving new parking requests through parallel processing. The K-D tree also provides for localized distributed computation and communication, which makes DFPS scalable.

To conceal drivers' destinations from the central dispatcher, DFPS uses a novel entropy-based spatial cloaking technique, where each driver can entertain parking assignment services without revealing her real destination and without seeking help from any centralized third party. In addition to spatial cloaking [9], [10], techniques such as location perturbation and obfuscation [11] and dummy location [12] can also solve the problem of location privacy protection. However, they cannot be used in our settings because they may lead to parking assignments far from destinations [13], [14]. The basic idea of our entropy-based cloaking technique is that each driver submits her parking request with a cloaked region as her destination, instead of her real destination. Specifically, for each real destination, the entropy-based cloaking technique selects the nearest neighbouring destinations to construct a cloaked region, which contains both the real destination and the selected new destinations. The cloaked region must satisfy a k anonymity privacy requirement: in addition to the real destination, the region must have at least another k 1 possible destinations that are not distinguishable from the real destination. When constructing a region, an entropy of distance method [15] is employed to avoid the clustering problem (i.e., multiple destinations clustered in a small area, making it easier for an attacker to exploit the driver's destination). The method selects k 1 destinations that are evenly distributed to form the cloaked region. The technique also requires that the cloaked region contains at least a minimum number of available parking spaces to ensure that a parking space close to the real destination is likely to be available when the driver approaches it.

DFPS does not assume that all drivers use our system. It relies on subscribed drivers to submit observation reports regarding the parking spaces occupied by drivers that are not part of our system. DFPS avoids allocating the reported spaces for a period of time proportional with the age of the observation reports; then, it reconsiders these spaces.

We have evaluated DFPS through extensive simulations using SUMO [16] and PeerSim [17], a real map of a part of New York City with 1024 parking spaces, and as many as 768 drivers looking for parking. The results show that DFPS scales well. In particular, it eliminates all computation from the centralized dispatcher and reduces its communication load by a factor of two. DFPS can decrease the average travel time by 26% when compared with the centralized system, in addition to not disclosing the drivers' destinations to the dispatcher.

The rest of this article is organized as follows. Section 2 presents the system model of DFPS together with its scalability and privacy goals. Section 3 describes the entropy-based cloaking technique to conceal drivers' destinations in the parking requests. Subsequently, we explain the K-D tree structure of the cooperative drivers in Section 4. Section 5 presents the parking assignment algorithm. The privacy analysis and performance evaluation are shown in Sections 6 and 7, respectively. Section 8 reviews related work, and Section 9 concludes the article.

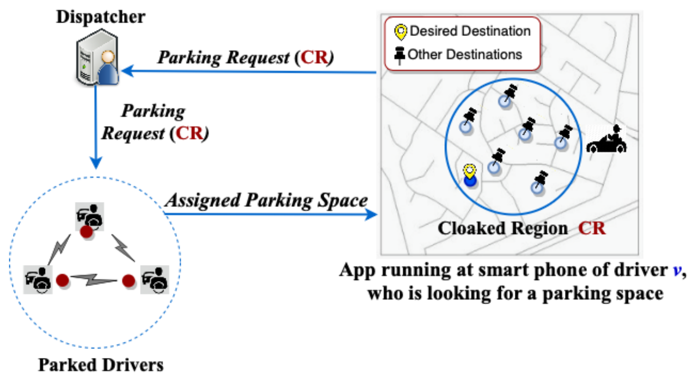


Fig. 1: DFPS System Architecture.

2 System Overview

This section presents an overview of DFPS, with emphasis on its design goals and system/threat models.

2.1 Design Goal

Our design goal is to propose a solution that solves two intrinsic problems in a centralized system for parking assignment: scalability and privacy. In a centralized system, the server responsible for communication with drivers and parking request processing could be a bottleneck. Also, processing parking requests requires drivers to disclose their desired destinations to the server. This could lead to major privacy concerns for the drivers and may prevent this solution from being adopted. Thus, the following system objectives should be considered to achieve the design goal:
Scalability: The parking assignment process should be distributed and performed efficiently on the drivers' smart phones rather than at the central server to reduce the computation and communication burden on the server and to achieve better scalability.

Privacy: The drivers' destinations should be protected. When a driver submits her parking requests through the central server, the server cannot identify her desired destinations and cannot link different parking requests submitted by the same driver at different times.

2.2 System Model

Under the aforementioned system objectives, we propose the following system model to implement DFPS (Figure 1). The system consists of the following two entities: a parking app running on the drivers' smart phones and a parking assignment dispatcher running on a server.

Drivers in the system are divided into three categories, based on their status: (1) drivers who are looking for parking spaces, (2) parked drivers, and (3) departing drivers. The drivers in the first category determine their cloaked regions, submit their requests along with the regions, and then wait to receive parking spaces while heading to their destinations. Once parking assignments are made, they drive to the parking spaces. When a driver has parked (second category), she cooperates with other parked drivers to provide on-demand parking service for the drivers looking for parking spaces. Departing drivers report when they leave the system such that DFPS can update the status of the parking spaces. Drivers use their smart phones to communicate with the system and with other drivers through the Internet (e.g., over 5G).

Dispatcher is a cloud server that receives parking requests from the drivers looking for parking spaces, and forwards them to parked drivers who perform parking assignments. The dispatcher also serves as a bootstrapping unit to initialize the whole system. During the system's adoption period, when a few number of parked drivers provide the parking service, the dispatcher could share with the drivers parking availability information derived from historical parking statistics or real-time sources (e.g., street images from video cameras).

The high-level workflow of the DFPS architecture in Figure 1 illustrates the life-cycle of a parking request in the system, from generation to completion. When a request is generated, an entropy-based cloaking technique is used to protect privacy. Specifically, each driver has its own privacy profile that specifies the desired level of privacy. A privacy profile includes two parameters, a pseudonym and an integer k. The pseudonym ensures the pseudonymity of the parking request by concealing the driver's identity and k indicates that a driver wants her destination to be k anonymous, i.e., indistinguishable among k 1 neigh-bour destinations. In other words, the driver wants to find a cloaked region that includes at least k 1 neighbour destinations to conceal her desired destination from the dispatcher. The larger the value of k, the more strict the privacy requirement is. The entropy-based cloaking technique generates cloaked regions (detailed in Section 3), which are sent along with parking requests.

After a parking request with a cloaked region is generated at the app on the smart phone, a question that arises is when the request should be sent and parking spaces are assigned. Assigning parking spaces when drivers are far away from their destinations increases the likelihood that the assigned spaces are taken by drivers who are not subscribed to DFPS (i.e., unsubscribed drivers) and reduces the utilization of parking spaces. Assigning parking spaces too late may result in an increase in driving time and bad user experience, especially when the system may not be able to find a parking space close enough to the destination. Therefore, the cloaked regions serve a dual-purpose. As discussed, they provide privacy protection for drivers' destinations. In addition, DFPS uses them to determine when a driver's parking request has to be sent to the dispatcher: a parking request is sent to the dispatcher when the driver is about to reach the cloaked region.

In DFPS, the sizes of cloaked regions are determined by the following few factors: 1) the need to preserve privacy: larger regions tend to contain more destinations and preserve privacy better; 2) the need for finding optimal parking spaces for drivers: with more available parking spaces, larger regions tend to provide more opportunities for DFPS algorithms to perform parking assignment optimizations; and 3) the need to avoid interference from unsubscribed drivers: with smaller regions, the contention of unsubscribed drivers for assigned parking spaces is less intense.

All the requests are sent to the dispatcher first. Then, the dispatcher forwards them to parked drivers. The work required to serve requests is divided among parked drivers. This protects a driver's destination at the dispatcher side, minimizes the workload on the dispatcher, and maximizes the system scalability. We partition the whole area of a city into regions, and assign a parked driver to manage a region and allocate parking spaces in that region. We use a structured overlay network to organize parked drivers and the regions that they are in charge of. Once a driver parks in her assigned space, she is assigned to a region, joins the overlay network, and starts to serve parking requests from the drivers who are looking for parking spaces in her region.

When creating an overlay network based on the phones of the drivers that act as parking managers, we borrow ideas from peer-to-peer (P2P) networks. There are many successful real-life examples of such networks, but we list here just two: Skype, in its original version, and more recently blockchain on smart phones [18]. Like in traditional P2P networks, the parking managers share their resources in exchange for access to the service. If the service is valuable, people are willing to participate (e.g., Skype). Of course, a parking solution can be implemented in a centralized way, but this paper demonstrates how a decentralized solution achieves better scalability and privacy.

Similar to P2P networks, DFPS needs to avoid free riders (i.e., drivers who submit requests, but never participate in the overlay network). The drivers must participate in the overlay network as parking managers in order to benefit from the free parking assignment service. After installing the app on their phones, the drivers are allowed a few "free" requests. After that, they need to prove that they worked as parking managers. The DFPS app on the phone records how many assignments it performed. The policy could be that a new request can be submitted after performing N assignments. If the driver turns off the app after each time it parks and does not participate in the overlay network, the app will not allow new requests after a while.

The software for DFPS and the server for its dispatcher can be provided and maintained by city municipalities, as part of smart city initiatives. DFPS can reduce traffic congestion and pollution in the cities by reducing the time wasted by drivers looking for free parking spaces. A number of cities have already deployed sensors to help drivers find parking [2], but these solutions are costly. DFPS does not require such an investment; its hardware cost is just maintaining the server for the dispatcher. As shown in the paper, the dispatcher performs just simple coordination tasks, and thus it does not require substantial resources. Therefore, many cities may prefer a cheaper, scalable, and privacy-preserving solution that is easy to deploy on drivers' smart phones.

2.3 Threat Model and Privacy Goals

Threat Model. We assume that the dispatcher is honest-but-curious, i.e., it follows the protocol correctly, but may try to analyze the information available in the system in order to find private information about the drivers. For example, the dispatcher could be interested in learning personally identifiable information about the drivers such as their identity and their destinations. The dispatcher may also be interested in linking different parking requests made by the same driver, which could reveal over time private information about the drivers. The dispatcher could learn the type and the frequency of visited places for each driver, which most drivers would prefer to keep private. For example, if a driver sends frequently a request for a parking space close to a hospital building, the dispatcher may infer that this driver may have certain health conditions. In another example, the sexual orientation of a person could be inferred if they park often close to an LGBTQ bar. Similar inferences can be made regarding many other personal aspects that people may want to keep private. We also assume that parked drivers do not collude with the dispatcher (i.e., share information) or act maliciously in any other ways in order to break the privacy of other system users.

