Privacy-Preserving Truth Discovery in Crowd Sensing Systems

CHENGLIN MIAO, WENJUN JIANG, and LU SU, State University of New York at Buffalo
YALIANG LI, Tencent Medical AI Lab
SUXIN GUO, State University of New York at Buffalo
ZHAN QIN, The University of Texas at San Antonio
HOUPING XIAO, Georgia State University
JING GAO and KUI REN, State University of New York at Buffalo

The recent proliferation of human-carried mobile devices has given rise to the crowd sensing systems. However, the sensory data provided by individual participants are usually not reliable. To better utilize such sensory data, the topic of truth discovery, whose goal is to estimate user quality and infer reliable aggregated results through quality-aware data aggregation, has drawn significant attention. Though able to improve aggregation accuracy, existing truth discovery approaches fail to address the privacy concerns of individual users. In this article, we propose a novel privacy-preserving truth discovery (PPTD) framework, which can protect not only users' sensory data but also their reliability scores derived by the truth discovery approaches. The key idea of the proposed framework is to perform weighted aggregation on users' encrypted data using a homomorphic cryptosystem, which can guarantee both high accuracy and strong privacy protection. In order to deal with large-scale data, we also propose to parallelize PPTD with MapReduce framework. Additionally, we design an incremental PPTD scheme for the scenarios where the sensory data are collected in a streaming manner. Extensive experiments based on two real-world crowd sensing systems demonstrate that the proposed framework can generate accurate aggregated results while protecting users' private information.

CCS Concepts: • **Information systems** → **Information systems applications**; • **Security and privacy** → **Privacy protections**;

Additional Key Words and Phrases: Crowd sensing, truth discovery, privacy-preserving

This work was sponsored in part by US National Science Foundation under grant IIS-1319973, IIS-1553411, CNS-1262277, CNS-1566374, CNS-1652503, and CNS-1742845.
A preliminary work has been presented in ACM SenSys 2015 [41].
Authors' addresses: C. Miao, W. Jiang, L. Su, S. Guo, J. Gao, and K. Ren, Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260; emails: {cmiao, wenjunji, lusu}@buffalo.edu, suxinguo@gmail.com, {jing, kuiren}@buffalo.edu; Y. Li, Tencent Medical AI Lab, Palo Alto, CA 94301; email: yaliangli@tencent.com; Z. Qin, Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249; email: zhan.qin@utsa.edu; H. Xiao, J. Mack Robinson College of Business, Georgia State University, Atlanta, GA 30303; email: hxiao@gsu.edu.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Association for Computing Machinery.
1550-4859/2019/01-ART9 $15.00
https://doi.org/10.1145/3277505

ACM Reference format:
Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin, Houping Xiao, Jing Gao, and Kui Ren. 2019. Privacy-Preserving Truth Discovery in Crowd Sensing Systems. *ACM Trans. Sen. Netw.* 15, 1, Article 9 (January 2019), 32 pages.
https://doi.org/10.1145/3277505

1 INTRODUCTION

The recent proliferation of increasingly capable and affordable mobile devices (e.g., smartphones, smartwatches, smartglasses) packed with a plethora of on-board sensors (e.g., GPS, accelerometer, compass, camera) has given rise to crowd sensing, a newly emerged sensing paradigm where the collection of sensory data is outsourced to a crowd of users participating in the sensing task. Recently, a large variety of crowd sensing systems [3–6, 12, 19–22, 31, 46–48, 52] have been developed, serving a wide spectrum of applications that have significant impact on our daily lives, including urban sensing, smart transportation, environment monitoring, localization, healthcare, public opinion analysis, and many others.

