Integrating IoT-Sensing and Crowdsensing with Privacy: Privacy-Preserving Hybrid Sensing for Smart Cities

HANWEI ZHU and SID CHI-KIN CHAU, Australian National University
GLADHI GUARDDIN, Universitas Indonesia
WEIFA LIANG, City University of Hong Kong

Data sensing and gathering is an essential task for various information-driven services in smart cities. On the one hand, Internet of Things (IoT) sensors can be deployed at certain fixed locations to capture data reliably but suffer from limited sensing coverage. On the other hand, data can also be gathered dynamically through crowdsensing contributed by voluntary users but suffer from its unreliability and the lack of incentives for users' contributions. In this article, we explore an integrated paradigm called "*hybrid sensing*" that harnesses both IoT-sensing and crowdsensing in a complementary manner. In hybrid sensing, users are incentivized to provide sensing data not covered by IoT sensors and provide crowdsourced feedback to assist in calibrating IoT-sensing. Their contributions will be rewarded with credits that can be redeemed to retrieve synthesized information from the hybrid system. In this article, we develop a hybrid sensing system that supports explicit user privacy—IoT sensors are obscured physically to prevent capturing private user data, and users interact with a crowdsensing server via a privacy-preserving protocol to preserve their anonymity. A key application of our system is smart parking, by which users can inquire and find the available parking spaces in outdoor parking lots. We implemented our hybrid sensing system for smart parking and conducted extensive empirical evaluations. Finally, our hybrid sensing system can be potentially applied to other information-driven services in smart cities.

CCS Concepts: • **Security and privacy → Privacy-preserving protocols**; **Hardware security implementation**; • **Computer systems organization → Sensor networks;**

Additional Key Words and Phrases: Privacy, IoT, crowdsensing, hybrid sensing, machine learning, crowdsourcing, smart cities, smart parking

ACM Reference format:
Hanwei Zhu, Sid Chi-Kin Chau, Gladhi Guarddin, and Weifa Liang. 2022. Integrating IoT-Sensing and Crowdsensing with Privacy: Privacy-Preserving Hybrid Sensing for Smart Cities. *ACM Trans. Internet Things* 3, 4, Article 31 (September 2022), 30 pages.
https://doi.org/10.1145/3549550

This project was supported by ARC Discovery Project no: GA69027/DP200101985.
Authors' addresses: H. Zhu and S. C.-K. Chau, Australian National University, Canberra ACT 2601, Australia; emails: {henry.zhu, sid.chau}@anu.edu.au; G. Guarddin, Universitas Indonesia, Pondok Cina, Beji, Depok City, West Java 16424, Indonesia; email: adin@cs.ui.ac.id; W. Liang, City University of Hong Kong, 83 Tat Chee Ave, Kowloon Tong, Hong Kong; email: weifa.liang@cityu.edu.hk.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2022 Copyright held by the owner/author(s).
2577-6207/2022/09-ART31
https://doi.org/10.1145/3549550

1 INTRODUCTION

Smart cities improve the quality of life by offering information-driven services. For instance, intelligent transportation services can adapt to dynamic users' demands for service optimization. To enable responsive and efficient information-driven services, it relies critically on timely and comprehensive data sensing and gathering to synthesize the global knowledge of user usages, behaviors, and patterns for service planning and management. For example, one of the key information-driven services is to enable smart parking in open spaces. According to a recent survey [25], searching for available parking spaces is one of the topmost crucial activities of city dwellers. Currently, a large number of parking spaces are located in open areas (e.g., streets and campuses), with limited knowledge of their instantaneous statuses. Without sufficient knowledge of available parking spaces, drivers often spend numerous hours cruising around and competing for limited parking spaces. A more effective solution is to gather the availability data of parking spaces in a temporal and spatial manner from diverse sources and then disseminate the knowledge of available parking spaces to users in a timely fashion.

There are two major approaches for effective data sensing and gathering in smart cities:

— **IoT-sensing:** IoT sensors can be deployed at specific fixed locations to capture their surrounding data reliably. Among the variety of sensors available in the market (e.g., short-range motion detection and long-range visual detection), one of the most common media is vision-based capturing with visual images and videos for object classification and detection, which has been used in monitoring traffic congestions, parking availabilities, and pedestrian flows. Mounted cameras can overlook an ample open space and can be installed flexibly. More importantly, cheaper cameras with higher resolution are increasingly available, which improves the quality and reliability of the captured images. However, IoT sensors may suffer from limited sensing coverage due to either the positioning of the sensors or the amount of deployed sensors. Also, IoT sensors sometimes experience calibration and privacy issues.

— **Crowdsensing:** Data can be gathered through crowdsensing contributed by voluntary users. Users or user-owned sensors can provide their surrounding data in a dynamic fashion. For example, users can report their observations captured by their smartphones. Navigation apps, such as Waze and Google Maps, continuously collect users' mobility data to infer traffic conditions, predict traveling delays, and generate traffic alerts. Some mobile apps allow users to share personal observations, such as public transport waiting times, congestion statuses, parking availabilities, and petrol prices. However, crowdsensing is not always reliable because it is based on the voluntary contributions of users. There is no guarantee that the crowdsensed data is always accurate or up-to-date.

Although IoT-sensing and crowdsensing are different, they are often complementary to each other in terms of coverage and sensing reliability. Hence, there is a potential to harness both approaches. In this article, we explore an integrated paradigm called *hybrid sensing*, by which both IoT-sensing and crowdsensing are utilized to enhance the accuracy, coverage, and reliability of data sensing and gathering. Specifically, the benefits of such integration include:

(1) While IoT sensors are deployed at fixed locations, crowdsensing can offer additional dynamic sensing coverage.
(2) Since crowdsensing is often difficult to initiate due to limited user engagement, IoT-sensed data can be used to bootstrap the process and further attract user contributions.
(3) IoT-sensed data can be calibrated and enhanced based on crowdsourced user feedback as part of crowdsensing.

However, to realize the full potential of hybrid sensing, we need to address the following two major issues explicitly:

— **Privacy:** While sensors are deployed to collect data, they are not supposed to identify individual people or vehicles. There are increasing concerns about privacy violations by these sensors. The images and video footages of pedestrians and vehicles may expose their identities and behaviors, which easily lead to unintended criminal consequences (e.g., stalking and theft). Even if the administrators of these vision-based sensor systems intentionally discard the private data, their systems may still be infiltrated by hackers for malicious purposes. Particularly, there have been several webcam-related hacking and data breach instances where cameras were fully controlled by hackers, and the video footages were leaked to the Internet. On the other hand, users are also wary of sharing private personal data. In particular, the breach of personal privacy (e.g., identity and location information) remains a top concern for data sharing with personal and mobile devices.
— **Incentives:** Ideally, crowdsensing is expected to benefit users by offering global knowledge to them from large amounts of shared data and hence motivates more users to share their personal data to improve the quality of crowdsensing. However, in reality, users are not keen to contribute to crowdsensing because they sometimes can enjoy benefits without any contribution as free riders. Insufficient data sharing will lead to a degraded quality in crowdsensing, undermining the willingness of further voluntary contributions of users. Thus, there is a need to incentivize crowdsensing to provide high-quality user-contributed data.

Remarkably, our hybrid sensing system addresses these issues in a complementary manner. First, our system supports explicit user privacy—IoT sensors are obscured physically to prevent capturing private user data, and users interact with a crowdsensing server via a privacy-preserving protocol to preserve their anonymity. Second, users are incentivized to provide sensing data not covered by IoT sensors and provide additional crowdsourced feedback to assist in calibrating IoT-sensing. Their contributions will be rewarded with credits that can be redeemed to retrieve synthesized crowdsensing information from our system.

Note that hybrid sensing is a broad concept that can be applied in various information-driven services in smart cities, such as reporting garbage disposals, traffic conditions, and wildlife sightings. In this article, we focus on the application of smart parking to demonstrate the usefulness and practicality of our system. Particularly, our hybrid sensing smart parking system enables users to inquire and find geographically distributed available parking spaces.

As illustrated in Figure 1, our smart parking system deploys Raspberry Pi-based fish-eye cameras for capturing the footages of parking spaces, which are obscured by blurry filters to enhance privacy. We developed an onboard machine learning system to recognize objects from captured blurry videos in real-time. On the one hand, complementary to the captured sensor data from the deployed cameras, users can also submit the observed available parking spaces through a mobile app via a privacy-preserving protocol using cryptographic commitments, zero-knowledge proofs, and anonymous credentials.

On the other hand, to improve detection accuracy and to reduce the effort of training, in our machine learning system, we utilized crowdsourced classification data on segmented images from the captured footages to calibrate the machine learning model. The user-contributed data will then be validated by the server for usefulness. To incentivize user contributions, users will obtain credits for their valuable contributions while preserving their anonymity. The credits, as incentives, can be redeemed for parking inquiries. Since not every parking lot has been deployed with camera systems, crowdsensing will be particularly valuable to expand the coverage of IoT-sensing.

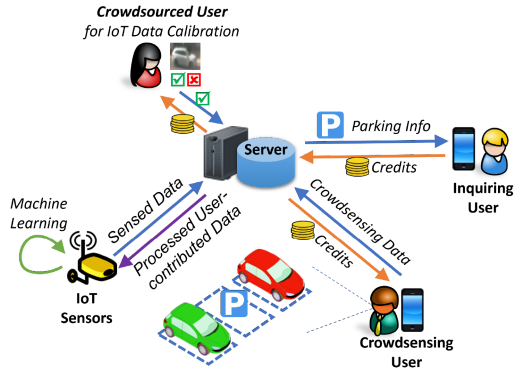


Fig. 1. System design of a hybrid sensing system for smart parking.

Organization: The rest of this article is organized as follows. We first review the related work in Section 2. Then, we present the privacy-preserving IoT-sensing part in Section 3 and the preliminaries of cryptographic components in Sections 4. Based on the cryptographic components, we develop our privacy-preserving crowdsensing protocol in Section 5. Next, we describe our hybrid sensing system implementation in Section 6, and present an evaluation study of the implemented system in Section 7. Finally, Section 8 concludes this article. Some further analysis and technical proofs are included in the Appendix.

2 RELATED WORK

In this section, we introduce the background and related work of our hybrid sensing system. This section is divided into four subsections. The first two subsections provide an overview of the existing smart parking systems and security issues in IoT-sensing. Section 2.3 surveys crowdsensing and its security issues. Finally, Section 2.4 addresses the novelty of this work with a comparison with the existing literature.

2.1 Smart Cities and Smart Parking

Smart cities and smart appliances [1] enrich our daily lives by providing information-driven services and resource management. Since the states of services and resources are dynamic and uncertain, smart cities often require real-time sensing and tracking.

Among the service sectors in smart cities, many projects have developed vision-based smart parking systems with the development of computer vision technologies. Compared with traditional sensor-based approaches (e.g., using infrared [4] or ultrasonic [36] sensors), vision-based systems have drastic improvements in detection accuracy [35]. Moreover, camera lenses are inexpensive and easy to install. Therefore, cameras can be integrated into embedded systems and then connected to the Internet to monitor parking lots in real-time.

Note that most current studies focused on detecting the parking spaces in a parking lot. For example, [31] introduced a monitoring system where four cameras were set up in buildings around a parking lot, and the system can send and display real-time parking information. Similarly, [8] proposed a system that counts the number of cars parked and identifies the vacant spaces. Furthermore, [3] suggested a method based on the coordinates of parking spaces. Compared to the methods above, [20] presented a more robust solution using deep learning. Their system can infer the availability information of a large parking lot within a second. In this article, we applied a machine learning method to infer the real-time occupancy information so that our system can provide a high detection accuracy and be integrated with crowdsourcing.

2.2 IoT-Sensing

As [5] suggested that security and privacy issues tend to be ignored or have not been thoroughly investigated due to the development complexity of integrating both hardware and software components in most IoT applications. In the literature, the typical solutions are data perturbation and cryptography. For example, [22] implemented a privacy-preserving collaborative learning model, using a multiplicative random projection approach to obfuscate training data at each resource-limited IoT object. The authors claimed that their approach outperformed the other approaches based on additive noisification. Moreover, [19] proposed a variational autoencoder which can achieve adversarial model-free anonymization. The results demonstrated the feasibility of anonymizing data in real-time on resource-constrained edge devices. Additionally, [26] presented a privacy-preserving IoT-camera-based target locating system based on homomorphic encryption, which can protect the target's privacy and the video footages provided by IoT cameras. On the other hand, a less common yet powerful and straightforward solution is based on hardware. Hardware-based privacy-preserving methods have an innate advantage in security. Notably, [37] presented a privacy-preserving optical filter that can defocus the images before capture. However, their setup was complicated and thus difficult to deploy on a large scale. Besides, [2] proposed a real-time occupancy recognition system for buildings using a low-cost embedded system. The authors put a frosted lens in front of the camera so that people's identities in the video feed were no longer recognizable, but their experiments did not consider the effects of different blurriness levels.

In parking monitoring applications, only a few papers have considered privacy when designing their systems. For instance, [24] proposed a real-time sensor-based parking monitoring and billing system. The authors suggested that to protect user privacy, the sensors deployed should only broadcast sensitive information, such as the vehicle position, when a vehicle is inside the detection area of a parking lot. Moreover, [7] implemented a smart camera network for parking monitoring, where the authors recommended executing all processing in cameras so that no information is sent out. However, both methods above are still vulnerable to attacks. Inspired by [2], we implemented a hardware-based privacy-preserving IoT-sensing system by detecting vehicle flows in our preliminary work [53]. Based on our previous papers [52, 53], we managed to further improve the method of detection by directly monitoring parking spaces.