A determined attacker could potentially be willing to expend resources and have a physical presence at a location in the real world in order to determine a driver's destination. We assume that the attacker's ability to execute such an attack is limited because it is expensive; whereas the attacker may be able to cover a small number of locations, it is not feasible nor cost effective to execute such an attack at scale.

The role of the dispatcher could be played by a telecommunication company which can also act as a service provider (e.g., AT&T). This gives an unscrupulous dispatcher the chance to identify drivers' real identities as well as their destinations by tracking their location. In this work, we do not consider such a strong adversary.

Finally, we assume that the communication between entities in the system is protected using standard security mechanism against interference from external adversaries.

Privacy Goals. To mitigate the aforementioned privacy threats, DFPS needs to achieve the following privacy goals:
Driver identity privacy: Drivers should not have to reveal personally identifying information (such as their real identity). This helps preserve the privacy of the drivers as related to their real-world persona, which may otherwise act as a deterrent against using the system.

Unlinkability of parking requests: Given two parking requests, the system should not be able to tell if they are made by the same driver or by different drivers. This prevents building a profile of a driver over time, which may eventually lead to revealing a driver's real-world identity and their parking request history.

Parking destination privacy: Given a parking request, the system should not learn the real destination of the parking request. This also prevents the system from learning information about a driver's real identity by correlation with the destination of parking requests.

There are well-known techniques to address the first two privacy goals, for example drivers can use a randomly chosen pseudonym for each parking request. Also, to prevent the dispatcher from linking multiple requests from a driver's IP address, drivers send requests through a network anonymizer [19]. Our primary focus in this article is to achieve the third goal, that of parking destination privacy.

3 Privacy-Aware Parking Request

Protecting driver's destination privacy against the attacks at the dispatcher side is one of DFPS's goals. Knowing the driver's destination could disclose sensitive information (e.g., interests, the most visited places, etc.), which can cause privacy breaches, even if the driver uses a new pseudonym with every parking request [20], [21]. Privacy concerns in location-based services exist on two fronts: location privacy and query privacy [22], [23]. Spatial cloaking is a privacy protection mechanism used to protect both location and query privacy [24], [25]. The main idea is to blur a piece of location information by replacing the exact location with a cloaked region that contains the location and some other similar locations so as to satisfy the user's privacy requirement, e.g., k anonymity [20] (i.e., the cloaked region contains at least k users). This mechanism is widely used because of its high efficiency. However, it must be improved to be usable in DFPS due to two drawbacks in our problem settings. First, it fails to consider the distribution of destinations, and a cloaked region with many destinations clustered together could be very small. Thus, the user privacy may be negatively affected. Second, it requires additional system resources from trusted third parties (e.g., location anonymizer [24], [26], peer users [27]). This makes the system more complex and may bring additional vulnerabilities.

DFPS proposes an entropy-based cloaking technique that overcomes these drawbacks. DFPS generates cloaked regions that satisfy the privacy requirements of drivers by achieving k anonymity. The construction of a cloaked region satisfies four requirements: (1) the cloaked region is not centered around the actual destination to avoid "center-of-cloaked-region" attack [28], [29]; (2) the destinations in the region are not very close to each other to avoid the clustering problem; (3) the region has enough open parking spaces to ensure the effectiveness of the parking assignment algorithm; and (4) the process of generating the region does not rely on trusted third-party servers.

These requirements are considered in the four phases of the cloaked region construction: (a) basic k-anonymity region creation; (b) center adjustment; (c) entropy-distance adjustment; and (d) parking-availability adjustment. Figure 2 illustrates these phases, with a running example. In the figure, 13 destinations are represented with solid circles, and d_v is the real destination of the driver. We assume that the required k anonymity level is four, i.e., $k = 4$. The operations in these phases are as follows:

Phase (a): Basic k-anonymity region creation: This phase starts by applying the k nearest neighbours algorithm (KNN) to determine the k 1 nearest destinations to dv. Then, as shown in Figure 2(a), it defines a circular region A centered at d_v that encompasses the k 1 nearest destinations (i.e., $NumDest(A)$ k).

Phase (b): Center adjustment: For the different requests with the same destination, phase (a) always generates the same cloaked region. This makes the solution vulnerable to "center-of-cloaked-region" privacy attack — an attacker could guess that the destination closest to the center of the cloaked region is the real destination of the driver. Thus,

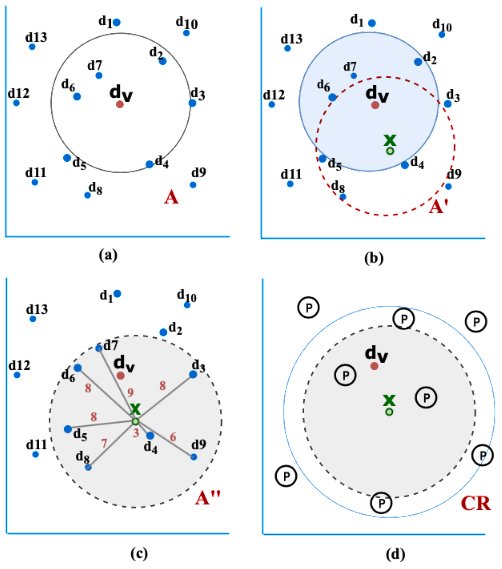


Fig. 2: Entropy-based cloaking technique. (a) Basic k-anonymity region creation; (b) Center adjustment; (c) Entropy-distance adjustment; (d) Parking-availability adjustment.

we propose an adjustment scheme for the region's center. As shown in Figure 2(b), DFPS randomly selects a point x in A and finds a set of $k-1$ nearest destinations to x. Phase (a) guarantees that x is not too far from dv. DFPS then constructs a new region A', which is centered at x and contains the k-nearest destinations (including the real destination d_v).

To further increase the parking destination privacy, DFPS could use the solution proposed in [30] which adds random noise when shifting the center to guarantee that point x is not too close to the real destination d_v.

Phase (c): Entropy-distance adjustment: Region A' satisfies requirement (1), but not requirements (2) and (3). It is possible that the distance between the $k-1$ neighboring destinations in A' is small, making it easier for an attacker to infer the driver's destination. Knowing the driver's identity helps an attacker to physically inspect which destination the driver is located at. To prevent this type of attack, the cloaked region may be further expanded while keeping x as its center. This adjustment is conducted using the entropy of distance method [15]. If the destinations of A' are located on fewer than $k/2$ segments (i.e., they are too close to each other), DFPS selects a different set of $k-1$ neighboring destinations and ensures that: (i) the selected $k-1$ destinations are evenly distributed in the new, expanded region; and (ii) the new region is not expanded too much.

We first use the KNN algorithm to find 2k nearest neighbour destinations around x. The area containing all these destinations is inevitably large. Thus, we examine the distribution of these 2k destinations to determine a smaller region A'', which contains $k-1$ destinations that can hide well the location of the real destination, based on the entropy values defined below. The number 2k is experimentally determined and provides a good trade-off between performance and overhead. In a practical deployment, DFPS can allow users to adjust a threshold for this method (in addition to the anonymity parameter k).

DFPS selects the $k-1$ destinations in this phase such that these destinations have a large entropy value, indicating that they are evenly distributed in the region. For this, DFPS forms m destination groups, each of which has $k-1$ destinations randomly selected from the 2k destinations. For each group, DFPS calculates two values: 1) an aggregated distance D, which is the sum of the distances between each of the $k-1$ destinations and the center x, and 2) an entropy value H, which measures the uncertainty of the destinations (more uncertainty indicates more even distribution). The entropy of a group is calculated using the following equation.

$$H(n) = -\sum_{i=1}^{k-1} \omega_{n_i} \log \omega_{n_i} \qquad (1)$$

In the equation, ω_{n_i} is the the weight of the destination i, which is defined as follows:

$$\omega_{n_i} = \frac{dist(x; d_i)}{\sum_{j=1}^{2k} dist(x; d_j)} \quad (i = 1; \ldots; k-1) \qquad (2)$$

Among m groups, we select the group to determine the cloaked region A'' as follows. If the D values of the groups are equal, we select the group whose destinations are more evenly distributed (larger entropy value). Otherwise, we select the group with the largest D value.

Figure 2(c) depicts how a cloaked region is expanded using the entropy of distance method. The area of A' is first expanded to cover 2k=8 nearest destinations including dv. Lines between the destinations indicate the spatial neighbor relation between each destination and x, and the values marked on the lines indicate the distances. Assume that three (m=3) groups of destinations are randomly selected, $G1=\{x; d_5; d_8; d_9\}$, $G2=\{x; d_3; d_5; d_6\}$, and $G3=\{x; d_V; d_7; d_8\}$. According to formula 1, the entropy on distances is obtained. The total distance of destinations is also calculated for each group. The total distance of destinations in G1 is less than those of G2 and G3. The total distances of G2 and G3 are almost equal; however, the destinations in G2 are more evenly distributed than the ones in G3. According to the entropy of distance method, G2 is chosen in this example. Then, the cloaked area A'' is computed based on group G2. The cloaked area shown in the figure is after the shrinking done according to G2.

Phase (d): Parking-availability adjustment: this phase happens when the DFPS app at the driver sends a parking request for region A'' to the dispatcher. As explained, A'' guarantees k-anonymity destination protection (i.e., the dispatcher cannot tell the real destination from $k-1$ other destinations, which are not clustered together). Upon receiving the request, the dispatcher has to decide whether to forward it to the peer-to-peer network of parked drivers as is or to expand the region further. The decision is based on the parking availability in A''. The dispatcher dynamically maintains the spatial distribution of parking availability in the whole city as described in [1]. If the number of available spaces in A'' is less than a threshold P_{min}, the dispatcher expands the region to encompass the closest available parking spaces to center x that are not within A'' until the number of parking spaces in the region reaches P_{min}. This is done to avoid the parking space scarcity problem. In DFPS, there is always a chance that an unsubscribed driver will take a parking space before the driver assigned to that space arrives there. The probability of this situation to happen is much higher when there is parking space scarcity in a region. Expanding the region up to P_{min} parking spaces reduces the likelihood of such a situation.