However, in crowd sensing applications, the sensory data provided by individual participants are usually not reliable, due to various reasons such as poor sensor quality, lack of sensor calibration, background noise, incomplete views of observations, and even the intent to deceive. Therefore, the power of crowd sensing can be unleashed only by properly aggregating unreliable information from different participating users who inevitably submit noisy, conflicting, and heterogeneous data. When aggregating crowd sensing data, it is essential to capture the difference in the quality of information among different participating users. Some users constantly provide truthful and meaningful data while others may generate biased or even fake data. In this case, traditional aggregation methods (e.g., averaging and voting) that regard all the users equally would not be able to derive accurate aggregated results.

Therefore, an ideal approach should be able to involve the probability of a user providing accurate data in the form of user weight when aggregating sensory data, and make the aggregated results close to the information provided by reliable users. The challenge here, however, is that the user reliability is usually unknown *a priori* and should be inferred from collected data. To address this challenge, the problem of *truth discovery* [25, 32–37, 39, 40, 51, 56–59], i.e., to discover truthful facts from unreliable user information, has recently been widely studied. The common principle shared in truth discovery approaches is that a particular user will have higher weight if the data provided by him is closer to the aggregated results, and a particular user's data will be counted more in the aggregation procedure if this user has a higher weight. A variety of truth discovery approaches have been proposed to calculate user weight and aggregated results in a joint manner based on this principle.

The truth discovery approaches, though having brought significant improvement to the aggregation accuracy, fail to take into consideration an important practical issue in the design of crowd-sensing systems, i.e., *the protection of user privacy*. In many crowd sensing applications, the final aggregation results can be public and beneficial to the community or society, but the data from each individual user may contain private personal information and, thus, should be well protected. For example, aggregating health data, such as treatment outcomes, can lead to better evaluation of new drugs or medical devices' effects but may jeopardize the privacy of participating patients. The geotagging campaigns can provide accurate and timely localization of specific objects (e.g., litter, pothole, automated external defibrillator) by aggregating the reports of participants, however, at the risk of leaking participants' sensitive location information. Through crowd wisdom, even extremely difficult questions can be solved via aggregating the answers of a large crowd. However, personal information of individual users can be inferred from their answers.

Sometimes, user reliability is another sensitive information that should also be protected. On one hand, from user reliability information, together with his observation values, the attacker may be able to infer the personal information of the user, such as major, education level, age, gender, language, and even personality. On the other hand, in practical crowd sensing applications, the participating users usually trade their data with the system administrator for rewards, and the leakage of user reliability may lead to malicious manipulation of data price. For these reasons, in some crowd sensing applications (such as the aforementioned health data aggregation, geotagging, and crowd wisdom), user reliability should be kept private.

Therefore, it is essential to design a privacy-preserving truth discovery scheme for the crowd-sensing applications where there exists variability in user reliability degrees and the privacy of users' data and reliability information is susceptible to leakage. Toward this end, we propose a novel privacy-preserving truth discovery (PPTD) framework. This framework makes use of a homomorphic cryptosystem [8] and can guarantee both high accuracy and strong privacy. The proposed PPTD framework works as follows. Each participating user will first send the encrypted summation of distances between his own observation values and the estimated aggregated values to the cloud server. Then, the cloud server updates users' weights in encrypted form without decrypting the received distances information and sends the updated weight to each user. Next, each user calculates the ciphertexts of weighted data using the received encrypted weight. Finally, the final results are estimated by the cloud server based on the ciphertexts received from users. The advantage of our proposed framework is that it can accurately calculate the final aggregated results while protecting the privacy of user data, and at the same time, the weight information is not disclosed to any party.

Additionally, in order to deal with massive data, we design a parallel PPTD scheme using the MapReduce framework [10], and, thus, the PPTD procedure can be conducted in a parallel and distributed manner. As for the scenarios where the sensing data are collected in a streaming manner, we also propose an incremental PPTD scheme that can aggregate the sensing data in real time and introduce less computational overhead compared with the basic PPTD framework.