2.3 Crowdsensing

Crowdsensing can enable smart cities to adapt to dynamic user behaviors and has been extensively applied to various systems in industries [12]. Notable commercial platforms, like navigation apps (e.g., Waze, Google Maps, and Apple Maps), often collect users' mobility data (with or without users' awareness) to generate global knowledge of traffic conditions and patterns. Users can share various personal observations of services, such as bus arrival times, available parking spaces, and congestion statuses, to help other users plan their trips and activities [16, 40]. There are several projects (e.g., [13, 43]) that incorporate crowdsensing data into smart cities for the purposes of monitoring citizen mobility and managing transportation services.

As data sharing is becoming more prevalent in smart cities, there have been substantial concerns over user privacy. In particular, any data leakage may lead to severe identity exposure since a large number of shared data contain sensitive information such as locations and faces. Therefore, a comprehensive solution is needed to protect users. Similar to IoT-sensing, two main approaches to preserve users' privacy in crowdsensing are data perturbation and cryptography. Among the data perturbation techniques, [46, 48] leveraged k-anonymity to cluster users into groups for anonymity preserving, whereas [23, 45, 49] utilized differential privacy to obfuscate

Table 1. Comparisons Between Our Privacy-Preserving Hybrid Sensing System and Previous Studies

	Crowd-sensing	Privacy Protection	Incentivization	IoT/Camera Sensing	Privacy Protection
[2–4, 8, 20, 31, 35, 36]	×	×	×	✓	×
[26]	×	×	×	✓	Cryptographic
[19, 22]	×	×	×	✓	Data Perturbation
[37]	×	×	×	✓	Optical Hardware
[7, 24]	×	×	×	✓	Software based
[53]	×	×	×	✓	Physical Filter
[40]	✓	×	✓	×	×
[13, 16, 43]	✓	×	×	×	×
[23, 46, 48, 49]	✓	Data Perturbation	✓	×	×
[45]	✓	Data Perturbation	×	×	×
[41, 42, 50, 51]	✓	Cryptographic	✓	×	×
[14]	✓	Cryptographic	×	×	×
[27, 44]	✓	×	×	✓	×
[47]	✓	×	✓	✓	×
This work	✓	Cryptographic	✓	✓	Physical Filter

sensing data and results. However, several studies (e.g.,[17, 30, 34]) have applied various techniques to show the practicality of deanonymizing users' hidden identities and other sensitive information based on their trajectories and other online activities. More importantly, since most data perturbation techniques introduce noises to the dataset, the data utility is reduced. In contrast with data perturbation, other research studies have applied cryptographic techniques (e.g., homomorphic encryption and public-private key signatures [41, 42, 50, 51]) to protect users' privacy. Among all the cryptographic techniques, zero-knowledge proofs has been widely applied in areas such as blockchain (e.g., [11, 15, 28]), secure multi-party computation (e.g., [9, 21]), and electronic voting or auctions (e.g., [18, 38, 39]). In our hybrid sensing system, we applied efficient zero-knowledge proofs, and designed privacy-preserving protocols that ensure user privacy in their participation.

2.4 Comparisons to Our Work

There are previous studies that hybridized crowdsensing with sensor networks in the areas such as pollen detection [44], indoor positioning [27], and urban data collection [47]. However, to the best of our knowledge, this work is the first to integrate IoT-sensing and crowdsensing in parking space monitoring while considering privacy issues for both methods. Particularly, the comparisons between our privacy-preserving hybrid sensing system with the aforementioned papers are tabulated in Table 1.

We highlight the novelty and key contributions of this article as follows:

(1) We implemented a privacy-preserving hybrid sensing system based on parking monitoring using cameras obscured by physical lenses at different blurriness levels.
(2) We applied a machine learning method to predict the occupancy of parking lots whose real-time video feed is blurred. The training process can be integrated with crowdsourcing, which improves detection accuracy while incentivizing voluntary users.
(3) We developed a mobile application for privacy-preserving crowdsensing parking monitoring using cryptographic techniques, which can be integrated with an IoT-sensing system.

3 PRIVACY-PRESERVING IOT-SENSING

In this section, we present the IoT-sensing part of our privacy-preserving hybrid sensing system for parking monitoring. To capture visual data from parking spaces, we deployed Raspberry Pi-based systems equipped with 160° fish-eye cameras, which can offer a wide panoramic field of view. The

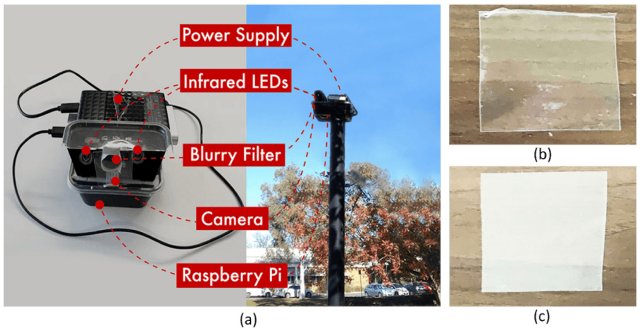


Fig. 2. Hardware setup. (a) The Raspberry Pi camera system. (b) A blurry filter made of frosted glass. (c) A blurry filter made of frosted glass with translucent tape on both sides.

video streams captured have a resolution up to 1280×720, with a frame rate of 25 frames per second. The hardware setup is depicted in Figure 2(a). Each device incorporates a blurry filter, a camera, two infrared LEDs, wireless connectivity, and a battery-based power supply.

3.1 Obscured Cameras with Blurry Filters

In our camera system, the video footages are instantly processed on Raspberry Pi for rapid object recognition and classification. However, the system is inevitably connected to a wireless network for data sharing and hence is susceptible to hacking and attacks. In particular, the captured footages of parking spaces could be leaked. Private information such as vehicle license plates and the faces of pedestrians may be disclosed for unknown criminal purposes. To mitigate the privacy concern of our camera system, we deliberately obscure the cameras by hardware-based filters. We put frosted and blurry filters in front of the cameras to prevent capturing detailed footages of vehicles and pedestrians. Note that hardware-based filters are more robust than any software-based privacy methods, such as software-controlled blurring/resolution reduction or data deletion by discarding footages in system memory. Since the whole camera system could be hijacked by hackers, and any software-based privacy methods would be disabled readily, hardware-based filters provide a stronger privacy guarantee and cannot be circumvented without physical access to the sites.

In this work, we applied various physical blurry filters to create different blurriness levels for the cameras. Figures 2(b) and (c) show the frosted filters used in the system. Particularly, we adopted a piece of frosted glass in (b) and a piece of frosted glass with double-sided translucent tape in (c). The translucent tape can increase light transmittance and hence reduce blurriness, as the clearness of the filters shows. By obscuring one or more filters, we managed to simulate different levels of blurriness, as depicted in Figures 3(a), (b), (c), and (d). It is evident that as the blurriness level increases, it is increasingly hard to recognize any objects in the video. In this work, we consider the effectiveness of a blurred camera-based parking monitoring system. Such a system is considered privacy-preserving if sensitive information (such as the license plates of cars or the identity of the pedestrians) in the videos is not identifiable. According to the above definition, even the least blurred filter, as shown in Figure 3(b), satisfies our privacy preservation requirement.

Next, we quantify the level of blurriness by a suitable metric for subsequent experimental evaluation. We applied a metric in [33] based on edge sharpness because it aligns well with human perception. This metric computes the average of the maximum difference of each pixel compared with its neighboring pixels. Given any blurred image and its original image, the blurriness level of

Fig. 3. Effects of using blurry filters. (a) The original image. (b) Using one frosted filter with translucent tape. (c) Using two frosted filters with translucent tape. (d) Using three frosted filters with translucent tape. (e) Using one frosted filter without translucent tape.

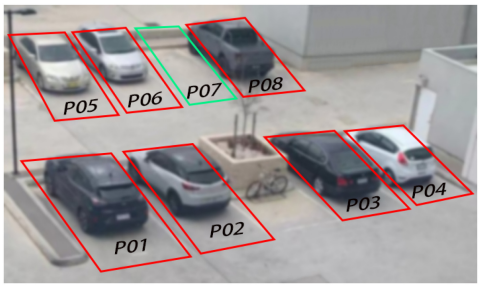


Fig. 4. An example of a captured parking space footage.

the blurred image is defined as follows:

$$\texttt{Blurriness} = \frac{|X - Y|}{X}, \tag{1}$$

$$X = \frac{\sum_{i=1}^{M} \max_{j \in \mathcal{N}_i}(x_i - x_j)}{M}, \tag{2}$$

$$Y = \frac{\sum_{i=1}^{N} \max_{j \in \mathcal{N}_i}(y_i - y_j)}{N}, \tag{3}$$

where, x_i and y_i refer to the pixel values of the ith pixel in the original and blurred images, respectively. M and N are the total numbers of pixels in the original and blurred images. $\mathcal{N}_i$ is the set of neighboring pixels of the ith pixel. Based on the definition above, the blurriness levels are 0.4059, 0.5629, 0.6574, and 0.7967 for the various blurry filters in Figure 3.

3.2 Machine Learning for Parking Space Recognition under Blurry Filters

After applying blurry filters, our system should still be capable of recognizing parking availabilities from camera-based sensors. This section describes the system that utilizes machine learning for parking space recognition executed on Raspberry Pi with blurry filters. Figure 4 illustrates an example of our system capturing blurry footages of parking spaces via obscured cameras. The system administrators will first define a region of interest (namely, the parking spaces) on the screen. Through machine learning, the Raspberry Pi device can detect if there is a vehicle in each parking space (presented in different colors) based on the trained model. Note that the training data can be obtained through both IoT-sensing or crowdsourcing (e.g., Amazon Mechanical Turk).

A high-level flowchart of the recognition algorithm is depicted in Figure 5. The most critical components in the recognition algorithm are detecting the moving objects in the image as well as deciding if there are vehicles in the parking spaces. Specifically, the first method is implemented

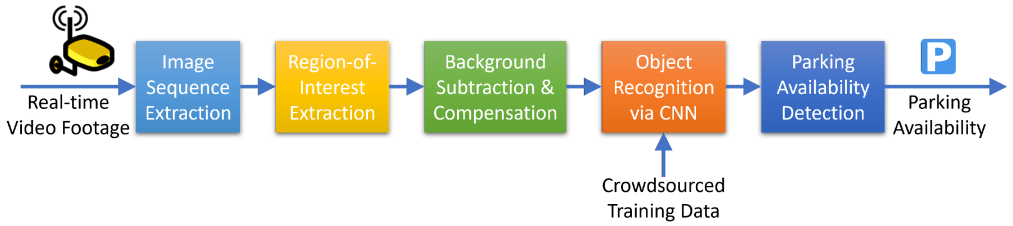


Fig. 5. A flowchart showing the recognition algorithm.

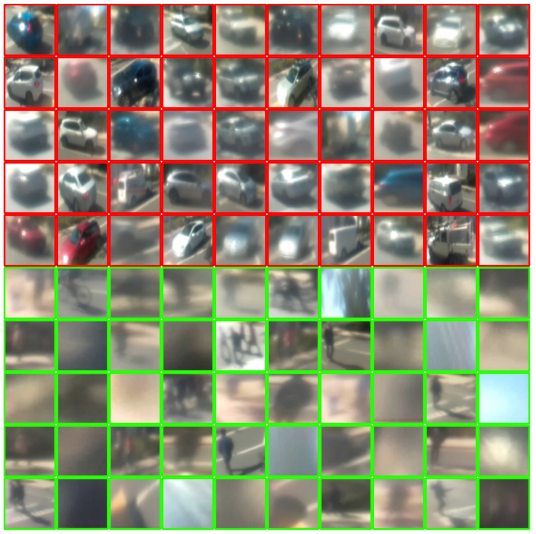


Fig. 6. An example of extracted images with different blurriness levels used in the training sets where pictures with red frames are classified as cars, and pictures with green frames are non-cars.

through background subtraction, where a background image is identified and compared to the current frame of the image. A moving object will be detected if the image is different from the background. For simplicity, we only consider large objects, such as vehicles and pedestrians, moving in the video. The detection will pause if all large objects in the video are observed to be stationary.

To predict the availability of each parking space, we applied a **Convolutional Neural Network (CNN)** classifier, which is widely used in analyzing visual imagery with high accuracy. A typical architecture of CNN consists of an input layer, some convolutional layers, some pooling layers, a connected layer, and an output layer. An example of the training data is shown in Figure 6, where some extracted images are classified as vehicles and non-vehicles. Practically, the cameras can be deployed almost anywhere to capture the training footages given that blurry filters are used. The CNN model will be more accurate and scalable by deploying a large scale of cameras in different locations so that the training set contains diverse images of vehicles pointed with different angles and heights, as illustrated in Figure 6. Figure 7 shows our CNN model, which includes three convolution blocks with a max pooling layer and a ReLU activation function in each of them. A fully connected layer, known as the dense layer, consists of 128 units on top of it and can be activated by a ReLU activation function and an L2 regularization. In our model, there is also a dropout layer to prevent overfitting.

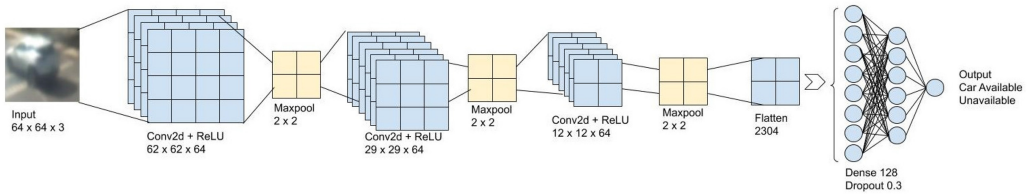


Fig. 7. The CNN model used in this work.