Figure 2(d) depicts how the cloaked region A^{00} is expanded to satisfy the parking availability condition. We assume that $P_{min} = 3$, and the number of available parking spaces within A^{00} is two. A^{00} will be expanded to contain at least three available parking spaces.

The Dispatcher over-estimates the number of available parking spaces as it uses only information from the DFPS parked drivers. This is because unsubscribed drivers may take parking spaces presumed to be available by our system (this problem is addressed in our previous work [5] based on keeping track of spaces occupied by unsubscribed drivers and on avoiding assigning these spaces for a period of time). Thus, the value of P_{min} should be reasonably large to tolerate this over-estimation. This value is determined experimentally as a function of the number of destinations and the total number of spaces in the region in order to provide a good trade-off between destination privacy and system optimization.

An alternative solution that avoids phase (d) is to make the cloaked region significantly larger than A^{00} before submitting the request. In this way, there will be a high chance to find parking spaces available in the region. However, this alternative solution may assign the parking spaces too early (when the drivers are far away from their parking spaces), and the parking spaces may be taken by unsubscribed drivers before the subscribed drivers arrive there. Thus, we choose to apply phase (d) instead of this alternative solution.

4 Overlay Network Structure & Operation

DFPS uses a peer-to-peer overlay network to organize parked drivers and the regions managed by the parked drivers. The organization of parked drivers needs to satisfy the following requirements:

Scalability: The drivers must be organized in a scalable way to share the workload effectively.

Fast Routing: Given a request, DFPS must quickly identify the driver managing the region where the parking space is requested.

Adaptability: The overlay network must adapt quickly and with low overhead to high churn (i.e., parked drivers coming and going frequently).

4.1 K-D Tree Network Structure in DFPS

To meet the above requirements, DFPS organizes the overlay network of the parked drivers as a K-D tree. A K-D tree is a k-dimensional binary search tree for partitioning and spatially indexing data distribution in a k-dimensional space [8], [31]. A node in the K-D tree is associated with three types of information: a value, a rectangular representation (i.e., a region) containing a set of data points, and the coordinates of these data points.

Each node in the K-D tree represents a region. The region corresponding to a parent node is divided into sub-regions corresponding to the children of that node. The

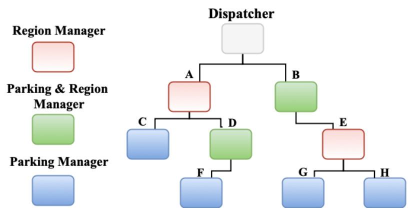


Fig. 3: The roles associated with nodes in the K-D tree.

locations of the parking spaces in a region are represented as data points associated with the node for that region. The node's value is the location of the first driver parked in the corresponding region when the region and the node are created. For brevity, we call this value the location of the node.

DFPS associates a parked driver with each tree node. The tasks of forwarding parking requests and allocating parking spaces, as well as the data structures required to manage these tasks, are associated with nodes. Since the nodes follow a tree structure, DFPS can manage the tasks and data structures in a hierarchical way, which leads to good scalability.

There are two roles that may be associated with a node in the K-D tree as shown in Figure 3: 1) region manager which forwards parking requests, divides a region into two sub-regions when necessary, and assigns sub-regions to drivers that park in these sub-regions; 2) parking manager which assigns parking spaces within the region associated with the node. Depending on its position in the tree, a node may have one or both of these roles. A leaf node acts only as parking manager for its region (i.e., nodes C, F, G, and H). An internal node that has two children acts only as region manager (i.e., nodes A and E). An internal node that has only one child acts as parking manager for the sub-region that is not covered by the child node, and it acts as a region manager by forwarding requests to its child or by assigning the sub-region not covered by the child to a driver that parks in that sub-region (i.e., nodes B and D).

Since parking space allocation is handled by the phones of parked drivers, we also refer to the parked drivers as the region manager or the parking manager of the regions corresponding to the node (depending on the node's role).

4.2 Joining and Departing K-D Tree

The K-D tree grows dynamically when more drivers park. When a driver informs its parking manager that it had parked, the parking manager creates a sub-region and a new node for the sub-region. Then, it attaches the new node as the child of the node it manages and assigns the newly parked driver to manage the new node and the parking spaces in the corresponding sub-region. Thus, the newly parked driver becomes the parking manager for these parking spaces. Over time, it also becomes a region manager when it has to divide this sub-region.

When a parking manager creates sub-regions, it divides its region based on the location of its parking space. This design has two advantages over evenly splitting the entire region among all the parked drivers. First, it helps to evenly distribute parking requests to parking managers. Due to hot spots, the destinations of drivers are not distributed

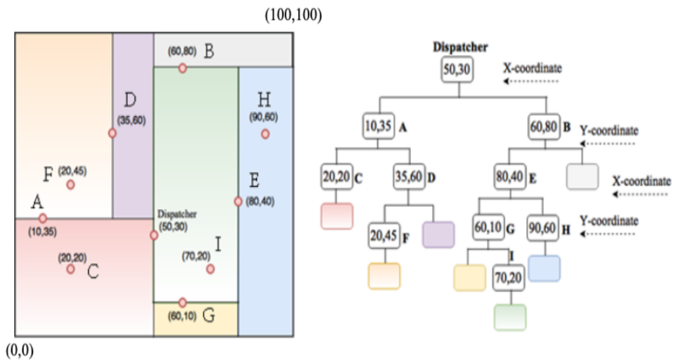


Fig. 4: Example showing how a K-D tree grows when drivers A, C, B, D, E, F, G, H, I park in a 100x100 2D region. Dots in the sub-regions represent parking spaces and letters represent parked drivers. The numbers in each box of the K-D tree are the 2D coordinates of the parking space of the corresponding driver.

evenly in the region. If the entire region is evenly partitioned among parking managers, the drivers managing hot areas might be overloaded. In contrast, the method employed by DFPS guarantees that more parking managers are assigned to hot areas than cold areas. Second, sub-regions are created dynamically within a small region where the driver parks. Other regions are not affected. This minimizes the changes to regions and the associated overhead, such as exchanged messages.

Each node in the K-D tree is represented as a value (the parking location of a parked driver, i.e., its x, y-coordinates) and a rectangular region, whose parking spaces will be managed the parked driver associated with this node. The region of the parent node is split into two rectangular sub-regions based on the K-D tree splitting rule: all dimensions are selected alternatively, e.g., x, y, x, and then y; the split is at the parking location of the driver managing the region, along the selected dimension. After the split, the driver is on the boundary of the two sub-regions.

Figure 4 shows a 100x100 2D region as an example of how the regions and the location points of the K-D tree are matched. Initially, the region is managed by the dispatcher, whose location is set to (50, 30). The region is split when driver A arrives and parked at (10, 35). Suppose the x dimension is selected first. The region is split along the line x=50. The sub-region containing A is now managed by A. When another driver (i.e., C) arrives and parks in this sub-region, it is further split. This time, y dimension is selected, and the split is at A's y coordinate (i..e, y=35). For a location point, its region is the smallest region containing the point. Since all the regions are rectangular, it can be easily determined by comparing its coordinates against the x and y values of the region boundaries.

When parked drivers leave their parking spaces, the tree nodes associated to those drivers must be updated. For each node managed by a leaving driver, if the node is a leaf, the node is deleted from the K-D tree and its parent node (i.e., the corresponding driver) takes over the parking space allocation task associated with the node. If the node is an internal node, one option is to apply existing solutions for deleting K-D tree internal nodes [31]. However, because the sub-trees rooted at the internal node's children must be re-organized to form another sub-tree, these solutions can be very expensive and may cause considerable overhead, especially when the sub-trees are large.

Instead of deleting an internal node, DFPS assigns the node to another parked driver. DFPS considers the following two situations:

The node is the parking manager of the region containing its location (the node value). In this case, the node has only one child node. DFPS will find the driver who is managing the child node, and assign the node to this driver.

The location of the node is managed by another node, i.e., another node is the parking manager of the region containing its location. DFPS will first locate the parking manager, then assign the node to the driver of the parking manager. In the example shown in Figure 4, when E leaves, driver H will be asked to take care of the node of E, since H is in charge of the parking space allocation in the region where E parked. Thus, later on, when the parking space is re-allocated by H to another driver K, K can be assigned to manage the node.

4.3 Request Forwarding

Each parking request, which includes the pseudonym of the driver and the cloaked region computed at the driver's phone, is forwarded down the tree from the root (i.e., dispatcher) until it reaches the corresponding parking manager, which will assign a parking space in its region. If privacy were not considered, then request forwarding will simply use the K-D tree to reach the parking manager that manages the region including the destination. However, when we consider privacy, the cloaking region may intersect several regions managed by different parking managers. This problem is illustrated by the protocol shown in Algorithm 1.

Algorithm 1 Parking request forwarding procedure

```
1: Upon node n receiving a parking request (v_pseudonym, CK)
2: if (n is not a parking manager) then
3:     Forward the request to the children whose regions intersect with CK
4: else if (n has no children) then
5:     //Region(n) is the region managed by n when acting as parking manager
6:     Send the coordinates of Region(n) to the driver
7: else
8:     //n is a parking manager and a region manager
9:     if (CK \ Region(n)) and (CK \ Region(child(n))) then
10:        Send the coordinates of Region(n) to the driver and forward the request to the child
11:    else if (CK \ Region(n)) and (:(CK \ Region(child(n)))) then
12:        Send the coordinates of Region(n) to the driver
13:    else
14:        Forward the request to the child node
15:    end if
16: end if
```

Upon node n receiving a parking request along with the cloaked region CK, n's state is examined to determine if it can answer this request or forward it down the tree. If n is a region manager, then the request is forwarded to the children whose regions overlap with CK (lines 2-3). CK could overlap the region of one child or both. If n is a parking manager and its region overlaps with the CK, then the region coordinates are sent to the driver (lines 4-6). If n is both a parking manager and a region manager, then both its region and its child's region have to be examined (lines 7-8). If CK overlaps with both regions, n sends the coordinates of its region to the driver and forwards the request to the child (lines 9-10). If only n's region overlaps with CK, n sends the coordinates of its region to the driver (lines 11-12). Otherwise, it forwards the request to the child (lines 13-14).