In summary, our contributions in this article are:

(1) We propose a novel PPTD framework for crowd sensing systems, which can accurately aggregate sensory data while protecting both user observation and user reliability from being disclosed.
(2) A parallel extension of PPTD is also designed so that the truth discovery procedure can be conducted in parallel when processing large-scale crowd sensing data.
(3) In order to deal with the scenarios where the sensing data are collected in a streaming manner, we extend PPTD and propose an incremental PPTD scheme so that the sensing data can be aggregated more efficiently.
(4) We conduct extensive experiments on both real crowd sensing systems and a synthetic dataset. The results validate the claim that our proposed schemes can generate accurate aggregated results while protecting the privacy of user data and weight.

In the remaining parts of this article, we first define the problem in Section 2 and give preliminary in Section 3. Then, the details of the proposed PPTD framework are provided in Section 4. In Sections 5 and 6, we present the parallel scheme and incremental scheme, respectively. We analyze the privacy of our proposed framework and discuss it in Sections 7 and 8, respectively. In Section 9, we conduct a series of experiments to demonstrate the claims given in this article. We discuss related work in Section 10 and conclude the article in Section 11.

2 PROBLEM DEFINITION

In this section, we describe the problem settings of our proposed PPTD framework. Our framework contains two different types of crowd sensing parties: *cloud server* and *users*. Among them, users are the crowd participants who perform sensing tasks with their mobile devices either voluntarily or for financial incentives, and cloud server is a platform that collects user data and conducts data aggregation. Additionally, we use *objects* to represent the entities or questions assigned by the cloud server and use *observation values* to denote the sensory readings or answers provided by crowd users. Also, the true result or answer for each task or question is represented as *ground truth* in our problem.

In practical crowd sensing systems, the security threats mainly come from the parties themselves (i.e., cloud server and users). For the sake of curiosity or financial purpose, the cloud server may try to deduce the observation and reliability values of each user. On the other hand, each user may also try to infer the information of other parties. Thus, it is of paramount importance to preserve the privacy of users' observation values. Moreover, in order to prevent any party from maliciously manipulating the data price in the scenarios where crowd users trade their data with the cloud server, we propose to protect the reliability value of each user from being disclosed to any party (including the user himself). Certainly, our proposed framework can be easily modified to make each user's weight known only to himself, which is discussed in detail in Section 8. In this article, we assume that all the parties are semi-honest [38], which means all the parties strictly follow the protocol we design, but each party will try to infer the private information of other parties based on the intermediate results he obtains during the execution of the protocol. Additionally, we assume that the parties in our framework have no collusions, which means they will not collude with each other outside the designed protocol. These assumptions are reasonable in most crowd sensing scenarios, since (1) the parties want to get correct results and, thus, would follow the protocol for their mutual benefits, and (2) crowd users usually do not know each other, and even if they know each other, they are probably not willing to disclose private information to others.

We formally define the problem targeted in this article as follows:

Suppose there are K users, denoted as $\mathcal{K} = \{1, 2, \ldots, K\}$, and a cloud server $\mathcal{S}$ that released M objects represented as $\mathcal{M} = \{1, 2, \ldots, M\}$. Let x_m^k denote the observation value provided by the k-th user for the m-th object and w_k denote the weight of the k-th user. For each object, there is a ground truth that is not known by all the parties in the framework. Our goal is to let server $\mathcal{S}$ accurately calculate the estimated values $\{x_m^*\}_{m=1}^M$ of the ground truths for all the objects based on the information collected from users. In this procedure, each observation value (i.e., x_m^k) should not be disclosed to any party except the user who provides this value (i.e., the k-th user). Also, the weight information $\{w_k\}_{k=1}^K$ should not be disclosed to any party in the system.

To solve this problem, we propose a PPTD framework based on homomorphic cryptosystem, which enables the cloud server to conduct truth discovery on encrypted sensing data so that the private information could be effectively protected while the ground truths can be accurately estimated.

3 PRELIMINARY

Since truth discovery and homomorphic encryption technology are two important components in our proposed framework, we introduce the concepts and general procedures of them in this section.