3.3 Crowdsourcing Training Data for Machine Learning

The training data for machine learning can be obtained through crowdsourcing. Specifically, the training datasets can be obtained from either an online crowdsourcing platform or the IoT sensors deployed in parking lots. The acquired datasets can effectively bootstrap the training process,[1] and the continuous data input from the IoT sensors can further improve detection accuracy. In our system, the IoT sensors extract some segmented images from the captured footages (e.g., Figure 6), and users[2] will have the opportunities to contribute to the training sets by classifying whether there is a car or not in a segmented image (which is equivalent to generating labels) via a mobile app. Our system then applies the majority voting technique to aggregate the crowdsourced data and minimize errors in labels. Users will earn credits if they submit labels. More details are provided in Sections 5 and 6.

4 PRELIMINARIES OF CRYPTOGRAPHIC COMPONENTS

Before presenting our privacy-preserving crowdsensing protocol, we introduce some relevant preliminaries of cryptographic components. Our privacy-preserving system relies on several basic cryptographic components, including cryptographic commitments, zero-knowledge proofs, and public-private key signatures. More details can be found in a cryptography textbook [10].

First, we denote $\mathbb{Z}_p = \{0, \ldots, p-1\}$ by the set of integers modulo a large prime number p, which will be used for encrypting private data. For brevity, we simply write "$x+y$" and "$x \cdot y$" for modular arithmetic without explicitly mentioning "$\mathsf{mod}\ p$". We consider a finite group $\mathbb{G}$ of order p. Then, we pick g, h as two generators of $\mathbb{G}$, such that they can generate every element in $\mathbb{G}$ by proper powers, namely, for each $e \in \mathbb{G}$, there exists $x, y \in \mathbb{Z}_p$ such that $e = g^x = h^y$. The classical discrete logarithm assumption states that given g^x, it is computationally hard to obtain x, which underlies the security of many cryptosystems, including ours.

4.1 Cryptographic Commitments

A cryptographic commitment allows a user to hide a secret (e.g., crowdsensed data). We use the Pedersen commitment, which is perfectly hiding (i.e., an adversarial cannot unlock the secret) and computationally binding (i.e., an adversarial cannot associate with another secret in polynomial time). To commit a secret value $x \in \mathbb{Z}_p$, a user first picks a random number $\mathsf{r} \in \mathbb{Z}_p$ to mask the commitment. Then, he computes the commitment by $\mathtt{Cm}(x, \mathsf{r}) = g^x \cdot h^{\mathsf{r}}$. Note that the Pedersen commitment satisfies the homomorphic property: $\mathtt{Cm}(x_1 + x_2, \mathsf{r}_1 + \mathsf{r}_2) = \mathtt{Cm}(x_1, \mathsf{r}_1) \cdot \mathtt{Cm}(x_2, \mathsf{r}_2)$. In this article, we simply write $\mathtt{Cm}(x)$ without specifying r.

[1] In practice, the training process can be bootstrapped using online crowdsourced datasets if the on-site images captured from IoT sensors are insufficient.

[2] We assume only registered users in our hybrid sensing system are allowed to contribute crowdsourced data generated from the IoT sensors because if those data are submitted to any crowdsourcing platform, malicious users could leverage all images available and potentially deanonymize our video streams.

4.2 Zero-Knowledge Proofs

In a zero-knowledge proof, a prover (e.g., crowdsensing user) convinces a verifier (e.g., server) of the knowledge of a secret without revealing the secret (e.g., crowdsensed data). For example, to show the knowledge of (x, r) for $\mathtt{Cm}(x, \mathsf{r})$ without revealing (x, r). A zero-knowledge proof should satisfy completeness (i.e., a prover knowing the secret can always convince the verifier), soundness (i.e., a prover not knowing the secret cannot convince the verifier), and zero-knowledge (i.e., the verifier cannot learn the secret).

4.2.1 Σ-Protocol. A general approach to provide zero-knowledge proofs is based on *Σ-protocol*. This can prove the knowledge of x such that $f(x) = y$ with concealed x, where y and $f(\cdot)$ are revealed and $f(\cdot)$ is not computationally invertible. Suppose $f(\cdot)$ satisfies the homomorphic property: $f(a + b) = f(a) + f(b)$. The Σ-protocol proceeds as follows:

(1) First, the prover sends a commitment $y' = f(x')$ for a random x', to the verifier.
(2) Next, the verifier generates a random challenge β and sends it to the prover.
(3) The prover replies with $z = x' + \beta \cdot x$ (which does not reveal x).
(4) Finally, the verifier checks whether $f(z) \stackrel{?}{=} y' + \beta \cdot y$.

One can show that Σ-protocol satisfies completeness, soundness, and zero-knowledge. Next, we present some examples of Σ-protocol that will be utilized in our protocol.

4.2.2 Zero-Knowledge Proof of Committed Value ($\mathtt{zkpCm}$*).* Given $\mathtt{Cm}(x, \mathsf{r})$, a prover wants to convince a verifier of the knowledge (x, r). Σ-protocol can be applied as follows:

(1) The prover first randomly generates $(x', \mathsf{r}') \in \mathbb{Z}_p^2$ and sends the corresponding commitment $\mathtt{Cm}(x', \mathsf{r}')$ to the verifier.
(2) The verifier sends a random challenge $\beta \in \mathbb{Z}_p$ to the prover.
(3) The prover replies with $z_x = x' + \beta \cdot x$ and $z_{\mathsf{r}} = \mathsf{r}' + \beta \cdot \mathsf{r}$.
(4) The verifier checks whether $g^{z_x} \cdot h^{z_{\mathsf{r}}} \stackrel{?}{=} \mathtt{Cm}(x', \mathsf{r}') \cdot \mathtt{Cm}(x, \mathsf{r})^{\beta}$.

Denote a zero-knowledge proof of committed value for $\mathtt{Cm}(x, \mathsf{r})$ by $\mathtt{zkpCm}[x]$. If a prover wants to convince a verifier of the knowledge of x in $\mathtt{Cm}(x, \mathsf{r})$ with a given random mask r, the prover can set $\mathsf{r}' = 0$ and the verifier can construct $z_{\mathsf{r}} = \beta \cdot \mathsf{r}$ by himself.

4.2.3 Zero-Knowledge Proof of Membership ($\mathtt{zkpMbs}$*).* Given a set $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathtt{Cm}(x, \mathsf{r})$, a prover wants to convince a verifier of the knowledge $x \in \mathcal{X}$ without revealing x. Σ-protocol can be applied as follows:

(1) Suppose $x = x_i \in \mathcal{X}$. The prover first randomly generates $(x'_j, \mathsf{r}'_j) \in \mathbb{Z}_p^2$ and computes the commitment $\mathtt{Cm}(x'_j, \mathsf{r}'_j)$ for all $j \in \{1, \ldots, n\}$. Then, the prover randomly generates $\beta_j \in \mathbb{Z}_p$ for each $j \in \{1, \ldots, n\} \backslash \{i\}$, and computes

$$z_{x_j} = \begin{cases} x'_j + (x_i - x_j)\beta_j, & \text{if } j \in \{1, \ldots, n\} \backslash \{i\} \\ x'_i, & \text{if } j = i \end{cases}$$

Next, the prover sends $(\mathtt{Cm}(x'_j, \mathsf{r}'_j), z_{x_j})_{j=1}^n$ to the verifier.
(2) The verifier sends a random challenge $\beta \in \mathbb{Z}_p$ to the prover.
(3) The prover sets $\beta_i = \beta - \sum_{j \neq i} \beta_j$, then computes $z_{\mathsf{r}_j} = \mathsf{r}'_j + \mathsf{r} \cdot \beta_j$ for all $j \in \{1, \ldots, n\}$, and sends $(\beta_j, z_{\mathsf{r}_j})_{j=1}^n$ to the verifier.
(4) The verifier checks whether $\beta \stackrel{?}{=} \sum_{i=1}^n \beta_j$ and

$$g^{z_{x_j}} \cdot h^{z_{\mathsf{r}_j}} \stackrel{?}{=} \mathtt{Cm}(x'_j, \mathsf{r}'_j) \cdot \left(\frac{\mathtt{Cm}(x, \mathsf{r})}{g^{x_j}}\right)^{\beta_j} \text{for all } j \in \{1, \ldots, n\}$$

Denote a zero-knowledge proof of member for $x \in \mathcal{X}$ by $\mathtt{zkpMbs}[x, \mathcal{X}]$.

4.2.4 Zero-Knowledge Proof of Non-negativity ($\mathtt{zkpNN}$*).* Given $\mathtt{Cm}(x, \mathsf{r})$, a prover wants to convince a verifier of the knowledge of $x \geq 0$ without revealing x. Suppose $x < 2^m$. We aim to prove there exist $(b_1, \ldots, b_m)$ such that $b_i \in \{0, 1\}$ for $i \in \{0, \ldots, m\}$ and $\sum_{i=1}^{m} b_i \cdot 2^{i-1} = x$. Σ-protocol can be applied as follows:

(1) The prover sends $(\mathtt{Cm}(b_i, \mathsf{r}_i))_{i=1}^{m}$ to the verifier, and provides $\mathtt{zkpMbs}[b_i, \{0, 1\}]$ for each b_i to prove that $b_i \in \{0, 1\}$. Also, the prover randomly generates $\mathsf{r}' \in \mathbb{Z}_p$ and sends the commitment $\mathtt{Cm}(0, \mathsf{r}')$ to the verifier.
(2) The verifier sends a random challenge $\beta \in \mathbb{Z}_p$ to the prover.
(3) The prover replies with $z_{\mathsf{r}} = \mathsf{r}' + \beta \cdot (\sum_{i=1}^{m} \mathsf{r}_i \cdot 2^{i-1} - \mathsf{r})$.
(4) The verifier checks whether $h^{z_{\mathsf{r}}} \stackrel{?}{=} \mathtt{Cm}(0, \mathsf{r}') \cdot \mathtt{Cm}(x, \mathsf{r})^{-\beta} \cdot \prod_{i=1}^{m} \mathtt{Cm}(b_i, \mathsf{r}_i)^{\beta \cdot 2^{i-1}}$

Denote a zero-knowledge proof of the non-negativity of x by $\mathtt{zkpNN}[x]$.

4.3 Non-Interactive Zero-Knowledge Proofs

The interactive Σ-protocol can be converted to a non-interactive zero-knowledge proof by Fiat-Shamir heuristics which remove the challenge provided by the verifier. Let $\mathcal{H}(\cdot) \mapsto \mathbb{Z}_p$ be a cryptographic hash function. Given a list of commitments $(\mathtt{Cm}_1, \ldots, \mathtt{Cm}_r)$, one can map to a single hash value by $\mathcal{H}(\mathtt{Cm}_1 | \cdots | \mathtt{Cm}_r)$, where the input is the concatenated string of $(\mathtt{Cm}_1, \ldots, \mathtt{Cm}_r)$. In a Σ-protocol, one can set the challenge $\beta = \mathcal{H}(\mathtt{Cm}_1 | \cdots | \mathtt{Cm}_r)$, where $(\mathtt{Cm}_1, \ldots, \mathtt{Cm}_r)$ are all the commitments generated by the prover prior to the step of verifier-provided challenge (Step 2 of Σ-protocol). Hence, the prover does not need to wait for the challenge from the server but instead generates the random challenge himself. The verifier will generate the same challenge with the same procedure for verification. We denote the non-interactive versions of the previous zero-knowledge proofs by $\mathtt{nzkpCm}$, $\mathtt{nzkpSum}$, $\mathtt{nzkpMbs}$, and $\mathtt{nzkpNN}$, respectively.

4.4 Public-Private Key Signatures

Cryptographic signatures can verify the authenticity of some given data. Suppose a signer has a pair of public and private keys $(K^{\mathsf{p}}, K^{\mathsf{s}})$ for an asymmetric key cryptosystem (e.g., RSA). To sign a message m, the signer first maps m by a cryptographic hash function $\mathcal{H}(m)$ (e.g., SHA-3). Then, the signature of m is the encryption $\mathtt{sign}_{K^{\mathsf{s}}}(m) = \mathtt{Enc}_{K^{\mathsf{s}}}[\mathcal{H}(m)]$. Given (m, K^{p}), anyone can verify the signature $\mathtt{sign}_{K^{\mathsf{s}}}(m)$ by checking whether the decryption $\mathtt{Dec}_{K^{\mathsf{p}}}[\mathtt{sign}_{K^{\mathsf{s}}}(m)] \stackrel{?}{=} \mathcal{H}(m)$.

5 PRIVACY-PRESERVING CROWDSENSING

In this section, we present the crowdsensing part with a privacy-preserving crowdsensing protocol, which is complementary to the privacy-preserving IoT-sensing system presented in Section 3. The protocol draws on the basic cryptographic components of cryptographic commitments, zero-knowledge proofs, and public-private key signatures.