This process works recursively until the coordinates of all regions that intersect with CK are sent to the driver, along with the identities of their parking managers. The app at the driver determines in which region her destination is located and then communicates directly with the parking manager of that region. In this communication, the app at the driver requests a parking space from the parking manager using its exact destination, not the cloaked region. The parking manager then performs parking assignment, as described in Section 5. In this way, only one parking manager learns the driver's destination. All the other nodes in the tree that processed the original request know only the cloaked region CK.

4.4 Load Balancing

Each parking manager receives requests for its region. However, the distribution of destinations and requests coupled with the tree-structure of the network can cause a heavy load on certain managers. Heavy load leads to slow downs and significant battery consumption on the impacted phones. To avoid this problem, DFPS applies a simple load balancing mechanism. An overload threshold is determined by each parking manager as the difference between the local load (i.e., the number of requests that have been processed by the phone) and a load threshold . If > , an imbalance is detected and the phone removes itself from the system. The threshold can be determined experimentally on each phone such that the phone does not consume more than a small fraction of its battery power serving DFPS requests. As described in Section 4.2, a phone of another parked driver will replace this phone in the overlay network and will handle any pending requests inherited from the removed phone.

4.5 Failure Recovery

DFPS employs the mechanism proposed in [31] to ensure that the system continues to function in the presence of failures or disconnections of the phones of parked drivers. For example, the phones may fail without warning if the drivers decide to turn them off. Failure or disconnection of the phones is detected by periodic gossip messages from their neighbors. Each neighbor knows the region boundary of the failing node w and maintains a replica of the data it stores (e.g., the number of available spaces and total number of spaces) in order to restore the data and improve availability. In addition, a parent maintains a list of requests forwarded to w and requests assigned by w in case of failure. To avoid consistency problems, a disconnected parking manager will not attempt to reconnect to the system when the wireless connection becomes available again. Finally, let us note that the overload threshold at parking managers (used for load balancing as described in Section 4.4) also reduces the effect of node failures or disconnections.

5 Parking Space Assignment

In DFPS, each parking manager periodically runs the same parking space assignment algorithm to satisfy the outstanding requests that have been forwarded to it. The algorithm computes an assignment for these requests, aiming to optimize the total travel time of the drivers.

5.1 Assumptions and Problem Statement

The set of curbside parking spaces in the region is denoted $S = fs_1; s_2; ::::; s_m g$. The set of drivers whose requests have been forwarded to the parking manager because their destinations are within its region are denoted $V = fv_1; v_2; ::::; v_n g$. To the greatest extent possible, each driver will be assigned a parking space in the region that contains her destination. If this is not possible (e.g., all the parking spaces in that region are taken), the driver will receive a parking space in a nearby region.

The destinations of the drivers are geographical locations, such as shops, banks, houses, hotels, etc. Their addresses are converted into latitude and longitude coordinates, similar to the locations of parking spaces. The drivers are assumed to be moving independently based on legal speeds and the congestion levels on different road segments. We also assume that the parking request is submitted to the dispatcher when the driver is about to reach the cloaked region. Then, the request is forwarded to the corresponding parking manager.

An assignment of parking spaces to drivers is defined as $Y: V \; ! \; S$, where y_{ij} is the assignment of a driver v_i 2 V to a parking space s_j 2 S:

$$y_{ij} = \left(\begin{array}{ll} 1; & \text{if } v_i \text{ is assigned to } s_j \\ 0; & \text{otherwise} \end{array} \right. \quad 1 \;\; i \;\; n; 1 \;\; j \;\; m \tag{3}$$

$$\overset{n}{\underset{i=1}{X}} y_{ij} \;\; 1; \quad 1 \;\; j \;\; m \; (i:e:; s_j \; 2 \; S) \tag{4a}$$

$$\overset{m}{\underset{j=1}{X}} y_{ij} = 1; \quad 1 \;\; i \;\; n \; (i:e:; v_i \; 2 \; V) \tag{4b}$$

Constrains 4a and 4b ensure, respectively, that a driver receives at most one space and that a space is not assigned to more than one driver.

For a set of drivers and a set of parking spaces, there may exist a large number of assignments. The algorithm is to find an assignment that can minimize the total travel time of the drivers. The travel time $TC(v_i)$ of a driver v_i is the time period from the moment when the driver submits her request until she arrives at her destination. It consists of two parts, the driving time and the walking time:

T_d (O_{v_i}, s_j) is the driving time of driver v_i from the moment she submits her request from location O_{v_i} until she parks at the parking space s_j.

T_w (s_j, D_{v_i}) is the walking time of the driver from the moment she parks until the moment she arrives at her destination D_{v_i}.

5.2 Parking Space Assignment Algorithm

Each parking manager in DFPS assigns parking spaces in its region in the same way as the dispatcher assigns parking spaces in our centralized solution. The detailed algorithm can be found in [5]. A brief description is included below for convenience.

A parking manager assigns parking spaces periodically to outstanding requests. Each manager can adjust the period as a function of its outstanding request queue size to achieve a good trade-off between assignment performance and overhead. In each period, the manager first pre-allocates to the driver of each outstanding request the closest available parking space to her destination. The pre-allocation adapts the solution to the flow problem described in [32] to minimize the total walking time of these drivers.

The actual assignment of parking spaces takes place based on the urgency of the demands for parking spaces (i.e., how close the drivers are to their destinations). We apply a modified version of the compound laxity algorithm [5] to determine how long a request can be delayed before it must be assigned a space. Specifically, in each period, the drivers with the most urgent demand are selected. Their pre-allocated parking spaces are officially allocated to them.

The algorithm described above is named FPA. It assumes that subscribed drivers are generally representative of the entire driving population and all spaces in the region are available to them. To consider the cases in which spaces may be taken silently by unsubscribed drivers, the algorithm is enhanced to track the spaces taken by unsubscribed drivers and avoid assigning these spaces. This algorithm, described in [5], is named FPA-1.

FPA and FPA-1 are used under the assumption that there are still available parking spaces in the region. However, in DFPS, the assignment of parking spaces is done by multiple parking managers. It is possible that a parking manager runs out of parking spaces, but still has outstanding requests. In such a case, DFPS allows a parking manager to acquire parking spaces from nearby parking managers temporarily to satisfy her outstanding requests. For this purpose, DFPS uses a multi-indexing technique to locate nearby regions and find the best available spaces that are close to the destinations of the parking requests. This process is detailed in [1]. The multi-indexing technique is also leveraged to pass the parking availability information up the tree to the dispatcher, which needs it for phase (d) of the cloaked region construction.

Let us note that, unlike the dispatcher, the parking managers know the real destination of the drivers and the parking assignment is optimized based on these destinations.

6 Privacy Analysis

This section analyzes how the DFPS protocol satisfies the desired privacy goals.

Driver Identity privacy: A driver uses a randomly-generated pseudonym identity with every parking request, which is completely unrelated to her true identity. This pseudonym is not identifiable by attackers at the dispatcher side because (1) the parking request is a snapshot query (i.e, a request submitted just once by the driver); (2) the pseudonymity mechanism is effective due to the discrete characteristic of the parking behavior (i.e., the average time interval between two parking demands is long enough). Thus, the attacker cannot infer the driver identity from a parking request.

Unlinkability of parking requests: Given two parking requests, no one should be able to tell if they are made by the same driver or by different drivers. This is achieved by the use of pseudonyms and cloaked regions. In other words, with each parking request, a driver's privacy is protected by (1) replacing her true identity with a randomly-generated pseudonym; (2) constructing the cloaked regions such that two requests from the same driver to the same destination will result in different cloaked regions.

The security of inferring information about the driver's identity from the pseudonym and the security of linking two requests purely based on pseudonyms can be reduced to the security of the underlying function used to generate the pseudonyms, which are indistinguishable from random strings.

Parking destination privacy: By design, the cloaked area contains k destinations, which ensures the driver's true destination is hidden among these k destinations. However, to infer a driver's destination, the attacker may deploy a "center-of-cloaked-region" privacy attack [28], [29], i.e., the destination is inferred to be at the center of the cloaked region. The attacker may also be physically present at a location to determine a driver's destination.

The parking request in DFPS includes provisions to mitigate these attacks. To alleviate the "center-of-cloaked region" attack, phase (b) of the parking request protocol randomly shifts the center of the cloaked region. Thus, even if the same driver chooses to travel to the same destination on different occasions, the cloaked area (which contains the true destination) will appear differently to the dispatcher. Although the cloaked area contains k destinations after phase (b) of the parking request protocol, it may still be possible that these k destinations be clustered in a small region. If the attacker decides to be physically present in this small region, the driver's true destination may be determined based on direct observation. To mitigate this issue, phase (c) of the parking request protocol uses the entropy of distance method to select a cloaked region which contains k destinations that are located on more than $k/2$ segments and are evenly distributed in the cloaked region. As a result, the probability of inferring the true destination within the cloaked region remains $1=k$.

Due to the fact that the dispatcher manages the entire space before any driver parks, the real destinations of the first right and left managers will be known, which can lead to privacy leakage. However, this has a very minor effect on the privacy of the whole system.

In extreme cases, for example when the dispatcher receives only one request during a long period of time, the dispatcher could learn the parking space of the driver based on the availability statistics it receives from the overlay network. However, the dispatcher will not learn the actual destination of the driver since each parking request is served and assigned only by a parking manager who manages the region that the driver's destination is located in, which is our main goal in terms of privacy. Although DFPS tries to assign an available parking space in close proximity of the destination, there may not be available spaces near the destination, or the destination merely has no parking spaces to begin with. Therefore, it is not easy for the dispatcher to link the parking space to the querying driver. Furthermore, the pseudonyms and anonymizers will protect the driver's identity, so the dispatcher will not be able to link this parking space to other parking spaces of the same driver.

7 Experimental Evaluation

In this section, we experimentally evaluate the performance of DFPS, in terms of (1) scalability and load balancing, (2) assessing the benefits of DFPS with and without privacy compared to the centralized system;(3) measuring the benefits of DFPS in terms of travel time when compared to a Naive parking assignment solution, which resembles what drivers normally do; (4) investigating the DFPS performance under different privacy protection mechanisms; (5) understanding the relationship between the average travel time and the size of the cloaked regions; (6) analysing the impact of the privacy level on the average travel time.