3.1 Truth Discovery

Toward the goal of resolving conflicts in multiple noisy data sources, truth discovery has been widely studied in various domains. Although there are differences in the ways to compute user

weights and estimate ground truths, the common procedure of existing truth discovery approaches can be summarized as follows. A truth discovery algorithm usually starts with a random guess of ground truths, and then iteratively conducts weight update and truth update until convergence.

Weight Update: In this step, we assume the estimated ground truth of each object is fixed. The basic idea is that a user's weight should be assigned a high value if this user provides data that is close to the estimated ground truths. Typically, the user weights are calculated as follows:

$$w_k = f\left(\sum_{m=1}^{M} d\left(x_m^k, x_m^*\right)\right), \tag{1}$$

where f is a monotonically decreasing function, and $d(\cdot)$ is the distance function that can measure the difference between users' observation values and the estimated ground truths. In this article, we adopt the weight calculation function of CRH [33, 37] as f due to its good practical performance:

$$w_k = \log\left(\frac{\sum_{k'=1}^{K}\sum_{m=1}^{M} d(x_m^{k'}, x_m^*)}{\sum_{m=1}^{M} d(x_m^k, x_m^*)}\right) \tag{2}$$

The distance function $d(\cdot)$ will be chosen based on the application scenarios. The proposed framework can handle various applications by plugging different functions. In this article, we discuss two example functions for applications involving continuous or categorical data, the two most common data types in crowd sensing applications.

For the applications (e.g., environment monitoring) where the sensory data are continuous (e.g., temperature and humidity), we adopt the following normalized squared distance function:

$$d\left(x_m^k, x_m^*\right) = \frac{(x_m^k - x_m^*)^2}{std_m}, \tag{3}$$

where std_m is the standard deviation of all observation values for object m. For the applications (e.g., crowd wisdom) where the data are categorical (e.g., multiple-choice answer), there are usually multiple candidate choices, and only one of them is correct. In this case, we define an observation vector $x_m^k = (0, \ldots, \underset{q}{1}, \ldots, 0)^T$ to denote that user k selects the q-th choice for object m. We then use the squared distance function to measure the difference between observation vector x_m^k and the estimated ground truth vector x_m^*:

$$d\left(x_m^k, x_m^*\right) = \left(x_m^k - x_m^*\right)^T \left(x_m^k - x_m^*\right) \tag{4}$$

Truth Update: In this step, we assume the weight of each user is fixed. Then, we can estimate the ground truth for the m-th object as

$$x_m^* \leftarrow \frac{\sum_{k=1}^{K} w_k \cdot x_m^k}{\sum_{k=1}^{K} w_k} \tag{5}$$

For continuous data, x_m^* represents the estimated ground truth value. But for categorical data, x_m^* is actually a probability vector in which each element represents the probability of a particular choice being the truth. The final estimation should be the choice with the largest probability in vector x_m^*.

The general truth discovery procedure can be described by Algorithm 1. The algorithm starts with randomly guessing ground truth for each object, then iteratively updates users' weights and estimated ground truths until some convergence criterion is satisfied. Usually, the convergence criterion is set depending on the requirements of specific applications. For example, it can be a threshold of the change in the estimated ground truths in two consecutive iterations.

ALGORITHM 1: Truth Discovery Algorithm

Input: Observation values from K users: $\{x_m^k\}_{m,k=1}^{M,K}$

Output: Estimated ground truths for M objects: $\{x_m^*\}_{m=1}^{M}$

```
Randomly initialize the ground truth for each object;
repeat
    for each user k do
        Update weight based on estimated ground truths (e.g., Equation (2));
    end
    for each object m do
        Update the estimated ground truth based on current weights (e.g., Equation (5));
    end
until Convergence criterion is satisfied;
return The estimated ground truths {x*_m}_{m=1}^M;
```