5.1 Threat Model

Our crowdsensing system aims to preserve the privacy of contributing users while being robust against the possible attacks of dishonest users. Firstly, dishonest users may submit multiple (sometimes inconsistent) crowdsensed data entries of a single parking space to the server. Since our protocol is privacy-preserving, it should not reveal users' identities, which enables dishonest users to potentially earn more credits. Hence, the submitted crowdsensed data entries should be verified to prevent duplication and inconsistency while preserving user anonymity. Secondly, dishonest users may attempt to claim more credits for their contributed data than they ought to. We need

Table 2. Notations Used in the Protocol

K^{p}	Public key of the server
K^{s}	Private key of the server
$\mathtt{Cm}(x, \mathsf{r})$	Cryptographic commitment of x and random mask r
Q	Credential of a user
$\mathtt{sign}_Q$	Server's signature on the credential Q
s	User's secret key
q	Credential identifier
b	User's balance of credits
b_0	User's default initial balance
c_{q}	Credits required for each parking availability inquiry
R	Crowdsensed data entry of a user for the jth parking space at time t
$\tau^{j,t}$	User's ticket associated with each data entry
$a^{j,t}$	User's indicator of the observed availability of the jth parking space at time t
$v^{j,t}$	Aggregate outcome of the submitted data entries $\{a^{j,t}\}$
$c^{j,t}$	Eligible credit of the data entry for ticket $\tau^{j,t}$
$\mathbb{T}_{\mathsf{D}}$	Table of crowdsensed data entries
$\mathbb{T}_{\mathsf{C}}$	Table of eligible credits
$\mathbb{T}_{\mathsf{Q}}$	Table of used credentials

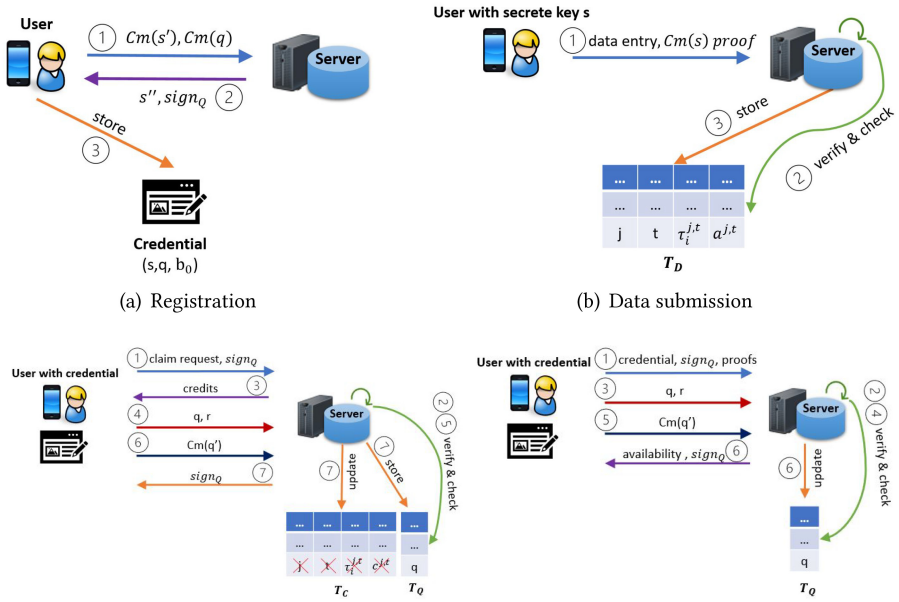
to ensure the consistency of credit claiming with respect to users' actual contributions, subject to anonymity. In particular, our protocol supports the following requirements of privacy-preserving crowdsensing:

(1) **Anonymity:** The identities of users should not be revealed or leaked from the server data. However, all data may be associated with only randomized identification.
(2) **Unlinkability:** Each data entry cannot be linked to reveal the data sources. One cannot distinguish whether the data entries are from different users or the same user.
(3) **Cheating Prevention:** The protocol should be able to prevent cheating behaviors (e.g., duplicate data submissions, inconsistent credit claims), even though the data are anonymized and unlinkable to a specific user.
(4) **Incentive Support:** Users can still earn appropriate credits with verification for their useful data contributions in a privacy-preserving manner.

Note that there are several assumptions in our model. We only consider anonymity in the server, concerning the recorded persistent data, not in the communication channels. Also, we assume that the communication channels are protected from external attacks, such as eavesdropping, IP hijacking, and man-in-the-middle attack. Finally, users can use proper security communication systems to ensure anonymity and security in the communication channels, such as anonymous routing and IPSec.

5.2 Privacy-Preserving Protocol

In this subsection, we present the details of our privacy-preserving protocol between the server and users. Our protocol ensures anonymity, unlinkability, cheating prevention, and incentive support. Some key notations are listed in Table 2. There are five stages in our protocol, which are system setup, user registration, crowdsensed data submission, credit claiming, and result inquiry. See Figure 8 for an illustration of the key stages in the privacy-preserving crowdsensing protocol.



(a) Registration (b) Data submission

(c) Credit claiming (d) Result inquiry

Fig. 8. An illustration of the key stages in the privacy-preserving crowdsensing protocol. (a) The registration stage. (b) The data submission stage. (c) The credit claiming stage. (d) The result inquiry stage.

5.2.1 System Setup. First, the server generates the cryptographic parameters and notifies all users. The server chooses a proper $\mathbb{Z}_p$ and a finite group $\mathbb{G}$ of order p with generators g, h. The server also sets an appropriate collision-resistant cryptographic hash function $\mathcal{H}(\cdot)$. The cryptographic parameters should be decided to meet the desired security and efficiency requirements.

5.2.2 User Registration. When a new user joins the crowdsensing system, the user needs to register with the server. After registration, the user obtains a credential $Q = (s, q, b)$, which consists of a secret key s (used to claim credits), a credential identifier q (used to verify the freshness of the credential and prevent double-spending), and a balance of credits b. The steps of this stage are illustrated in Figure 8(a). The details of the steps are described as follows.

Registration Stage Details:

(1) A new user randomly generates $(s', q) \in \mathbb{Z}_p^2$, and then sends cryptographic commitments $\texttt{Cm}(s')$ and $\texttt{Cm}(q)$ to the server.
(2) The server randomly picks $s'' \in \mathbb{Z}_p$, and computes $\texttt{Cm}(s) = \texttt{Cm}(s') \cdot \texttt{Cm}(s'', 0)$. Then, the server generates a signature $\texttt{sign}_Q = \texttt{sign}(\texttt{Cm}(s)|\texttt{Cm}(q)|\texttt{Cm}(b_0))$, where b_0 is the default initial balance. Next, the server sends $(s'', \texttt{sign}_Q)$ to the user.
(3) The user computes $s = s' + s''$, and stores $Q = (s, q, b_0)$ with $\texttt{sign}_Q$. Here, s and q are not known by the server.

Note that secret key s is jointly generated by a user and the server to prevent replay attack since any user may reuse another user's $\texttt{Cm}(s)$ and $\texttt{Cm}(q)$ in the registration to masquerade another user.

5.2.3 Crowdsensed Data Submission. After the registration, the user can submit crowdsensed data (i.e., observed parking availabilities) to the server. A crowdsensed data entry will be associated with a secret key s in a privacy-preserving manner. The user can claim credits for verified useful

contributions subsequently by s. Let $a^{j,t}$ be a user's observed indicator variable of the availability of the jth parking space at time t. $a^{j,t} = 1$ indicates that the parking space is available. Otherwise, $a^{j,t} = 0$. The steps of this stage are illustrated in Figure 8(b). The details of the steps are described as follows.

Data Submission Stage Details:

(1) The user computes $\tau^{j,t} = \mathtt{Cm}(s, \mathcal{H}(j|t))$. The user also generates $\mathtt{nzkpCm}[s]$ for $\mathtt{Cm}(s, \mathcal{H}(j|t))$ with a known random mask $\mathcal{H}(j|t)$.
(2) The user sends the data entry $R = (j, t, \tau^{j,t}, a^{j,t})$ along with $\mathtt{nzkpCm}[s]$ to the server.
(3) The server verifies $\mathtt{nzkpCm}[s]$, and checks if there is a duplicate data entry with the same $(j, t, \tau_i^{j,t})$ on the table of crowdsensed data entries $\mathbb{T}_{\mathtt{D}}$. The server also checks time t. R will be rejected, if $\mathtt{nzkpCm}[s]$ fails, or there is a duplicate entry on $\mathbb{T}_{\mathtt{D}}$, or t is not the recent time slot.

Note that the ticket $\tau^{j,t}$ using $\mathcal{H}(j|t)$ for the random mask is to prevent a user from submitting duplicate data entries. The secret key s in $\mathtt{Cm}(s, \mathcal{H}(j|t))$ ensures that the user is the only one who can claim the credit subsequently.

5.2.4 Credit Claiming. The users can claim credits from their verified contributions, which can be redeemed when inquiring about the results of the crowdsensing system. The server will periodically update $v^{j,t}$, the aggregate outcome of the submitted data entries $\{a^{j,t'}\}$. A simple aggregation is based on the majority vote of $\{a^{j,t'} : |t - t'| \leq \epsilon\}$. Namely, the server takes a majority vote on the set of submitted data entries within time interval $[t - \epsilon, t + \epsilon]$. If the user's data entry matches the outcome of the majority vote of all users, i.e., $a^{j,t'} = v^{j,t}$ for $t' \in [t - \epsilon, t + \epsilon]$, then the contributed users will be eligible for credits.

Next, the server will add $C = (j, t, \tau^{j,t}, c^{j,t})$ to the table of eligible credits $\mathbb{T}_{\mathtt{C}}$, where $c^{j,t}$ is the credits for the data entry with corresponding ticket $\tau^{j,t}$. An updated credential with a new credential identifier and a new balance will be created. A user with ticket $\tau^{j,t}$ can claim the respective credits in the protocol. The steps of this stage are illustrated in Figure 8(c). The details of the steps are described as follows.

Credit Claiming Stage Details:

(1) The user submits claim request $(j, t, \tau^{j,t})$ to the server, along with credential commitments $(\mathtt{Cm}(s), \mathtt{Cm}(q), \mathtt{Cm}(b))$, signature $\mathtt{sign}_Q$, and $\mathtt{nzkpCm}[s]$.
(2) The server verifies $\mathtt{sign}_Q$ and $\mathtt{nzkpCm}[s]$ based on $(\mathtt{Cm}(s), \mathtt{Cm}(q), \mathtt{Cm}(b))$. The claim will be rejected, if $\mathtt{sign}_Q$ or $\mathtt{zkpCm}[s]$ fails.
(3) The server returns $C = (j, t, \tau^{j,t}, c^{j,t})$ for the corresponding ticket $\tau^{j,t}$ to the user.
(4) The user proceeds to update his credential by revealing $(q, \mathtt{r})$ in $\mathtt{Cm}(q)$ to the server.
(5) The server checks if $(q, \mathtt{r})$ are committed values of $\mathtt{Cm}(q)$, and check q in the table of used credentials $\mathbb{T}_{\mathtt{Q}}$. The claim will be rejected, if $\mathtt{Cm}(q)$ check fails or q is already on $\mathbb{T}_{\mathtt{Q}}$.
(6) The user randomly generates $q' \in \mathbb{Z}_p$ and sends $\mathtt{Cm}(q')$ to the server.
(7) The server first computes $\mathtt{Cm}(b') = \mathtt{Cm}(b) \cdot \mathtt{Cm}(c^{j,t}, 0)$. Then, the server generates signature $\mathtt{sign}_Q = \mathtt{sign}(\mathtt{Cm}(s)|\mathtt{Cm}(q')\ |\mathtt{Cm}(b'))$, sends $\mathtt{sign}_Q$ to the user, adds q to $\mathbb{T}_{\mathtt{Q}}$, and removes $C = (j, t, \tau^{j,t}, c^{j,t})$ from $\mathbb{T}_{\mathtt{C}}$.

Note that the server needs to update $\mathbb{T}_{\mathtt{Q}}$ and $\mathbb{T}_{\mathtt{C}}$ after each claim to prevent submitting duplicate claims.

5.2.5 Result Inquiry. A user can make inquiry about the results (i.e., the availability of parking spaces) by spending an amount of c_q credits. The user needs to prove that his balance in the credential is sufficient, namely, $b - c_q \geq 0$, in a privacy-preserving manner. An updated credential with a new credential identifier and a new balance will be created. The steps of this stage are illustrated in Figure 8(d). The details of the steps are described as follows:

Result Inquiry Stage Details:

(1) The user submits credential commitments ($\mathtt{Cm}(s)$, $\mathtt{Cm}(q)$, $\mathtt{Cm}(b)$), signature $\mathtt{sign}_Q$, $\mathtt{nzkpCm}[s]$, and $\mathtt{nzkpNN}[b - c_q]$ to the server.
(2) The server checks $\mathtt{sign}_Q$, $\mathtt{nzkpCm}[s]$ and $\mathtt{nzkpNN}[b - c_q]$. The inquiry will be rejected, if checking $\mathtt{sign}_Q$, $\mathtt{nzkpCm}[s]$ or $\mathtt{nzkpNN}[b - c_q]$ fails.
(3) The user proceeds to update his credential by revealing $(q, \mathtt{r})$ in $\mathtt{Cm}(q)$ to the server.
(4) The server checks if $(q, \mathtt{r})$ are committed values of $\mathtt{Cm}(q)$, and checks q in the table of used credentials $\mathbb{T}_Q$. The claim will be rejected, if $\mathtt{Cm}(q)$ check fails or q is already on $\mathbb{T}_Q$.
(5) The user randomly generates $q' \in \mathbb{Z}_p$ and sends $\mathtt{Cm}(q')$ to the server.
(6) The server first returns the requested availability information. Next, the server computes $\mathtt{Cm}(b') = \mathtt{Cm}(b) \cdot \mathtt{Cm}(-c_q, 0)$, generates $\mathtt{sign}_Q = \mathtt{sign}(\mathtt{Cm}(s)|\mathtt{Cm}(q')|\mathtt{Cm}(b'))$, sends $\mathtt{sign}_Q$ to the user, and adds q to $\mathbb{T}_Q$.