The evaluation is done via simulations over a real road network. The experiments simulate two different scenarios: subscribed-drivers-only scenario, which assumes that all drivers in the system use DFPS; unsubscribed-drivers-interference scenario, which assumes there are a number of drivers who have not subscribed to DFPS and who may occupy, without notification, parking spaces known to the system as available.

We use SUMO/TraaS [16], an open source framework for running vehicular network simulations, and PeerSim [17], a simulation environment for P2P protocols. SUMO/TraaS simulates vehicles going to their destinations in a business district in Manhattan, New York City. PeerSim simulates the overlay peer-to-peer network of parked drivers. Figure 5(a) shows the road network used in the simulations, while Figure 5(a) illustrates an example of destinations and parking spaces along a few road segments. The total number of parking spaces is 1024, and the total number of destinations is 400.

We generate a set of trips for the drivers. While the starting locations of the vehicles are randomly chosen over the entire road network, the target destinations are chosen randomly from a small region in the center of the map to ensure enough contention for parking spaces. The travel time of a driver is the sum of the driving time and the walking time. In these experiments, the driving time is the sum of the time the driver takes to reach the cloaked region and the time from the edge of the cloaked region to the assigned parking space. The walking time is calculated from the assigned parking space to the destination. We consider a walking speed of 1.4 m/s [33]. DFPS starts each test with 1024 vacant parking spaces. The arrival rate of the requests falls within the range of 1 to 5 requests per second. For each experiment, we collect results from 5 runs and average them.

We compare the performance of DFPS with (1) our centralized system FPS [5] and (2) a version of DFPS without privacy protection (DFPS-wop) in terms of the average travel time. Unlike DFPS, which uses the edge of the cloaked region to trigger a parking assignment request, FPS and DFPS-wop trigger the request when the driver reaches a circle centered at the destination and with a radius (request

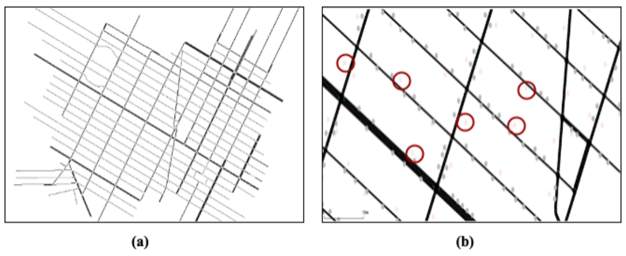


Fig. 5: Road network used in experiments (a). Example of zoomed-in road segments (b): parking spaces (gray dots) and destinations (red circles).

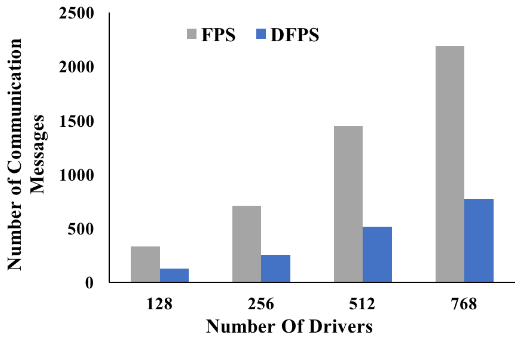


Fig. 6: Number of messages handled by the dispatcher/server in DFPS and FPS for a different number of drivers and a fixed number of destinations (8).

distance) set based on the parking availability in that region. The radius is initially set to the average length of the road segments within the whole region, and it is adjusted periodically based on the parking occupancy rate in the region: the radius is increased when the occupancy becomes higher. Each driver in DFPS selects the value of her k anonymity randomly from the set {3,5,7,9}.

7.1 Results and Analysis

Scalability and Load Balancing. Compared to our centralized system FPS, the computation bottleneck at the central server (i.e., dispatcher) in DFPS is removed, as the parking assignment is computed in a distributed fashion by the phones of the drivers. Therefore, DFPS scales better from a computation point of view.

The total computation time consumed in each period for the parking assignment algorithm is the sum of (1) finding a valid allocation that minimizes the total walking time to destinations and (2) determining the urgency of pending requests in order to assign spaces to the most urgent requests and minimizing the total driving time.

Minimum-cost network flow in a directed bipartite graph is used to generate a valid allocation. Its cost is $O(ve)$, where v is a number of nodes (i.e., m spaces + n drivers in the region) and e is the number of edges (i.e., equal to n, the number of drivers in the region). The cost of computation to determine request urgency and select urgent requests is $O((n + ')^2)$, where n is the number of drivers to be assigned parking spaces and $'$ is the number of drivers with high urgency. Thus, the total time for each parking manager is $O(n^2 + nm)$+$O((n + ')^2)$. Given that

TABLE 1: Data Exchanged in FPS, DFPS-wop and DFPS Systems with 768 Drivers, and 2 Destinations

	FPS	DFPS-wop	DFPS(k=3)	DFPS(k=5)	DFPS(k=7)	DFPS(k=9)
Data Exchanged (KB)	34	27	62	70	78	83

each parking manager handles only a limited number of parking spaces and drivers, as described in Section 4.4, this computation can easily be done on today's smart phones.

To generate a privacy-aware parking request, each driver needs to compute a cloaked region, and the dispatcher may need to adjust it. However, computing a cloaked region is a one-time cost and is fully distributed. Adjustment is required only by a small proportion of requests when parking areas become full. Due to the privacy-preserving request forwarding, some nodes will have to handle more messages than in the case without privacy. However, the computation cost for these operations is negligible compared to the computation cost of the parking assignment algorithm.

Figure 6 compares the number of messages handled by the dispatcher in DFPS and the server in FPS by varying the number of drivers from 128 to 768. The results show that DFPS reduces the number of messages by a factor of 2 or better when compared to FPS.

To handle privacy-aware parking requests, DFPS needs to contact all parking managers whose regions intersect the cloaked regions in the requests, incurring higher communication cost than contacting only one parking manager if privacy is not considered. Furthermore, the size of the messages is slightly larger in DFPS than in FPS, due to the additional protocol information to preserve privacy. To evaluate this cost, we measure the total amount of data exchanged in the system for FPS, DFPS (with privacy), and DFPS-wop (without privacy). The results in Table 1 show that the amount of data exchanged by each system in a scenario with 768 drivers and 2 destinations is in the order of tens of KB. We also notice that the amount of data for DFPS with k=3 is about double of that for DFPS-wop. Then, the amount of data for DFPS increases linearly with k. This demonstrates that there is a communication cost associated with privacy, but the system scales well.

Next, we investigate the effect of load balancing on parking managers. We compare the number of requests assigned by the phone of a parked driver when DFPS employs its load balancing mechanism (DFPS) and when it does not (DFPS*). The value of the load threshold in the load balancing mechanism is set to 10. Table 2 compares the maximum number of assigned requests in DFPS and DFPS*. As expected, DFPS scales better due to its load balancing mechanism. The maximum number of requests in DFPS* is about 20 times higher than the maximum number in DFPS. We also observe that the maximum in DFPS is 13, not 10 as expected (the load threshold). This is because of two reasons. First, a parking manager receives requests until it has served of them (while the others are pending). Second, a parking manager can not depart the network until it completes its set of requests with high urgency.

Figure 7 presents another measure of scalability: the average number of parking managers cooperating to serve the incoming parking requests in DFPS and DFPS*. The results show that the number of managers in DFPS increases by as much as 185% compared to DFPS*. Higher numbers

TABLE 2: Maximum Number of Requests Assigned by a Parked Driver for Different Numbers of Destinations and 768 Drivers

	Destinations		
	2	4	8
DFPS	13	12	12
DFPS*	268	166	100

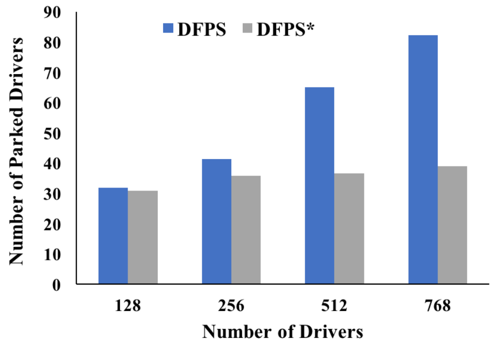


Fig. 7: Number of parked drivers involved in the assignment process for different total numbers of drivers and 8 destinations.

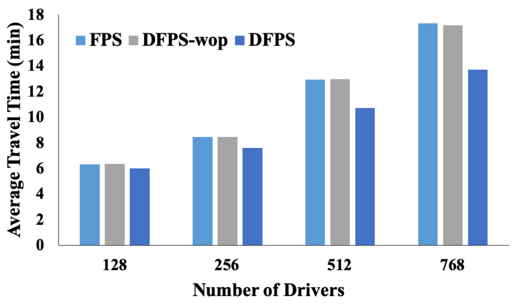


Fig. 8: Average travel time for a different number of drivers and 8 destinations.

are better because the load is distributed more evenly across the participants, and the system scales better.

Average travel time. The average travel time measures the performance of the system from a global point of view. Figure 8 shows the average travel time for DFPS compared to FPS and DFPS-wop in the subscribed-drivers-only scenario. In FPS and DFPS-wop, destination privacy is not considered. Their regions are constructed based only on parking availability. The results show that DFPS reduces the travel time by as much as 26% compared to DFPS-wop and FPS. We observe that a DFPS-wop and FPS have similar results.

The improvement in performance observed when comparing DFPS with DFPS-wop and FPS is due to combining the privacy and parking availability requirements when generating the cloaked region. Let us recall that DFPS-wop and FPS use just parking availability to generate their regions. The privacy requirement allows DFPS to optimize the size of the region better than DFPS-wop and FPS. This is due to the even distribution of the destinations, which also distributes better the available parking spaces. An optimized region allows for more effective parking assignment

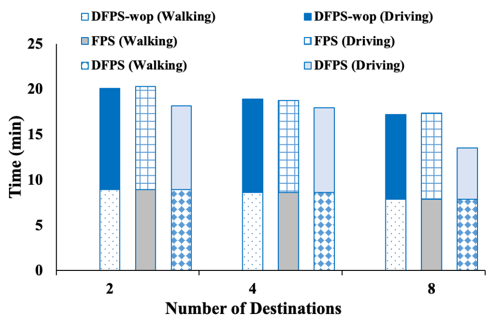


Fig. 9: Walking time and driving time for different number of destinations and 768 drivers.