3.2 Cryptographic Tools

3.2.1 Homomorphic Cryptographic Scheme. In our proposed PPTD framework, an additive homomorphic asymmetric cryptosystem is adopted. As it is widely known, there are two types of keys in the asymmetric cryptosystem: public key pk and private key sk. The public key is used to encrypt plaintext and the private key is used to decrypt the ciphertext. Considering a plaintext $m \in \mathbb{Z}_n$, where n is a large positive integer and $\mathbb{Z}_n$ is the set of integers modulo n, we denote the encryption of m as $E_{pk}(m)$. If a cryptographic scheme is said to be additive homomorphic, there should be two operators $\oplus$ and $\otimes$ that satisfy the following properties:

$$E_{pk}(m_1 + m_2) = E_{pk}(m_1) \oplus E_{pk}(m_2) \tag{6}$$

$$E_{pk}(a \cdot m_1) = a \otimes E_{pk}(m_1), \tag{7}$$

where m_1, m_2 are the plaintexts that need to be encrypted and a is a constant.

Based on the above properties, we can directly calculate the encrypted sum of plaintexts from the encryptions of them by conducting operators $\oplus$ or $\otimes$.

3.2.2 Threshold Paillier Cryptosystem. Although there are several additive homomorphic cryptographic schemes, we use the threshold variant of the Paillier scheme [9] in our framework, because it not only has additive homomorphic properties but also satisfies the design of a threshold cryptosystem, both of which allow us to conduct secure summation on the data collected from crowd users.

In this cryptosystem, a user can encrypt the plaintext $m \in \mathbb{Z}_n$ with the public key $pk = (g, n)$ as

$$c = E_{pk}(m) = g^m r^n \, mod \, n^2, \tag{8}$$

where $r \in \mathbb{Z}_n^*$ ($\mathbb{Z}_n^*$ denotes the multiplicative group of invertible elements of $\mathbb{Z}_n$) is selected randomly and privately by this user. According to Equations (6), (7), and (8), the homomorphic properties of this cryptosystem can be described as

$$\begin{aligned} E_{pk}(m_1 + m_2) &= E_{pk}(m_1) \cdot E_{pk}(m_2) \\ &= g^{m_1+m_2}(r_1 r_2)^n \, mod \, n^2 \end{aligned} \tag{9}$$

$$E_{pk}(a \cdot m_1) = E_{pk}(m_1)^a = g^{a m_1} {r_1}^{a n} \, mod \, n^2, \tag{10}$$

where m_1, m_2 are the plaintexts that need to be encrypted, and $r_1, r_2 \in \mathbb{Z}_n^*$ are the private randoms, and a is a constant.

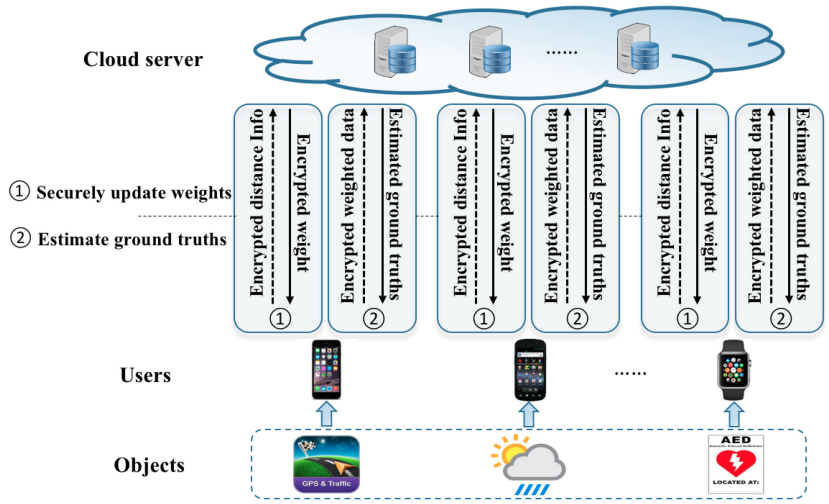


Fig. 1. Privacy-preserving truth discovery framework.