Finally, our crowdsensing protocol can be shown to satisfy anonymity and unlinkability, and we will present the detailed technical proofs in the Appendix.

6 SYSTEM IMPLEMENTATION

In this section, we present the system implementation of the hybrid sensing system, which includes a hardware prototype based on a Raspberry Pi system and a software prototype realized in a mobile app.

6.1 Hardware Prototype

We implemented IoT-sensing using Raspberry Pi 4B and deployed several Raspberry Pi 4Bs in one parking lot on campus. The Raspberry Pi consists of 4 × Cortex-A72 @ 1.5 GHz as its CPU and Broadcom VideoCore VI @ 500 MHz as its GPU. The model also includes a b/g/n/ac dual-band 2.4/5 GHz WiFi module, enabling real-time data transmission. We used Ubuntu 20.04 (64-bit) as the Raspbian operating system due to its efficiency compared with the Raspberry Pi OS (32-bit).

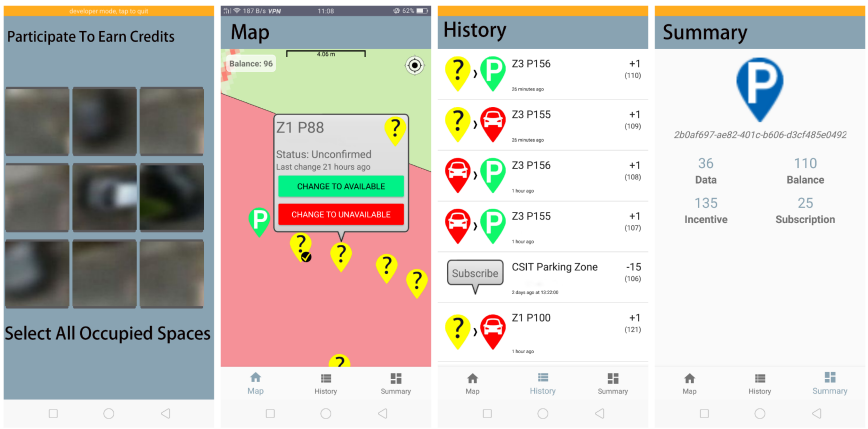
6.2 Software Prototype

We developed a software prototype for crowdsensing at various parking spaces on the campus. The frontend was implemented using OpenStreetMap with parking spaces marked. RabbitMQ was used to provide connectivity to the backend. The backend was built upon Python Django Rest API, and the cryptographic protocols were implemented using Java Cryptography Library with Py4J Library as a bridge to Django. The CNN model was developed based on Python 3.7, OpenCV 4.2, and TensorFlow 2.3. The training process was done using Google Colab. The detailed specifications of the CNN model are tabulated in Table 3.

Figure 9 presents the user interfaces of the mobile app. Users are able to view the vacancy information of a particular parking lot on the campus if they are around by using credits. There are two ways to gain credits in the system. Firstly, credits can be gained through crowdsourcing. For instance, a user can contribute to the training set when he identifies the occupied parking spaces within some random selected captured images from the server, as depicted in Figure 9(a). Another way to gain credits is through on-site crowdsensing. For instance, a user can submit the availability

Table 3. Summary of the CNN Model

Layer (type)	Output Shape
Input layer	(64, 64, 3)
2D convolution layer	(62, 62, 64)
Max pooling 2D layer	(31, 31, 64)
2D convolution layer	(29, 29, 64)
Max pooling 2D layer	(14, 14, 64)
2D convolution layer	(12, 12, 64)
Max pooling 2D layer	(6, 6, 64)
Flatten	2304
Dense layer	128
Dropout layer	128
Dense layer	1



(a) Crowdsourced data (b) Crowdsensed data (c) History (d) Credits

Fig. 9. User interfaces of the mobile app. (a) Submitting crowdsourced training data. (b) Viewing and updating parking space availabilities. (c) Viewing user history. (d) Displaying credit information in an user account.

information of specific parking spaces, as shown in Figure 9(b). Once the user-submitted data are processed by the server, the parking space will be marked as red if it is unavailable, green if it is available, or yellow if it is unconfirmed.[3] Note that we employ a simple majority voting mechanism to determine the status of a parking space. To incentivize participation, users will be credited when the submitted crowdsourced labels are consistent with the majority vote of all submitted crowdsensed data. Moreover, Figure 9(c) depicts the history of user activities, such as reporting availability information and credits used or spent. Finally, Figure 9(d) shows the summary page where users can view their current credits.

7 EVALUATION STUDY

This section presents an evaluation study of the feasibility, practicality, effectiveness, and scalability of our hybrid sensing system. We designed and conducted experiments for each component

[3]The status of a parking space will be unconfirmed if the voting results do not reach a majority or there are insufficient votes for a long time.

of our system, namely, the IoT-sensing, the crowdsourcing, and the hybrid sensing mobile app. This section is divided into four subsections. In each of the first three subsections, we present an evaluation of a component accordingly, and the last subsection includes a case study of our hybrid sensing system.

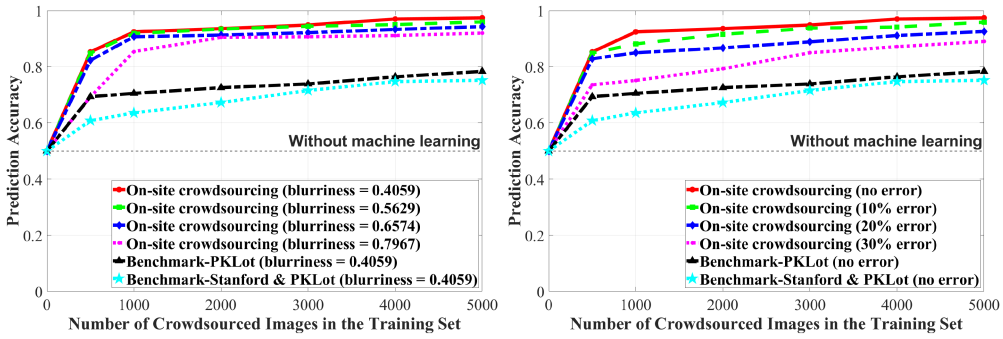
7.1 Privacy-Preserving IoT-Sensing Evaluation

First, we thoroughly investigate the prediction accuracy of our privacy-preserving IoT sensors embedded with an onboard machine learning model considering the effects of different blurriness levels and crowdsourced training datasets.

7.1.1 Experimental Setup. To simulate the operations in practical scenarios and to investigate the accuracy of our CNN model, we extracted 7,100 images from a video stream captured by a Raspberry Pi for each blurriness level, as shown in Figure 3. Among all the images collected, half of them (3,550) consisted of cars, which were labeled as cars, and the other half consisted of other objects (e.g., roads, pedestrians, and trees), which were labeled as non-cars. An example of the images collected is depicted in Figure 6. In reality, the on-site images can serve as the crowdsourced data collected from users in our mobile app. In addition, we utilized images from the Stanford Cars dataset [29] and the PKLot dataset [6] to generate the crowdsourced data for benchmarking the performance of the crowdsourced data collected on-site. Specifically, the Stanford dataset consists of 16,185 high-quality images of 196 classes of cars, while the PKLot dataset is made up of 12,416 extracted images of parking lots. We created a dataset with randomly picked 3,550 images of cars and 3,550 images of non-cars from the PKLot dataset. Because the Stanford dataset only contains images of cars, we created another dataset with 3,550 images of cars from Stanford and 3,550 images of non-cars from PKLot. For simplicity, we named the datasets as PKLot and Stanford, respectively. It is worth noting that one significant difference between the images from PKLot and Stanford and our on-site images is that images from PKLot and Stanford are not obscured by physical filters. Although the images from PKLot and Stanford may not capture the car features in a specific parking lot, the abundance of these images can be used to benchmark the prediction accuracy of our system as compared to on-site training images, and can bootstrap the training process without a need of sufficient IoT sensors.

For the on-site training images with the same blurriness level, we divided them into 6 training sets with different numbers of images, a validation set, and a test set. Particularly, we created training sets with 500, 1,000, 2,000, 3,000, 4,000, and 5,000 images, a validation set with 1,400 images, and a test set with 700 images. Note that the training sets with more images contain the training sets with fewer images. For example, the training set with 1,000 images comprises the training set with 500 images. For the sake of simplicity, we used the same validation set and test set for each training set under the same blurriness level. For PKLot and Stanford, we repeated the same process when allocating the images, except that the test sets were from our on-site images. As a result, the training image sizes are identical in all datasets. In our hybrid sensing model, we further assume that each user only contributes one image label to the training set. In other words, the number of training images is equivalent to the number of participants submitting crowdsourced data. Finally, we plotted the prediction accuracy for each dataset.

7.1.2 Experimental Results and Discussions. Figure 10(a) illustrates the prediction accuracy when there are a different number of crowdsourced images in the training set with different blurriness levels and different datasets. Additional results on training accuracy can be found in the Appendix. For the on-site crowdsourcing datasets with different blurriness levels, it is evident that the prediction accuracy all increases from 0.5 to above 0.9 with the initial 2,000 images. Then, the accuracy converges when there are more crowdsourced data. Moreover, the higher the



(a) Effect of blurriness and different datasets (b) Effect of errors

Fig. 10. Prediction accuracy of the CNN model when there are different numbers of crowdsourced data in the training set considering. (a) Different blurriness levels and crowdsourced datasets. (b) Different error rates in the on-site dataset with the blurriness level of 0.4059.

blurriness is, the lower the prediction accuracy, which aligns with intuition. It is worth mentioning that the CNN model could still achieve a prediction accuracy of 0.92 even when the blurriness level reaches 0.7967. This is because although the testing images are highly blurred, the training images have the same blurriness levels, which ensures that the model learns all the blurred features. In contrast, if the testing images are highly blurred but the training images are sharp, the prediction accuracy will decrease more (as illustrated in the Appendix). As a result, both online datasets perform worse than the on-site ones even though the blurriness level in the test images is only 0.4059. However, the model using both datasets can still reach an accuracy of almost 0.8, which we believe is sufficient for most real-world hybrid sensing models to start with. Therefore, we remark that obtaining online crowdsourced training images is applicable and practical when the IoT sensors need to be deployed on a large scale within a short time. In practice, to further improve the prediction accuracy of a specific sensor in a parking lot, we can collect more images from that parking lot. In our hybrid sensing system, more on-site images can be obtained through user contribution, as demonstrated in Figure 9(a).

7.2 Crowdsourcing Benefits to IoT-Sensing

In our hybrid sensing system, there is a need to carefully understand the potential benefits of crowdsourcing to enhance the accuracy of IoT-sensing. The accuracy of the IoT sensors depends directly on the quality of the training data. We have identified three possible scenarios where mislabeled data may exist in the training datasets: (1) malicious users may submit wrong information deliberately, (2) honest users may submit wrong information accidentally, and (3) when the crowdsourced images are highly blurred, it is difficult to differentiate between cars and non-cars. Therefore, it is crucial to understand how inaccurate data may impact our CNN model.

To simulate the errors obtained from the crowd, we adopted the same on-site training sets, validation sets, and test sets as in Section 7.1 but randomly interchanged 10%, 20%, and 30% of the labels (i.e., labeling cars as non-cars and vice versa) in the training sets. Similarly, we plotted the prediction accuracy of the CNN model using 500, 1,000, 2,000, 3,000, 4,000, and 5,000 training images.

Figure 10(b) displays the prediction accuracy of the CNN model when there are different levels of error percentages in the training set with a blurriness level of 0.4059. The prediction accuracy with various error rates under all blurriness levels can be found in the Appendix. It is evident

that the prediction accuracy with different error percentages all shows a rapidly growing trend when the number of crowdsourced images available in the training set increases from 0 to 1,000, and gradually converges to 90% or above when more training images become available. Also, it is noticeable that more errors there are in the crowdsourced images, the lower the prediction accuracy. However, when there are 30% errors in the training set, the prediction accuracy can still reach almost 90%, which is around 10% higher than the benchmarking datasets with no mislabeling error. This implies that our CNN model is capable of handling noises and errors in the training sets well. Additional experimental results are plotted according to different levels of blurriness in the Appendix.

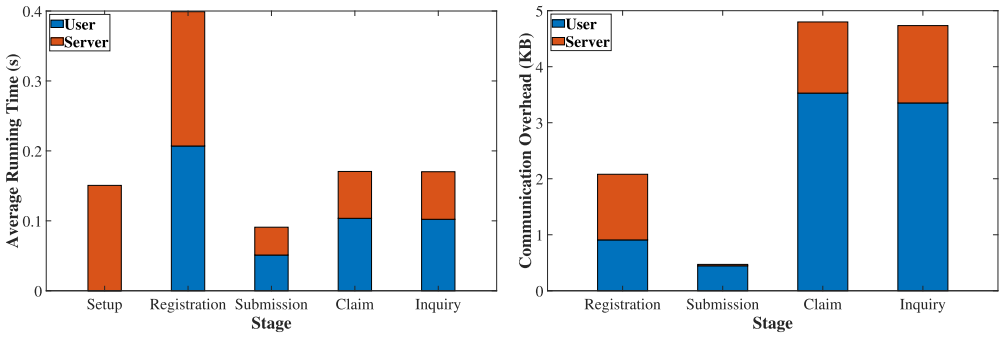
7.3 Performance of Hybrid Sensing System

Next, we evaluate the performance of our hybrid sensing app. We recorded the running time, the communication overhead, and the parking availability of our protocol between a server and a user device. We used a Raspberry Pi 4B with Quad-core Cortex-A72 and 4 GB of RAM as the user device. A Linux laptop with a 1.8 GHz 4-Core Intel Core i7-8550U processor, and 8 GB of RAM was used as the server. All the results were averaged over 10 instances.