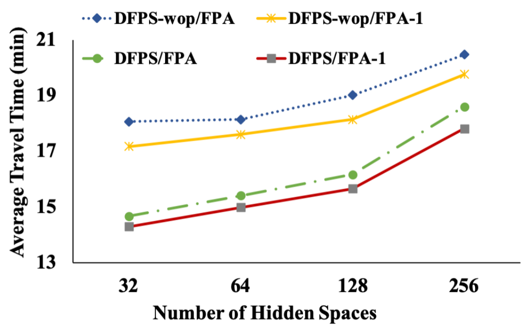


Fig. 10: Average travel time for different numbers of hidden spaces, 768 drivers, and 8 destinations.

optimizations.

For the same scenario, Figure 9 shows the average travel time when the number of destinations is varied from 2 to 8. The figure breaks down the travel time into two parts: driving time and walking time and shows that drivers spend most time on driving, and DFPS reduces the average travel times by mostly reducing the driving time. With 8 destinations, DFPS can reduce driving time by 67% and 64% compared to FPS and DFPS-wop, respectively. Reducing the driving time is very important, as this reduces traffic congestion and implicitly the gas cost and pollution. Since the number of parking spaces in the centroid area is limited, all systems can hardly reduce the walking time.

The next set of experiments evaluate the performance of DFPS and DFPS-wop in the unsubscribed-drivers-interference scenario. Each system, has two parking assignment algorithms. One (DFPS/FPA and DFPS-wop/FPA) just keeps trying to find another space if the assigned parking space is found to be taken by an unsubscribed driver. The other (DFPS/FPA-1 and DFPS-wop/FPA-1) keeps track of the spaces found to be taken by unsubscribed drivers, avoids them for a while, and tries them later. We call the spaces taken by unsubscribed drivers "hidden" spaces. These spaces are taken at the beginning of the simulation to help tracking them. More details on how we model the behavior of unsubscribed drivers can be found in [5]. Figure 10 shows that DFPS continues to perform better than DFPS-wop, even in the presence of unsubscribed drivers. We also notice that FPA-1 improves the performance for both systems, and DFPS/FPA-1 achieves the lowest average travel time. These results demonstrate that DFPS/FPA-1 adapts well to the interference caused by unsubscribed drivers.

Individual Travel Time Gains/Losses. In the subscribed-drivers-only scenario, we conduct an experiment to find out the gains and losses in travel time for individual drivers when comparing DFPS with a Naive solution, a baseline assignment algorithm that assumes the driver goes to the destination and, after arriving there, she starts a breadth-first-search for parking spaces along the nearby road segments. The Naive solution is similar to what most people do in real life. To measure the gains/losses, we calculate the ratio between the travel time obtained by the Naive solution and the travel time obtained by DFPS for each driver. If the ratio is higher than 1, the driver has benefited from DFPS. Otherwise, the driver has not.

Figure 11 plots the distribution of individual travel time

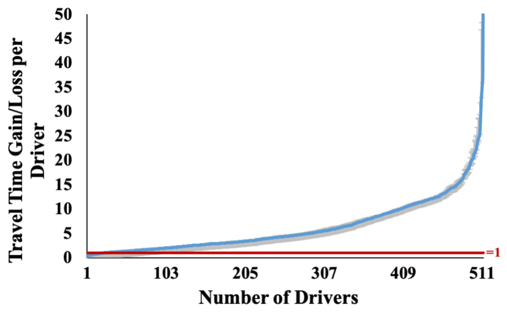


Fig. 11: Distribution of travel time gain/loss for 512 drivers and 8 destinations. Error bars are shown.

gains/losses for all drivers in the experiment. We observe that DFPS manages to improve the travel time for a large majority of drivers (over 95%). Many drivers reduce their travel times by more than an order of magnitude. These results are possible due to the high parking contention generated in the experiment, which leads to high traffic congestion and thus to very high travel times for the Naive solution. The small error bars in the figure demonstrate that these results are consistent across different simulation runs.

Entropy-based cloaking. To determine how our entropy-based cloaking technique affects travel times, we compare its performance with that of a simple k-anonymity technique, which creates cloaking areas containing the k   1 nearest neighbor destinations around the real destination. Figure 12 shows the average travel time for when DFPS works with either of these two methods in three cases: DFPS with subscribed-drivers-only, DFPS/FPA with unsubscribed-drivers-interference, and DFPS/FPA1 with unsubscribed-drivers-interference. The results show that DFPS with the entropy-based cloaking technique achieves better performance consistently for all three cases. Therefore, we conclude that the entropy-based cloaking improves both the destination privacy and the travel time. This is because its cloaked region is larger, with destinations spaced more evenly, and thus avoids parking contention and traffic congestion.

Effect of region size. To understand how the region size affects performance, we compare the average travel times for DFPS and two versions of DFPS-wop. The minimum number of available parking spaces in a region, P_{min}, is set to 3 for DFPS and one version of DFPS-wop, denoted DFPS-

TABLE 3: Average Travel Time for Different Region Sizes, Different Numbers of Destinations, and 768 Drivers

Destinations	2			4			8		
	DFPS	DFPS-wop(3)	DFPS-wop(6)	DFPS	DFPS-wop(3)	DFPS-wop(6)	DFPS	DFPS-wop(3)	DFPS-wop(6)
Region Size	319.6	356.2	405.8	309.3	366.8	403.6	684.2	371.5	394
Avg. Travel Time	18.1	17.8	16.5	17.9	16.9	16	13.5	15.8	15

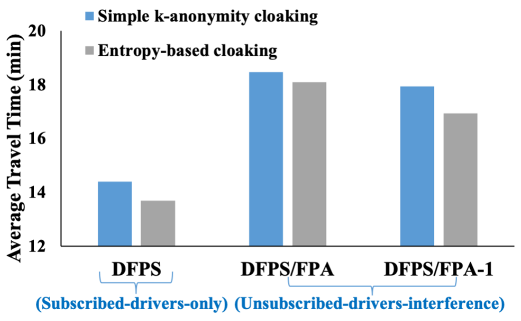


Fig. 12: Average travel time for simple k-anonymity cloaking vs. entropy-based cloaking for 768 drivers, 8 destinations, and 265 hidden spaces.

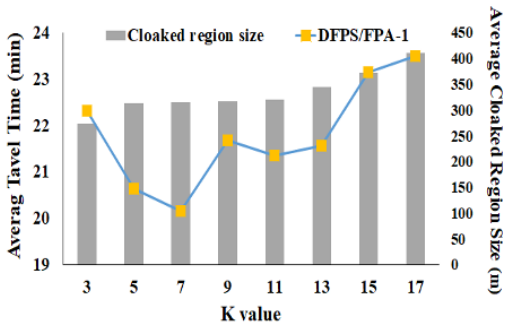


Fig. 13: Average travel time and cloaked region size for different values of k(-anonymity), 768 drivers, 2 destinations, and 265 hidden spaces.

wop(3), and to 6 for the other version of DFPS-wop, denoted DFPS-wop(6).

Table 3 shows that the average travel time gradually decrease with larger region sizes. The cloaked regions of DFPS with two and four destinations are smaller than the regions in DFPS-wop, due to two reasons: (1) There are many neighbour destinations around the 2 or 4 destinations chosen in the experiments; this helps reducing cloaked region sizes in DFPS. (2) The parking availability around the destinations is high when drivers submit their parking requests. However, we noticed that the region with eight destinations is larger in DFPS than the regions in DFPS-wop. The reason is that the distance between the 4 new destinations (in addition to the first 4) and their nearest neighbours are relatively large (i.e., sparsely populated region). Thus, to construct a region that satisfies DFPS privacy requirement, the region has to be expanded.

The results show that a slight increase in the region size can significantly improve the travel time in DFPS. This indicates that using larger k values is a good solution: it expected to increase the privacy protection and improve the travel time, at the same time. However, if the regions become too large, it is possible that the parking assignment is done too early and unsubscribed drivers may take some of the assigned spaces. Next, we investigate this trade-off between privacy protection as measured by the value of k and the average travel time.

Impact of increasing the privacy level on average travel time. Figure 13 shows how the average travel time and the region size vary with k in an unsubscribed-driver interference scenario. A number of 256 hidden spaces are taken by unsubscribed drivers gradually at a rate of two spaces every minute. We observe that increasing k leads to larger cloaked regions, which provide better privacy protection. The travel time, however, is not proportional with the region size. It gradually reduces until k = 7, when it achieves the best value, and then it increases. The slight decrease in the travel time for k = 11 vs. k = 9 can be explained by the fact that parking spaces are taken unevenly by unsubscribed drivers. For example, for k = 11, the unsubscribed drivers tend to occupy spaces farther away from the real destinations. The substantial increase from k=13 to k=17 is due to the large size of the cloaked region, which allows the close parking spaces to be taken by unsubscribed drivers. This forces the DFPS drivers to go to the assigned parking space, find that it is occupied, and then search for a new parking space. Overall, the results demonstrate that a good balance can be found between the level of privacy protection and the travel time (e.g., k = 7 in this experiment).

8 Related Work

This section discusses the parking problem from different perspectives: parking assignment algorithms, privacy-preserving parking systems, privacy-preserving VANET protocols, location privacy techniques, and parking occupancy detection.

8.1 Parking Assignment Algorithms

Parking assignment is studied in two contexts: centralized and distributed. The works in [5], [34] are examples of centralized solutions. Ayala et al. [34] developed a pricing model to minimize the system-wide driving distance. However, the proposed approach depends on pricing data and is offline in nature, as the number of drivers and resources are known in advance and do not dynamically change. In [5], we introduced FPS, a parking assignment system that dynamically adapts to new parking requests and shows good performance in reducing the total travel time for drivers. However, it has a scalability problem, inherent to a centralized solution.