In this article, the (p, t)-threshold Paillier cryptosystem is adopted, in which the private key sk is divided (denoted as $sk_1, sk_2, \ldots, sk_p$) and distributed to p parties. Any single party doesn't have the complete private key. If one party wants to accurately decrypt ciphertext c, it has to cooperate with at least $t - 1$ other parties. So in the decryption step, each party $i (1 \leq i \leq p)$ needs to calculate the partial decryption c_i of c with private key sk_i as

$$c_i = c^{2\Delta sk_i}, \tag{11}$$

where $\Delta = p!$. Then, based on the combining algorithm in Ref. [9], at least t partial decryptions can be combined together to get the plaintext m.

4 PRIVACY-PRESERVING TRUTH DISCOVERY

In this section, we discuss the details of our novel PPTD framework.

4.1 PPTD Overview

Figure 1 shows the framework of PPTD in crowd sensing systems. Before the truth discovery procedure, we assume a semantically secure (p, t)-threshold Paillier cryptosystem has been given (e.g., established by a trusted key management center). Here, p is the number of parties including both the cloud server and users, and t is the minimum number of parties needed to complete the decryption. Thus, each party in this framework has known the public encryption key $pk = (g, n)$, while the matching private decryption key has been divided and distributed to all parties (i.e., party i has got his private key share sk_i).

As shown in Figure 1, after the objects are assigned by the cloud server, the PPTD parties will iteratively conduct the following two phases:

Phase 1: Secure Weight Update. In this phase, each user first calculates the distances between his observation values and the estimated ground truths provided by the cloud server according to the distance functions, then encrypts the distance information and submits the ciphertexts to the cloud server. After receiving the ciphertexts from all users, the cloud server securely updates the weight in encrypted form for each user. Then, the ciphertext of updated weight is sent to each corresponding user.

Phase 2: Secure Truth Estimation. Based on the encrypted weight received from the cloud server, each user calculates the ciphertexts of weighted observation values without decrypting the weight and then submits them to the cloud server. When the cloud server receives all the ciphertexts of weighted observation values from crowd users, it is able to estimate the ground truth for each object.

The above two phases start with a random initialization of the ground truth for each object, and are then iteratively conducted until convergence. Throughout the PPTD procedure, all the operations are conducted on encrypted data. Thus, it is ensured that the observation values of each user are known only to himself and the user weights are not disclosed to any party in the crowd sensing system.

4.2 PPTD Mechanism

In this part, we will elaborate on the mechanism of the proposed PPTD framework. Before we get into the details of the aforementioned *Secure Weight Update* and *Secure Truth Estimation* phases, we will first introduce a *Secure Sum Protocol* designed to calculate the summation of the data collected from users without disclosing them to any unintended party of the system.

4.2.1 Secure Sum Protocol. According to Equations (2) and (5), the cloud server needs to calculate the summation of the data collected from users in order to update user weights and estimate ground truths. However, the plaintext of each user's data should not be accessible to the cloud server due to privacy concerns. To address this problem, we design a secure sum protocol based on the threshold Paillier cryptosystem [8]. As shown in Protocol 1, the proposed secure sum protocol can calculate the summation of users' data without disclosing any of them.

PROTOCOL 1: Secure Sum Protocol

Input: The value $v_k \in \mathbb{Z}_n$ from each user $k \in \mathcal{K}$
Output: The summation $\sum_{k=1}^{K} v_k$