7.3.1 System Running Time. We measured the running time of each stage in our protocol. Note that the number only reflects the time taken for computation without considering network latency. As shown in Figure 11(a), all the data in each stage could be processed within a short time. Specifically, the running time in the registration stage is the longest among all stages, with a value of around 0.4 seconds in total. This is because the stage involves generating 2 secret keys s and s' and computing 4 commitments $\mathtt{Cm}(s')$, $\mathtt{Cm}(q)$, $\mathtt{Cm}(s)$, and $\mathtt{Cm}(b_0)$. The running time of the rest stages is relatively small (all under 0.2 seconds), with the data submission stage being the shortest among all stages, which costs a total of 0.1 seconds. Moreover, it is noticeable that the running time on the server-side is slightly shorter than the user-side for most stages since the computing power of the Raspberry Pi is lower than the laptop, and users need to generate more information and send it to the server, as illustrated through the communication overhead shown below.

7.3.2 Communication Overhead. Figure 11(b) shows the average total volume of the transmission data in the registration, data submission, credit claiming, and inquiry stages. Note that the setup stage does not incur any communication cost. We observe that the overhead in all stages are quite small, and the overhead generated from the server-side are much smaller than the user-side in all stages except registration (0.9 KB by the user and 1.2 KB by the server) because users need to generate and send non-interactive zero-knowledge proofs. In addition, it is evident that the data submission stage incurs the least total overhead, with a size of about 0.5 KB, while the credit claiming stage and inquiry stage generate the most data, with a size of around 4.8 KB and 4.7 KB, respectively. Hence, the total data volume in the credit claiming and inquiry stage dominates the entire protocol.

7.3.3 Confirmation Time of Crowdsensing. To evaluate the performance of our privacy-preserving crowdsensing system, we focus on one crucial performance metric—the confirmation time of crowdsensed data in the server considering both IoT-sensing and crowdsensing. To simulate data arrivals in IoT-sensing and crowdsensing, we randomly generated 1, 5, 10, 15, 30, and 50 users and let those users send random availability information to the server simultaneously for the same parking space or a number of different parking spaces. Note that the IoT sensors, which regularly publish availability information, can serve as users and contribute crowdsensed data as well. Figure 12 displays the total parking availability confirmation time incurred. The



(a) Running time (b) Communication overhead

Fig. 11. Communication and computation cost of the hybrid sensing app. (a) The running time in different stages in the protocol. (b) The communication overhead in different stages in the protocol.

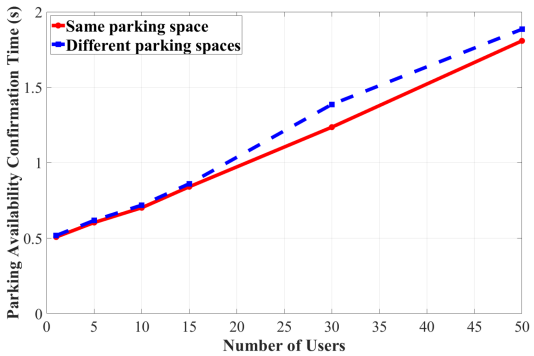


Fig. 12. Parking availability confirmation time when different number of users submit crowdsensed information simultaneously.

confirmation time demonstrates a linearly growing trend with increased users. When all users submit parking availability information for one parking space, the confirmation time starts from around 0.51 seconds with 1 user to around 1.81 seconds with 50 users. On the other hand, when users submit availability data for different parking spaces, the total confirmation time will be slightly higher, ranging from 0.52 ms (2% more) to 1.88 ms (around 4% more). Overall, we observe that our hybrid sensing app can process users' data and update the availability information accordingly within a short period, even when several users are sending requests at the same time.

7.4 A Hybrid Sensing Case Study

Finally, we present a case study of our hybrid sensing system to study the benefits of IoT-sensing and crowdsensing. We considered a parking lot with 4 parking spaces (namely, parking spaces 1−4) where parking spaces 1 and 2 could be covered by both IoT-sensing and crowdsensing. In contrast, parking spaces 3 and 4 could only be covered through crowdsensing. The setting was designed on purpose to simulate the scenario where some parking spaces in a parking lot cannot be detected by cameras due to coverage issues. We deployed a Raspberry Pi-based camera obscured by a blurry filter with a blurriness level of 0.4059. Meanwhile, we asked several volunteers to observe and

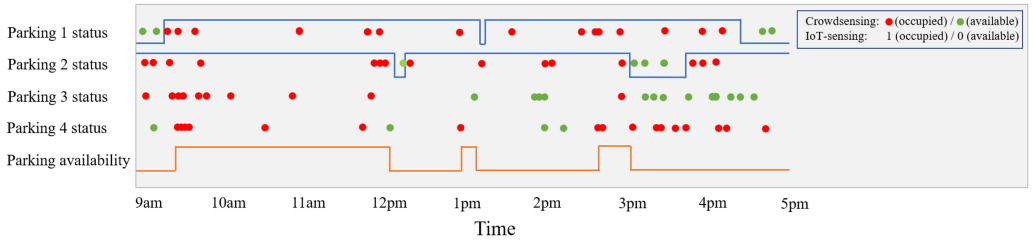


Fig. 13. A case study showing the parking availability information of a parking lot from 9am to 5pm.

contribute to the crowdsensed data for all parking spaces. We plotted the parking availability over time for each parking space and the aggregated availability for the whole parking lot. The overall parking availability was aggregated based on whether at least one parking space was available at a given time. The study was conducted during 9 am and 5 pm on a workday on our campus.

As shown in Figure 13, in parking spaces 1 and 2, the IoT sensor could provide continuous sensing data as illustrated in the binary sequences where a status of 1 represents occupied, and a status of 0 represents available. In contrast, the availability of parking spaces 3 and 4 was only reported by intermittent crowdsensing data, represented by red and green points. Due to the nature of data arrivals (i.e., IoT-sensing provides continuous data and crowdsensing provides intermittent data), it is obvious that IoT sensors can provide better reliability when the availability of a parking space changes but there is no crowdsensed data. For example, when parking space 2 became available right after 3 pm, the IoT sensor could detect it immediately, whereas there was a short delay before the arrival of the next crowdsensed data. Consequently, the overall parking availability could capture the sudden change shortly. However, although crowdsensed data are less reliable compared to IoT-sensed data, crowdsensing can provide a high sensing coverage. For instance, parking spaces 3 and 4 could only be covered through crowdsensing in our case study, and in practice, crowdsensing can reach more parking lots where IoT sensors are not yet available.

8 CONCLUSION

This article presents a privacy-preserving hybrid sensing system with an application for smart parking availability monitoring. We integrated IoT-sensing with crowdsensing, enhanced with privacy-preserving techniques. In particular, we obscured IoT sensors with physical blurry filters in IoT-sensing and applied a cryptographic solution based on cryptographic commitments, zero-knowledge proofs, and anonymous credentials in crowdsensing. We also trained a machine learning model for parking recognition under blurry filters using crowdsourcing. Proof-of-concept prototypes, including a Raspberry Pi system and a mobile app, were developed, and an evaluation study of the machine learning model and the effects of crowdsourcing were presented in the article.

In future work, we will address further research challenges. For instance, how to extend the functionalities of a hybrid sensing system to the setting of heterogeneous sensors. In particular, we will consider using multiple cameras with overlapping fields of view to improve the accuracy of machine learning object recognition. Furthermore, we will study how to apply more recent deep learning techniques to our Raspberry Pi camera system. Meanwhile, we will investigate the impact on the detection accuracy under different environmental conditions, such as lighting and weather. Additionally, we will study a well-balanced incentive mechanism for earning credits from crowdsensing that will dynamically adjust the credit rates in response to the behavior of users.

APPENDICES

A ADDITIONAL EXPERIMENTAL RESULTS

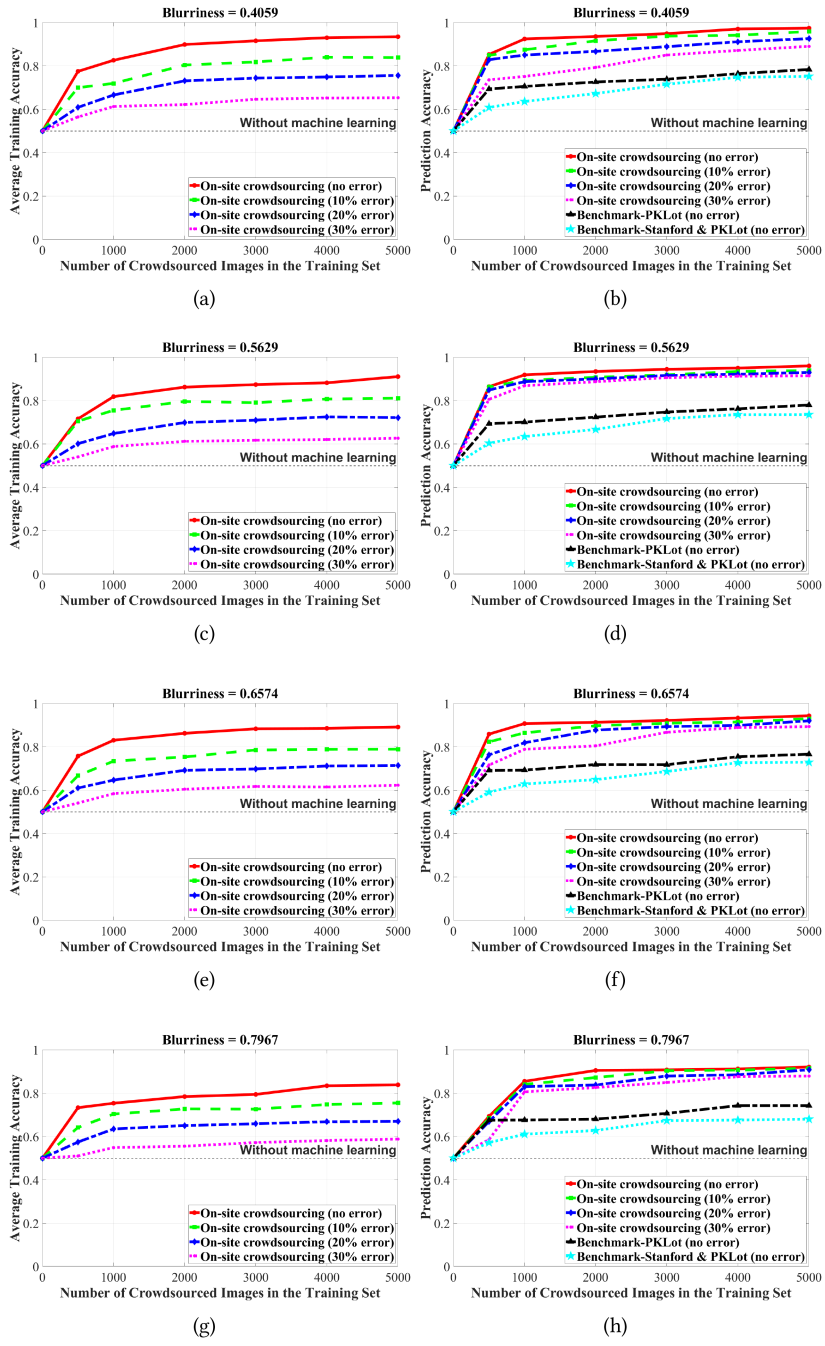


Fig. A.1. Average training accuracy and prediction accuracy vs. number of crowdsourced data at different blurriness levels using different datasets. The average training accuracy is calculated out of 50 epochs.

B SECURITY OF ZERO-KNOWLEDGE PROOFS

We first prove the security of the σ-protocol-based zero-knowledge proofs (i.e., zkpCm, zkpMbs, and zkpNN) by proving their completeness, soundness, and zero-knowledge.

B.1 Completeness Proof

It is straightforward to prove the completeness as one only needs to check if the verifier is convinced if an honest prover correctly executes the protocol.