Among the decentralized solutions, Ayala et al. [32] analyzes the parking problem as a competitive game in which individual, selfish drivers are competing for the available spaces. It is assumed that each vehicle has access to the location of the other vehicles, which raises privacy concerns

TABLE 4: Comparison Between DFPS and Other Privacy-Preserving Parking Solutions

	[36]	[37]	[14]	[13]	[6]	[7]	[38]	DFPS
Free Curbside Parking					X			X
Destination Privacy				X				X
Identity Privacy	X	X		X	X	X	X	X
Location/Parking Privacy		X	X	X	X	X		X
VANET-Based					X		X	
Distributed System	X	X					X	X
Parking Assignment	X		X	X		X		X
Parking Navigation					X		X	
Large Scale Coverage		X			X			X

and has technical difficulties in real-time. In [35], the parking assignment capitalizes on the ability of a trustworthy central controller to protect driver's destination and to construct a feasible assignment in a distributed fashion via the coordination of drivers. The problem with this solution is that the assignment computation and communication are burdens on the coordinator. The preliminary version of this article [1] presented a distributed parking assignment system that solves the scalability problem in the centralized FPS system, while maintaining the same level of travel time performance. Yet, it did not take driver's privacy into consideration. In this article, DFPS takes full advantage of drivers' smart phones to perform privacy-preserving and scalable parking assignment, while optimizing the total travel time for drivers.

8.2 Privacy-Preserving Parking

Several privacy-preserving parking techniques have been recently proposed. Table 4 illustrates the differences between DFPS and all these techniques. The table also demonstrates the benefits of DFPS when compared to these techniques. In the following, we briefly discuss each of these techniques in relation to DFPS.

PEPS [36] is a distributed scheme using blockchain technology to enable agents to lease their private parking spaces and make a profit. PEPS [36] targets private parking spaces only and does not protect the privacy of drivers' destinations. Differently, DFPS targets public free parking spaces and protects the privacy of drivers' destinations. Amiri et al. [37] propose a decentralized and privacy-preserving smart parking system using consortium blockchain and private information retrieval. A consortium blockchain is created by different parking lot owners to ensure security, transparency, and availability of the parking offers. This work preserves the drivers' privacy, but it works only for parking lots. DFPS can protect driver privacy, while assigning free curbside parking spaces close to the drivers' destinations.

ASAP [14] is an anonymous smart-parking and payment scheme in vehicular networks. It uses short randomizable signatures to provide anonymity and conditional privacy. ASAP handles private parking spots through a trusted server, which may suffer from data breaches. Furthermore, it is designed to achieve pseudonymity, but not unlinkability. DFPS, on the other hand, works as a scalable distributed system that prevents linking multiple parking assignments of the same driver. In addition, DFPS does not maintain driver data after a request has been served, thus preventing subsequent data breaches. Furthermore, DFPS provides a free parking solution, while ASAP involves payments. Huang et al. [13] propose a centralized privacy-preserving parking reservation scheme for securing the automated valet parking system AVP. The scheme preserves the privacy of drivers' real identities using anonymity. It also uses location obfuscation techniques (e.g., geo-indistinguishability and cloaking) to protect the drivers' desired destinations. Unlike this scheme, DFPS is decentralized and does not store user parking information in the system database, which has the risk of privacy breach and data loss. Further, the location obfuscation techniques of this scheme may assign parking spaces to drivers far away from their destinations. On the other hand, as shown in our evaluation, DFPS protects destination privacy while providing parking spots close to the destinations.

P-SPAN [6] is a smart parking navigation system, which uses a cloud server and roadside units (RSUs) to guide users to available parking lots at their destinations. P-SPAN hides drivers' identities using anonymous credentials. However, the cloud server can still identify the drivers from their parking locations or by linking their multiple parking locations. Furthermore, other sensitive information about drivers is released to the cloud server, e.g., current locations, destinations, and arrival times. This enables the cloud server to track drivers more easily. DFPS is designed to hide drivers' destinations and prevent linking their multiple parking locations by constructing a new cloaked region with each parking request for the same destination. PrivAV [7]is a secure and privacy-preserving automated valet parking scheme for self-driving vehicles. PrivAV extends anonymous authentication to support two-factor authentication to reduce the risks of vehicle theft while protecting the privacy. While location privacy is protected by the parking assistant, the driver's identity and destination may still be exposed. Unlike PrivAV, DFPS protects both the driver's identity and destination. Lu et al. [38] consider VANET with a privacy-preserving parking scheme for large parking lots, which enables RSUs to locate vacant parking spaces for arriving vehicles. Differently, DFPS covers free curbside parking spaces across entire cities and drivers do not depend on VANETs, which may have intermittent connectivity.

8.3 Privacy-Preserving VANET Protocols

To support safety or infotainment applications without exposing drivers' privacy, a variety of privacy-preserving vehicular communication protocols have been proposed in VANET. We discuss these protocols next, but we do not include them in Table 4 because they do not target parking assignment as DFPS. Chim et al. [39] proposed VSPN, which can guide vehicles to desired destinations in a distributed manner while preserving drivers' privacy. Identity privacy is preserved using pseudo-identities and anonymous credentials. Unfortunately, this protocol is vulnerable to internal attacks from vehicles as the master key is shared among all vehicles. Cho et al. [40] introduced a security-enhanced navigation system based on the concept of two-person multi-signature and identity-based cryptographic schemes. However, this work does not discuss how to collect traffic information, which is a difficult problem. Sur et al. [41] proposed a secure navigation system based on vehicular cloud, a trapdoor hash function, and zero-knowledge proofs. One problem with this solution is that the anonymous credentials can only be used once in order to avoid sharing of vehicles' credentials with unregistered users.

8.4 Location Privacy Approaches

Location obfuscation [11], space transformation [42], and cloaking [20], [27] are traditional approaches to preserving location and/or query privacy. The former means that given a query, attackers cannot learn the issuer's exact position; the latter enforces the unlinkability between the issuer and the query. Cloaking is more widely used than other techniques because of its efficiency. The main idea is to blur user locations into spatial cloaked regions that satisfy certain privacy requirements (e.g., k anonymity). Different from existing solutions, cloaking is utilized in a unique way in DFPS. First, existing solutions use it to break the linkability between users and their location and/or queries, while our solution aims to protect drivers' destinations. Second, existing solutions rely on a third party agent to perform cloaking in a centralized way or a peer-to-peer infrastructure for mobile users to perform cloaking collaboratively. DFPS does not assume these architectures, since it uses the smart phones of the drivers to compute the cloaked regions.

Location privacy preserving mechanisms (LPPMs) have been extensively studied. Cao et al. [43] investigated a new type of LPPM's privacy goal: protecting spatiotemporal events in continuous location-based services, e.g., hospital visits. Spatiotemporal event privacy guarantees flexible and customizable protection; however, it focuses only on protecting drivers' real-time locations. DFPS is designed to protect drivers' destinations and to prevent linking of multiple parking requests from the same driver.

Anonymity algorithms are proposed to form spatial cloaked regions. In [44], Abul et al. proposed a quad-tree-based anonymity algorithm which adopts a recursive method to continuously divide the space region in which the mobile user resides into four quadrants. In [24], Mokbel et al. proposed an anonymous algorithm based on the Casper model, which effectively improves the performance of the anonymity algorithm in [20]. However, there are some problems with these algorithms. In [44], the anonymity algorithms may form redundant regions in the process of constructing an anonymous region. In [24], the distribution of users is not considered and due to the lack of users in sparsely populated regions, the anonymous region will fail to be constructed. Our proposed privacy technique works well for both sparse and dense regions. It considers the distance between the real destination and its neighbouring destinations to construct a cloaked region that satisfies k anonymity and ensures that the destinations in the cloaked region are not clustered together.

8.5 Parking Occupancy Detection

Monitoring and sensing parking spaces approaches have been presented in order to track free parking spaces at any time. For instance, SFpark [2] installs sensors on streets to detect and provide real-time parking availability. ParkNet [3] uses ultrasonic sensor technology on a vehicle door to collect parking availability information and then build a real-time map for drivers. Although these solutions increase the probability of finding vacant spaces on streets, they have several shortcomings. First, the cost involved in deploying and maintaining the sensor infrastructure is high. Second, all drivers see the same parking availability map at any given time, and many of them will compete for the same spaces. This parking contention problem leads to traffic congestion and frustrating driving/parking experience.

An alternative proposal to the process of finding parking spaces is relying on estimated/predicted information. Authors in [45] used wireless ad hoc networking to estimate the probability of successful parking within a certain distance from the current location. Authors in [4] proposed an online parking guidance system that recommends parking spaces in real-time based on parking availability prediction. Parking payment terminals are also used in [46] to disseminate information about available parking spaces. While this type of work may be useful for guiding drivers to a region where they have a high likelihood to find a free parking space, it can also lead to parking contention and traffic congestion.

DFPS differs from the above research in three aspects. First, DFPS does not rely on expensive infrastructure, but on cooperative mobile phones, which is a cheaper, more convenient, and more flexible alternative. Second, DFPS allocates parking spaces to drivers to reduce parking contention, which improves the parking effectiveness when multiple drivers are looking for parking in the same region. Third, to the best of our knowledge, DFPS is the first such system to protect the privacy of drivers' destinations without help from a trusted third-party.

9 Conclusion

This article presented DFPS, a cost-effective and efficient distributed mobile system for parking assignment that can be implemented and deployed in real-life settings. DFPS uses the smart phones of the drivers to offload the computation of parking request assignments from a central server, and thus the assignment process becomes scalable in real-time. Parked drivers cooperate to serve parking requests in a distributed fashion while optimizing the social welfare of the whole system, i.e., minimizing the total travel time. DFPS protects the privacy of the drivers' destinations through a novel entropy-based cloaking technique, which guarantees k anonymity. The simulation results demonstrated that DFPS is scalable, effectively reduces the average travel time, and achieves better performance than a centralized system. Furthermore, the results show that achieving destination privacy does not hurt the travel time performance.

Acknowledgment

This research was supported by the National Science Foundation (NSF) under Grants No. DGE 1565478, SHF 1617749, and CNS 1801430. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

[1] A. Hakeem, N. Gehani, R. Curtmola, X. Ding, and C. Borcea, "Cooperative System for Free Parking Assignment," in Proceedings of the IEEE Vehicular Networking Conference (VNC), 2017, pp. 319–326.

[2] "SFpark," https://sfpark.org, San Francisco, CA, USA, [Online; accessed 02-April-2020].

[3] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "ParkNet: Drive-by Sensing of Road-side Parking Statistics," in Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys), 2010, pp. 123–136.

[4] K. S. Liu, J. Gao, X. Wu, and S. Lin, "On-Street Parking Guidance with Real-Time Sensing Data for Smart Cities," in Proceedings of the 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON), June 2018, pp. 1–9.

[5] A. Hakeem, N. Gehani, X. Ding, R. Curtmola, and C. Borcea, "On-The-Fly Curbside Parking Assignment," in Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services, ser. MobiCASE'16, 2016, pp. 1–10.

[6] J. Ni, K. Zhang, Y. Yu, X. Lin, and X. Shen, "Privacy-Preserving Smart Parking Navigation Supporting Efficient Driving Guidance Retrieval," IEEE Transactions on Vehicular Technology, vol. 67, no. 7, pp. 6504–6517, July 2018.

[7] J. Ni, X. Lin, and X. Shen, "Toward Privacy-Preserving Valet Parking in Autonomous Driving Era," IEEE Transactions on Vehicular Technology, vol. 68, no. 3, pp. 2893–2905, 2019.

[8] E. Nardelli, "Distributed K-D Trees," in Proceedings 16th Conference of Chilean Computer Science Society (SCCC'96), 1996, pp. 142–154.

[9] B.Gedik and L. Liu, "Location Privacy in Mobile Systems: A Personalized Anonymization Model," in Proceddings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05), June 2005, pp. 620–629.

[10] F. Dürr, P. Skvortsov, and K. Rothermel, "Position Sharing for Location Privacy in Non-Trusted Systems," in Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom), March 2011, pp. 189–196.

[11] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Protecting Location Privacy: Optimal Strategy Against Localization Attacks," in Proceedings of the ACM Conference on Computer and Communications Security. Association for Computing Machinery, 2012, p. 617–627.

[12] H. Kido, Y. Yanagisawa, and T. Satoh, "An Anonymous Communication Technique using Dummies for Location-based Services," in Proceedings of the ICPS '05. International Conference on Pervasive Services, 2005., 2005, pp. 88–97.

[13] C. Huang, R. Lu, X. Lin, and X. Shen, "Secure Automated Valet Parking: A Privacy-Preserving Reservation Scheme for Autonomous Vehicles," IEEE Transactions on Vehicular Technology, vol. 67, no. 11, pp. 11 169–11 180, Nov 2018.

[14] L. Zhu and M. Li and Z. Zhang and Z. Qin, "Asap: An anonymous smart-parking and payment scheme in vehicular networks," IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 4, pp. 703–715, 2020.

[15] L. Ni, F. Tian, Q. Ni, Y. Yan, and J. Zhang, "An anonymous Entropy-based Location Privacy Protection Scheme in Mobile Social Networks," EURASIP Journal on Wireless Communications and Networking, vol. 2019, no. 1, p. 93, Apr 2019.

[16] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO - Simulation of Urban MObility: An overview," in Proceedings of the Third International Conference on Advances in System Simulation (SIMUL), 2011, pp. 63–68.

[17] A. Montresor and M. Jelasity, "PeerSim: A Scalable P2P Simulator," in Proceedings of the IEEE Ninth International Conference on Peer-to-Peer Computing, 2009, pp. 99–100.

[18] "Blockchain Smartphones - Going Mobile," https://innovationatwork.ieee.org/blockchain-smartphones-going-mobile/.

[19] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, 2004, p. 21.

[20] M. Gruteser and B. Hoh, "On the Anonymity of Periodic Location Samples," in Proceedings of the Security in Pervasive Computing, 2005, pp. 179–192.

[21] T. Xu and Y. Cai, "Location Anonymity in Continuous Location-Based Services," in Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems, 2007.

[22] X. Chen and J. Pang, "Measuring Query Privacy in Location-based Services," in Proceedings of the Second ACM Conference on Data and Application Security and Privacy, 2012, pp. 49–60.

[23] C.-Y. Chow and M. F. Mokbel, "Enabling Private Continuous Queries for Revealed User Locations," in Proceedings of the 10th International Conference on Advances in Spatial and Temporal Databases, 2007, pp. 258–273.

[24] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The New Casper: Query Processing for Location Services Without Compromising Privacy," in Proceedings of the 32Nd International Conference on Very Large Data Bases, 2006, pp. 763–774.

[25] L. Sweeney, "K-anonymity: A Model for Protecting Privacy," Int. J. Uncertain. Fuzziness Knowl.-Based Syst., pp. 557–570, 2002.

[26] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing Location-based Identity Inference in Anonymous Spatial Queries," IEEE transactions on knowledge and data engineering, vol. 19, no. 12, pp. 1719–1733, 2007.

[27] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A Peer-to-peer Spatial Cloaking Algorithm for Anonymous Location-based Service," in Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems, 2006, pp. 171–178.

[28] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "MobiHide: A Mobilea Peer-to-Peer System for Anonymous Location-Based Queries," in Proceedings of the Advances in Spatial and Temporal Databases. Springer Berlin Heidelberg, 2007, pp. 221–238.

[29] C. Zhang and Y. Huang, "Cloaking Locations for Anonymous Location Based Services: A Hybrid Approach," GeoInformatica, vol. 13, no. 2, pp. 159–182, Jun 2009. [Online]. Available: https://doi.org/10.1007/s10707-008-0047-2

[30] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in Proceedings of the ACM SIGSAC Conference on Computer Communications Security, 2013, p. 901–914.

[31] G. Tsatsanifos, D. Sacharidis, and T. Sellis, "MIDAS: Multi-Attribute Indexing for Distributed Architecture Systems," in Advances in Spatial and Temporal Databases. Springer, 2011, pp. 168–185.

[32] D. Ayala, O. Wolfson, B. Xu, B. Dasgupta, and J. Lin, "Parking Slot Assignment Games," in Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2011, pp. 299–308.

[33] R. W. Bohannon, A. W. Andrews, and M. W. Thomas, "Walking Speed: Reference Values and Correlates for Older Adults," Journal of Orthopaedic & Sports Physical Therapy, vol. 24, no. 2, pp. 86–90, 1996.

[34] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin, "Pricing of Parking for Congestion Reduction," in Proceedings of the 20th International Conference on Advances in Geographic Information Systems, 2012, pp. 43–51.

[35] E. Alfonsetti, P. C. Weeraddana, and C. Fischione, "A Semi Distributed Approach for Min-Max Fair Car-Parking Slot Assignment Problem," arXiv preprint arXiv:1401.6210, 2014.

[36] L. Wang, X. Lin, E. Zima, and C. Ma, "Towards Airbnb-Like Privacy-Enhanced Private Parking Spot Sharing Based on Blockchain," IEEE Transactions on Vehicular Technology, vol. 69, no. 3, pp. 2411–2423, 2020.

[37] W. A. Amiri, M. Baza, K. Banawan, M. Mahmoud, W. Alasmary, and K. Akkaya, "Privacy-Preserving Smart Parking System Using Blockchain and Private Information Retrieval," in Proceedings of the International Conference on Smart Applications, Communications and Networking (SmartNets), 2019, pp. 1–6.

[38] R. Lu, X. Lin, H. Zhu, and X. Shen, "An Intelligent Secure and Privacy-Preserving Parking Scheme Through Vehicular Communications," IEEE Transactions on Vehicular Technology, vol. 59, no. 6, pp. 2772–2785, July 2010.

[39] T. W. Chim, S. M. Yiu, L. C. K. Hui, and V. O.K. Li, "VSPN: VANET-Based Secure and Privacy-Preserving Navigation," IEEE Transactions on Computers, vol. 63, no. 2, pp. 510–524, 2014.

[40] W. Cho, Y. Park, C. Sur, and K. Rhee, "An Improved Privacy-Preserving Navigation Protocol in {VANET}s," J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl., vol. 4, pp. 80–92, 2013.

[41] C. Sur, Y. Park, and K. H. Rhee, "An Efficient and Secure Navigation Protocol Based on Vehicular Cloud," Int. J. Comput. Math., vol. 93, no. 2, p. 325–344, Feb. 2016. [Online]. Available: https://doi.org/10.1080/00207160.2014.934685

[42] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private Queries in Location Based Services: Anonymizers Are Not Necessary," in Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '08. ACM, 2008, pp. 121–132.

[43] Y. Cao, Y. Xiao, L. Xiong, L. Bai, and M. Yoshikawa, "Protecting Spatiotemporal Event Privacy in Continuous Location-Based Services," IEEE Transactions on Knowledge and Data Engineering, pp. 1–1, 2019.

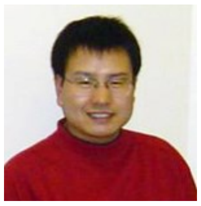
[44] O. Abul, F. Bonchi, and M. Nanni, "Never Walk Alone: Uncertainty for Anonymity in Moving Objects Databases," in Proceedings of the 24th IEEE International Conference on Data Engineering, April 2008, pp. 376–385.

[45] V. Verroios, V. Efstathiou, and A. Delis, "Reaching Available Public Parking Spaces in Urban Environments Using Ad Hoc Networking," in Proceedings of the IEEE 12th International Conference on Mobile Data Management, 2011, pp. 141–151.

[46] M. Caliskan, D. Graupner, and M. Mauve, "Decentralized Discovery of Free Parking Places," in Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks, 2006, pp. 30–39.

Abeer Hakeem is an Assistant Professor in the department of Information Technology at King Abdulaziz University (KAU), Saudi Arabia. She received a P.h.D degree in Computer Science from New Jersey Institute of Technology (NJIT). Her research interests include wireless networking, vehicular networking, mobile computing, location based services, and distributed systems.

Reza Curtmola is a Professor of Computer Science at NJIT. He received a Ph.D. degree in Computer Science from The Johns Hopkins University. His research focuses on the security of cloud services, security of the software supply chain, and applied cryptography. He is the recipient of the NSF CAREER award and has participated in several other projects funded by NSF and DARPA. Dr. Curtmola has published over 65 papers under the umbrella of cybersecurity.



Xiaoning Ding is an Associate Professor in the Department of Computer Science at NJIT. His interests are in the area of experimental computer systems, such as distributed systems, virtualization, operating systems, and storage systems. He earned his Ph.D. degree in computer science and engineering from the Ohio State University.

Cristian Borcea is a Professor in the Department of Computer Science at NJIT. He is also a Visiting Professor at the National Institute of Informatics in Tokyo, Japan. His research interests include mobile computing and sensing, ad hoc and vehicular networks, distributed systems, and cloud computing. Borcea received his Ph.D. degree from Rutgers University in 2004. He is a member of the ACM and IEEE.