- (zkpCm): We just need to show whether $g^{z_x} \cdot h^{z_r} \stackrel{?}{=} Cm(x', r') \cdot Cm(x, r)^{\beta}$. Because $z_x = x' + \beta \cdot x$ and $z_r = r' + \beta \cdot r$, it is evident that $g^{x'+\beta x} \cdot h^{r'+\beta r} = g^{x'+\beta x} \cdot h^{r'+\beta r}$.
- (zkpMbs): We need to prove whether $\beta \stackrel{?}{=} \sum_{i=1}^{n} \beta_j$ and $g^{z_{x_j}} \cdot h^{z_{r_j}} \stackrel{?}{=} Cm(x'_j, r'_j) \cdot (\frac{Cm(x,r)}{g^{x_j}})^{\beta_j}$. It is easy to prove the former equation because $\beta_i = \beta - \sum_{j \neq i} \beta_j$. For the latter one, when $j \in \{1, \ldots, n\} \backslash \{i\}$ we can get $g^{x'_j + (x_i - x_j)\beta_j} \cdot h^{r'_j + r \cdot \beta_j} \stackrel{?}{=} g^{x'_j} \cdot h^{r'_j} \cdot g^{(x-x_j)\beta_j} \cdot h^{r \cdot \beta_j}$ and hence $g^{(x_i - x_j)\beta_j} \stackrel{?}{=} g^{(x-x_j)\beta_j}$. The equality is achieved because $x = x_i$. Similarly, when $x = j$ we can get $g^{x'_i - x'_j} \stackrel{?}{=} g^{(x-x_j)\beta_j}$. The equality is achieved because $x = x_i = x_j$.
- (zkpNN): We need to prove whether $h^{z_r} \stackrel{?}{=} Cm(0, r') \cdot Cm(x, r)^{-\beta} \cdot \prod_{i=1}^{m} Cm(b_i, r_i)^{\beta \cdot 2^{i-1}}$. Substituting $z_r = r' + \beta \cdot (\sum_{i=1}^{m} r_i \cdot 2^{i-1} - r)$ into the equation, we can get $h^{r' + \beta \cdot (\sum_{i=1}^{m} r_i \cdot 2^{i-1} - r)} \stackrel{?}{=} h^{r'} \cdot g^{-x\beta} \cdot h^{-r\beta} \cdot g^{\sum_{i=1}^{m} b_i \cdot \beta \cdot 2^{i-1}} \cdot h^{\sum_{i=1}^{m} r_i \cdot \beta \cdot 2^{i-1}}$. Note that the product of the commitments can be changed to a summation due to the homomorphic property. Finally, we can get $1 \stackrel{?}{=} g^{\beta \cdot (\sum_{i=1}^{m} b_i \cdot 2^{i-1} - x)}$, which is easy to verify because $\sum_{i=1}^{m} b_i \cdot 2^{i-1} = x$.

B.2 Soundness Proof

To prove soundness, we require a *Knowledge Extractor* that can extract the knowledge the prover who wants to prove by making the prover successfully answer two random challenges β_1 and β_2, specifically:

- (zkpCm): The extractor sends a random challenge β_1 to the prover, and the prover replies with $z'_x = x' + \beta_1$ and $z'_r = r' + \beta_1$. Then, the extractor rewinds the execution with a new challenge β_2, and the prover replies with $z''_x = x' + \beta_2$ and $z''_r = r' + \beta_2$. Note that because of rewinding, the prover will always send the same x' and r' during each execution. Finally, the extractor is able to extract the prover's secret by computing $x = \frac{z'_x - z''_x}{\beta_1 - \beta_2}$ and $r = \frac{z'_r - z''_r}{\beta_t a_2}$.
- (zkpMbs): The extractor sends two random challenges β_1 and β_2 before and after rewinding, and the prover replies with $z_{r_j} = r'_j + r \cdot \beta_1$ and $z'_{r_j} = r'_j + r \cdot \beta_2$. Next, the extractor is able to extract $r = \frac{z_{r_j} - z'_{r_j}}{\beta_1 - \beta_2}$. By setting $z_r = r$, the simulator can compute $g^{z_{x_j}} \cdot h^r = Cm(x'_j, 0) \cdot \frac{Cm(x,r)}{g^{x_j}}$. Finally, for both $j \in \{1, \ldots, n\} \backslash \{i\}$ and $j = i$, the simulator can get $x = x_i \in \mathcal{X}$.
- (zkpNN): The extractor sends two random challenges β_1 and β_2 before and after rewinding, and the prover replies with $z_r = r' + \beta_1 \cdot (\sum_{i=1}^{m} r_i \cdot 2^{i-1} - r)$ and $z'_r = r' + \beta_2 \cdot (\sum_{i=1}^{m} r_i \cdot 2^{i-1} - r)$, accordingly. Next, the extractor is able to extract $\sum_{i=1}^{m} r_i \cdot 2^{i-1} - r$ by computing $\frac{z_r - z'_r}{\beta_1 - \beta_2}$. By setting $z_r = \sum_{i=1}^{m} r_i \cdot 2^{i-1} - r$, the simulator is able to get $h^{z_r} = Cm(x, r)^{-1} \cdot \prod_{i=1}^{m} Cm(b_i, r_i)^{2^{i-1}}$, and hence $x = \sum_{i=1}^{m} b_i \cdot 2^{i-1} \geqslant 0$.

B.3 Honest Verifier Zero-Knowledge Proof

To prove zero-knowledge, we need to construct a simulator, which interacts with the verifier and can eventually convince the verifier by simulating the transcript of the proof like a prover. However, the simulator does not know the knowledge throughout the interaction.

— (zkpCm): The simulator first sends some initial message to the verifier so that the random challenge β could be obtained. Next, the simulator rewinds the verifier, randomly generates $(x', r') \in \mathbb{Z}_p^2$, and send $Cm(x, r)^{-\beta} \cdot Cm(x', r')$ to the verifier as the initial message. Once the verifier challenges on β again, the simulator replies with $z_x = x'$ and $z_r = r'$.
— (zkpMbs): Similarly, the simulator first randomly sends some initial message to the prover to get the challenge β, and then rewinds the verifier. Next, the simulator randomly picks $x' = x'_i \in \mathcal{X}$, randomly generates $(x'_j, r'_j) \in \mathbb{Z}_p^2$ and β_j, and computes β'_i and z_{r_j}, according to the real prover. In addition, the simulator sets $Cm(x'_j, r'_j) = g^{z_{x_j}} \cdot h^{z_{r_j}} \cdot (\frac{Cm(x,r)}{g^{x_j}})^{-\beta_j}$ for all j. Finally, the simulator sends $(Cm(x'_j, r'_j), z_{x_j})_{j=1}^n$ to the verifier, and once it challenges on β again, the simulator replies with $(\beta_j, z_{r_j})_{j=1}^n$.
— (zkpNN): The simulator randomly send messages in step 1 to get the challenge β, and then rewinds the verifier. Next, the simulator randomly generates m, b_i, r_i, r', r'' and set $z_r = r' + \beta \cdot (\sum_{i=1}^m r_i \cdot 2^{i-1} - r'')$ and $Cm(0, r') = h^{z_r} \cdot Cm(x, r)^\beta \cdot \prod_{i=1}^m Cm(b_i, r_i)^{-\beta \cdot 2^{i-1}}$. Finally, the simulator completes step 1 as the real prover, using the values generated, and when the verifier challenges it on β again, the simulator replies with z_r.

C SECURITY ANALYSIS OF OUR SYSTEM

Our analysis uses the standard technique from [32], which is based on *Ideal/Real-World Simulation*. The rationale is that in an ideal system, there exists a trusted third party P who is able to perform computations and forward the messages between a user and a server. Here, we assume all the communications between users and P are secure, and hence, the ideal model satisfies all the security requirements. On the other hand, in a real-world model, because it is unlikely to deploy P a user communicates with a server through the proposed protocol. Consequently, the security can be defined by comparing if the outputs by all parties in the ideal model are the same as the real-world model. In other words, by observing all the outputs, an adversary cannot obtain more information in the real-world model compared to the ideal-world model.

Next, we define what the ideal-world model looks like, based on each functionality.

— *System Setup*: P keeps maintaining a table, which stores all the registration, balance, and transaction information.
— *User Registration*: When a user wants to join the system, he sends a registration request to P with all the necessary information, and P checks if the request is legitimate. Then, P informs the server that a new user, who is legitimate or not, wants to register, and finally, P forwards the server's decision on whether to accept this request or not back from the server to the user.
— *Crowdsensed Data Submission*: When a user wants to submit the observed availability information for parking space j, he sends all the necessary information to P, and P does all the calculations to validate the user and the data submitted. Finally, P informs the server that an anonymous user, who is eligible or not, wants to submit crowdsensed data and then forwards the server's decision back to the user.
— *Credit Claiming*: When a user wants to claim the credit for submitting the crowdsensed data for parking space j, he sends all the necessary information to P, and P does all the calculations to validate the user and the data submitted. Finally, P informs the server that an anonymous user, who is eligible or not, wants to claim the credit and then forwards the server's decision back to the user.
— *Result Inquiry*: When a user wants to inquiry about the availability information for parking space j, he sends all the necessary information to P, and P does all the calculations to validate

the user and the data submitted. Finally, P informs the server that an anonymous user, who is eligible or not, wants to send a parking inquiry and then forwards the server's decision back to the user.

The above ideal system satisfies all the properties listed in Section 5.1 because when the server interacts with **P**, (1) it does not know who initiated the request (the identity of the user), and (2) the legitimacy of the request is informed by **P** so that it can always decide to proceed the transaction when the request is legitimate and vice versa. Next, we formally define security for our system.

Theorem C.1. *Denote the ideal-world functionality as $\mathcal{F}$ and the proposed protocol as* Π. Π *is secure if for every static probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ in the real-world model, there is also a PPT $\mathcal{S}$ in the ideal-world model, who has black-box access to $\mathcal{A}$, such that:*

$$\boldsymbol{REAL}_{\mathcal{S},\Pi} \overset{stat}{\equiv} \boldsymbol{IDEAL}_{\mathcal{A},\mathcal{F}}, \tag{4}$$

where ***REAL*** *and* ***IDEAL*** *are the outputs of the real-world model and ideal-world model, respectively.*

In other words, informally speaking, a real-world model is secure if one cannot distinguish the difference between it and the ideal-world model since the ideal-world model satisfies all the predefined security requirements. We prove our protocol based on the theorem above. Our objective is to construct a simulator $\mathcal{S}$ for every possible adversary $\mathcal{A}$ so that (1) the outputs of both models are statistically indistinguishable, and (2) $\mathcal{A}$ cannot distinguish whether he is interacting with the honest party or honest party controlled by $\mathcal{S}$. We divide the proof into two parts. The first part covers the scenario where $\mathcal{A}$ corrupts and controls some users only, and the second part covers the scenario in which some users and the server are corrupted and controlled by $\mathcal{A}$. The simulator is constructed as follows:

Scenario 1: $\mathcal{A}$ corrupts some users only.

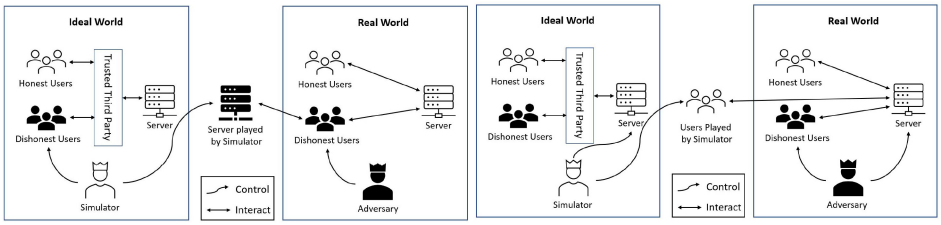
- *System Setup*: The simulator generates all the cryptographic parameters and sends them to the dishonest users controlled by $\mathcal{A}$.
- *User Registration*: In the real-world model, when a new dishonest user would like to initiate a registration request, the emulated server interacts with it and extracts s and q from the credential in order to identify the corrupted user later in the ideal-world model. Then, in the ideal-world model, $\mathcal{S}$ corrupts the same user to complete the registration with P.
- *Crowdsensed Data Submission:* In the real-world model, when a dishonest user launches a new submission request for parking space j, $\mathcal{S}$ extracts s from it and initiates a data submission request to P for the same user with secret key s in the ideal-world model. If P accepts the request, $\mathcal{S}$ continues.
- *Credit Claiming*: Similarly, in the real-world model, when a dishonest user launches a credit claiming request, $\mathcal{S}$ extracts s from it and initiates a credit claiming request to P on behalf of the same user with secret key s in the ideal-world model. If P accepts the request, $\mathcal{S}$ continues.
- *Result Inquiry*: Likewise, in the real-world model, when a dishonest user launches a parking inquiry request for parking space j, $\mathcal{S}$ extracts s from it and initiates a parking inquiry request to P on behalf of the same user with secret key s in the ideal-world model. If P accepts the request, $\mathcal{S}$ continues.

We remark that the outputs in both models are identical and $\mathcal{A}$ cannot distinguish between if it is interacting with the real server or the emulated one in this scenario because of the following:

- For honest users, as no party controls them, the outputs should always be the same in both models.

— For dishonest users, as the simulator can always obtain the dishonest users' output in the real-world model, he can easily identify the dishonest users and make the same request to P on behalf of the same users in the ideal-world model.
— For the server, as the outputs of the honest users and the dishonest users are identical in both models, the outputs and views of the server should be the same as well.

An illustrated is shown in Figure C.1(a).



(a) Scenario 1 (b) Scenario 2

Fig. C.1. (a) Scenario 1: $\mathcal{A}$ corrupts and controls some users only. (b) Scenario 2: $\mathcal{A}$ corrupts and controls some users and the server.

Scenario 2: $\mathcal{A}$ corrupts some users and the server.

— *System Setup*: In the real-world model, the users (controlled by $\mathcal{S}$) receive the cryptographic parameters generated by the server (controlled by $\mathcal{A}$).
— *User Registration*: In the ideal-world model, if the server (controlled by $\mathcal{S}$) receives a new user registration request from P for a user, a user played by $\mathcal{S}$ invokes the server in the real-world model to complete the registration protocol, and if it receives s'' and $sign_Q$ from the server, the server in the ideal-world model accepts the registration request.
— *Crowdsensed Data Submission*: In the ideal-world model, if the server receives a data submission request from P, and P indicates that an anonymous and legitimate user would like to launch a new request for parking space j, $\mathcal{S}$ controls a user to initiate the same protocol in the real-world model with the real-world server. If the server accepts the request, $\mathcal{S}$ invokes the ideal-world server to accept the request as well.
— *Credit Claiming*: Similarly, in the ideal-world model, if the server receives an anonymous and legitimate credit claiming request from P, $\mathcal{S}$ controls a user to initiate the same protocol in the real-world model with the real-world server. If the server accepts the request, $\mathcal{S}$ invokes the ideal-world server to accept the request as well.
— *Result Inquiry*: Likewise, in the ideal-world model, if the server receives an anonymous and legitimate parking inquiry request from P, $\mathcal{S}$ controls a user to initiate the same protocol in the real-world model with the real-world server. If the server accepts the request, $\mathcal{S}$ invokes the ideal-world server to accept the request as well.

We remark that this scenario also preserves the equivalence and indistinguishability because of the following:

— For all users, as the security parameters received from the real-world server are the same, the outputs should always follow the same distributions in both models.
— For the server, as $\mathcal{S}$ knows the decision made by the real-world server, it can always return the same decision to P.

An illustration is provided in Figure C.1(b). Therefore, our proposed protocol is secure based on the security definition.

ACKNOWLEDGMENTS

We are grateful to the helpful comments and suggestions offered by the associate editor and anonymous reviewers.

REFERENCES

[1] Muhammad Aftab, Sid Chi-Kin Chau, and Prashant Shenoy. 2020. Efficient online classification and tracking on resource-constrained IoT devices. *ACM Transactions on Internet of Things* 1, 3 (Aug. 2020), Article 20, 1–29.

[2] Muhammad Aftab, Chien Chen, Chi-Kin Chau, and Talal Rahwan. 2017. Automatic HVAC control with real-time occupancy recognition and simulation-guided model predictive control in low-cost embedded system. *Energy and Buildings* 154 (2017), 141–156.

[3] H. Al-Absi, J. Devaraj, P. Sebastian, and Yap Vooi Voon. 2010. Vision-based automated parking system. In *Proceedings of the International Conference on Information Science, Signal Processing and their Applications*. 757–760.

[4] G. Alessandretti, A. Broggi, and P. Cerri. 2007. Vehicle and guard rail detection using radar and vision data fusion. *IEEE Transactions on Intelligent Transportation Systems* 8, 1 (2007), 95–105.

[5] Nada Alhirabi, Omer Rana, and Charith Perera. 2021. Security and privacy requirements for the internet of things: A survey. *ACM Transactions on Internet of Things* 2, 1 (Feb. 2021), Article 6, 37 pages.

[6] Paulo Almeida, Luiz S. Oliveira, Eunelson Silva, Alceu Britto, and Alessandro Koerich. 2013. Parking space detection using textural descriptors. In *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics*. 3603–3608.

[7] G. Amato, P. Bolettieri, D. Moroni, F. Carrara, L. Ciampi, G. Pieri, C. Gennaro, G. R. Leone, and C. Vairo. 2018. A wireless smart camera network for parking monitoring. In *Proceedings of the 2018 IEEE Globecom Workshops*. 1–6.

[8] S. Banerjee, P. Choudekar, and M. K. Muju. 2011. Real time car parking system using image processing. In *Proceedings of the 2011 3rd International Conference on Electronics Computer Technology*, Vol. 2. 99–103.

[9] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. 2012. Multiparty computation secure against continual memory leakage. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*. 1235–1254.

[10] William J. Buchanan. 2017. *Cryptography*. River Publishers.

[11] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. 2018. Bulletproofs: Short proofs for confidential transactions and more. In *Proceedings of the 2018 IEEE Symposium on Security and Privacy*. 315–334.

[12] Andrea Capponi, Claudio Fiandrino, Burak Kantarci, Luca Foschini, Dzmitry Kliazovich, and Pascal Bouvry. 2019. A survey on mobile crowdsensing systems: Challenges, solutions and opportunities. *IEEE Communications Survey and Tutorials* 21, 3 (2019), 2419–2465.

[13] Chi-Kin Chau, Khaled Elbassioni, and Chien-Ming Tseng. 2017. Drive mode optimization and path planning for plug-in hybrid electric vehicles. *IEEE Transactions on Intelligent Transportation Systems* 18, 12 (2017), 3421–3432.

[14] Jianwei Chen, Huadong Ma, David S. L. Wei, and Dong Zhao. 2015. Participant-density-aware privacy-preserving aggregate statistics for mobile crowd-sensing. In *Proceedings of the 2015 IEEE 21st International Conference on Parallel and Distributed Systems*. 140–147.

[15] Jacob Eberhardt and Stefan Tai. 2018. ZoKrates - scalable privacy-preserving off-chain computations. In *Proceedings of the 2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data*. 1084–1091.

[16] A. S. El-Wakeel, J. Li, A. Noureldin, H. S. Hassanein, and N. Zorba. 2018. Towards a practical crowdsensing system for road surface conditions monitoring. *IEEE Internet of Things Journal* 5, 6 (2018), 4672–4685.

[17] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. 2014. De-anonymization attack on geolocated data. *Journal of Computer and System Sciences* 80, 8 (2014), 1597–1614.

[18] Gurchetan S. Grewal, Mark D. Ryan, Liqun Chen, and Michael R. Clarkson. 2015. Du-vote: Remote electronic voting with untrusted computers. In *Proceedings of the 2015 IEEE 28th Computer Security Foundations Symposium*. 155–169.

[19] Omid Hajihassani, Omid Ardakanian, and Hamzeh Khazaei. 2021. Anonymizing sensor data on the edge: A representation learning and transformation approach. *ACM Transactions on Internet of Things* 3, 1(Oct 2021), Article 8 , 26 pages.

[20] Ching-Chun Huang and Hoang Tran Vu. 2017. A deep learning network for vision-based vacant parking space detection system. In *Proceedings of the 2017 IEEE International Conference on Image Processing*. 4586–4586.

[21] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. 2007. Zero-knowledge from secure multiparty computation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*. 21–30.

[22] Linshan Jiang, Rui Tan, Xin Lou, and Guosheng Lin. 2021. On lightweight privacy-preserving collaborative learning for internet of things by independent random projections. *ACM Transactions on Internet of Things* 2, 2 (March 2021), Article 11, 32 pages.

[23] Haiming Jin, Lu Su, Houping Xiao, and Klara Nahrstedt. 2018. Incentive mechanism for privacy-aware data aggregation in mobile crowd sensing systems. *IEEE/ACM Transactions on Networking* 26, 5 (2018), 2019–2032.
[24] R. Kanan and H. Arbess. 2020. An IoT-based intelligent system for real-time parking monitoring and automatic billing. In *Proceedings of the 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies*. 622–626.
[25] Muhammad Khalid, Kezhi Wang, Nauman Aslam, Yue Cao, Naveed Ahmad, and Muhammad Khurram Khan. 2021. From smart parking towards autonomous valet parking: A survey, challenges and future works. *Journal of Network and Computer Applications* 175, 1 (2021), 102935.
[26] Youssef Khazbak, Junpeng Qiu, Tianxiang Tan, and Guohong Cao. 2020. TargetFinder: A privacy preserving system for locating targets through IoT cameras. *ACM Transactions on Internet of Things* 1, 3 (June 2020), Article 14, 23 pages.
[27] H. Kim, J. W. Kim, and B. Jang. 2019. Indoor positioning system using sensor and crowdsourcing landmark map update. In *Proceedings of the 2019 International Conference on Green and Human Information Technology*. 7–11.
[28] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. 2016. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *Proceedings of the 2016 IEEE Symposium on Security and Privacy*. 839–858.
[29] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3D object representations for fine-grained categorization. In *Proceedings of the 2013 IEEE International Conference on Computer Vision Workshops*. 554–561.
[30] J. Krumm. 2007. Inference attacks on location tracks. In *Proceedings of the International Conference on Pervasive Computing*.
[31] S. Lin, Y. Chen, and S. Liu. 2006. A vision-based parking Lot management system. In *Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics*, Vol. 4. 2897–2902.
[32] Joseph K. Liu, Willy Susilo, Tsz Hon Yuen, Man Ho Au, Junbin Fang, Zoe L. Jiang, and Jianying Zhou. 2016. Efficient privacy-preserving charging station reservation system for electric vehicles. *Computer Journal* 59, 7 (2016), 1040–1053.
[33] E. Mahmoud, S. Fawaz, A. Asim, and B. Haitham. 2018. A new method for full reference image blur measure. *International Journal of Simulation: Systems, Science and Technology* 19, 4 (May 2018), 1–5.
[34] Arvind Narayanan and Vitaly Shmatikov. 2008. Robust De-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*. 111–125.
[35] A. Natarajan, K. Bharat, G. R. Kaustubh, S. P. P. N., M. Moharir, N. K. Srinath, and K. N. Subramanya. 2019. An approach to real time parking management using computer vision. In *Proceedings of the 2nd International Conference on Control and Computer Vision*. 18–22.
[36] P. N. Pathirana, A. E. K. Lim, A. V. Savkin, and P. D. Hodgson. 2007. Robust video/ultrasonic fusion-based estimation for automotive applications. *IEEE Transactions on Vehicular Technology* 56, 4 (2007), 1631–1639.
[37] Francesco Pittaluga and Sanjeev J. Koppal. 2015. Privacy preserving optics for miniature vision sensors. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*. 314–324.
[38] Michael O. Rabin, Rocco A. Servedio, and Christopher Thorpe. 2007. Highly efficient secrecy-preserving proofs of correctness of computations and applications. In *Proceedings of the 22nd Annual IEEE Symposium on Logic in Computer Science*. 63–76.
[39] Sanjay Saini and Joydip Dhar. 2008. An eavesdropping proof secure online voting model. In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, Vol. 3. 704–708.
[40] F. Shi, D. Wu, D. I. Arkhipov, Q. Liu, A. C. Regan, and J. A. McCann. 2019. ParkCrowd: Reliable crowdsensing for aggregation and dissemination of parking space information. *IEEE Transactions on Intelligent Transportation Systems* 20, 11 (2019), 4032–4044.
[41] Gang Sun, Siyu Sun, Hongfang Yu, and Mohsen Guizani. 2020. Toward incentivizing fog-based privacy-preserving mobile crowdsensing in the internet of vehicles. *IEEE Internet of Things Journal* 7, 5 (2020), 4128–4142.
[42] Jiajun Sun and Huadong Ma. 2014. Privacy-preserving verifiable incentive mechanism for online crowdsourcing markets. In *Proceedings of the 2014 23rd International Conference on Computer Communication and Networks*. 1–8.
[43] Chien-Ming Tseng and Chi-Kin Chau. 2017. Personalized prediction of vehicle energy consumption based on participatory sensing. *IEEE Transactions on Intelligent Transportation Systems* 18, 11 (2017), 3103–3113.
[44] P. Tzamalis, P. Vikatos, and S. Nikoletseas. 2019. A hybridization of mobile crowdsensing, Twitter analytics, and sensor data for the holistic approach of Pollen onsets detection. In *Proceedings of the 2019 15th International Conference on Distributed Computing in Sensor Systems*. 188–191.
[45] Shaowei Wang, Liusheng Huang, Yiwen Nie, Xinyuan Zhang, Pengzhan Wang, Hongli Xu, and Wei Yang. 2019. Local differential private data aggregation for discrete distribution estimation. *IEEE Transactions on Parallel and Distributed Systems* 30, 9 (2019), 2046–2059.
[46] Xiong Wang, Zhe Liu, Xiaohua Tian, Xiaoying Gan, Yunfeng Guan, and Xinbing Wang. 2017. Incentivizing crowdsensing with location-privacy preserving. *IEEE Transactions on Wireless Communications* 16, 10 (2017), 6940–6952.
[47] Maoqiang Wu, Dongdong Ye, Jiawen Kang, and Rong Yu. 2017. Collaborative data collection with hybrid vehicular crowd sensing in smart cities. In *Proceedings of the 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. 1–6.

[48] Yao Wu, Yuncheng Wu, Hui Peng, Hong Chen, and Cuiping Li. 2016. MagiCrowd: A crowd based incentive for location-aware crowd sensing. In *Proceedings of the 2016 IEEE Wireless Communications and Networking Conference*. 1–6.
[49] Qichao Xu, Zhou Su, Minghui Dai, and Shui Yu. 2020. APIS: Privacy-preserving incentive for sensing task allocation in cloud and edge-cooperation mobile internet of things with SDN. *IEEE Internet of Things Journal* 7, 7 (2020), 5892–5905.
[50] Yuan Zhang, He Zhang, Siyuan Tang, and Sheng Zhong. 2016. Designing secure and dependable mobile sensing mechanisms with revenue guarantees. *IEEE Transactions on Information Forensics and Security* 11, 1 (2016), 100–113.
[51] Bowen Zhao, Shaohua Tang, Ximeng Liu, and Xinglin Zhang. 2021. PACE: Privacy-preserving and quality-aware incentive mechanism for mobile crowdsensing. *IEEE Transactions on Mobile Computing* 20, 5 (2021), 1924–1939.
[52] Hanwei Zhu and Sid Chi-Kin Chau. 2021. Integrating IoT-sensing and crowdsensing for privacy-preserving parking monitoring. In *Proceedings of the 8th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 226–227.
[53] Hanwei Zhu, Songzeng Fan, Xiyu Wang, and Sid Chi-Kin Chau. 2020. Privacy-preserving camera-based monitoring and tracking system for parking spaces. In *Proceedings of the ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 346–347.

Received October 2021; revised June 2022; accepted June 2022