1. According to Equation (8), each user $k \in \mathcal{K}$ encrypts value v_k and sends the ciphertext $E_{pk}(v_k)$ to the cloud server $\mathcal{S}$;
2. Server $\mathcal{S}$ calculates $C = E_{pk}(\sum_{k=1}^{K} v_k) = \prod_{k=1}^{K} E_{pk}(v_k)$ based on Equation (9);
3. Server $\mathcal{S}$ randomly selects $t-1$ users and sends C to them;
4. Each selected user k' calculates the partial decryption $C_{k'}$ of C based on Equation (11) and sends $C_{k'}$ to the cloud server;
5. Server $\mathcal{S}$ calculates its partial decryption $C_{\mathcal{S}}$ and then combines it with $t-1$ other partial decryptions received from users to get the summation $\sum_{k=1}^{K} v_k$;

As we can see, in this protocol, what the cloud server received from users are the encrypted values and partial decryptions. Moreover, all the calculations on the cloud server are conducted on encrypted data. What the cloud server can know at last is the summation of all the users' data, based on which, each user's data can not be inferred. So the privacy of users is preserved.

4.2.2 Secure Weight Update. The first phase in our proposed framework is the secure weight update for each user. As aforementioned, the weight information needs to be updated in encrypted form in order not to be disclosed to any party. A challenge here is that the cryptosystem we use is defined over an integer ring, but the values needed to be encrypted in our framework may not be integers. To tackle this challenge, we introduce a parameter L (a magnitude of 10) to round the fractional values. For example, the value h can be rounded by multiplying L as $\widetilde{h} = \lfloor hL \rfloor$. Here, we

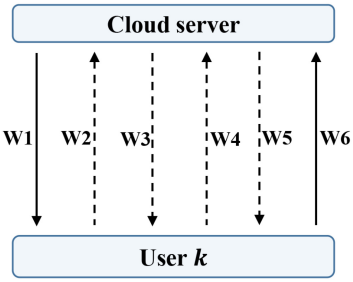


Fig. 2. Secure weight update for user k.

use $\widetilde{h}$ to denote the rounded integer of h and other values in this article will be represented in a similar way. The approximate value of h can be recovered by dividing L (i.e., $\widetilde{h}/L$).

Based on Equation (2), the encrypted weight can be updated as follows:

$$E_{pk}(\widetilde{w}_k) = E_{pk}\left(\left\lfloor L \cdot \left(log\left(\sum_{k'=1}^{K} Dist_{k'}\right) - log(Dist_k)\right)\right\rfloor\right), \tag{12}$$

where $Dist_k = \sum_{m=1}^{M} d(x_m^k, x_m^*)$ is the summation of distances between the k-th user's observation values $\{x_m^k\}_{m=1}^M$ and the estimated ground truths $\{x_m^*\}_{m=1}^M$. As we can see, in order for the cloud server to update $E_{pk}(\widetilde{w}_k)$, it needs to collect the information about $Dist_k$ from users. This procedure can be shown in Figure 2. For the sake of simplicity, we take the k-th user as an example in this figure.

Since the distance functions for continuous data and categorical data are different, we need to consider them separately when calculating distances. For categorical data, user k can easily calculate distances based on Equation (4). But for continuous data, we need to know the standard deviation std_m according to Equation (3), which is difficult to derive without knowing the observation values of other users. Next, we first introduce the common steps (**W1** and **W6** in Figure 2) for all the data types to update a user's weight, and then specifically discuss the calculation of std_m for continuous data (**W2**, **W3**, **W4**, and **W5** in Figure 2).

Step W1. The cloud server sends the estimated ground truths $\{x_m^*\}_{m=1}^M$ to user k. If it is the first iteration, the estimated ground truths will be randomly initialized. If it is not, the estimated ground truths are obtained from the previous iteration. When user k receives the estimated ground truths, he will first calculate two values: $Dist_k$ and $\log Dist_k$. Before the two values are submitted, user k needs to encrypt them for the purpose of privacy. For $Dist_k$, user k privately selects a random $r_{k1} \in \mathbb{Z}_n^*$, and then encrypts it as follows based on Equation (8).

$$E_{pk}(\widetilde{Dist_k}) = g^{\widetilde{Dist_k}} r_{k1}^n \ mod \ n^2 \tag{13}$$

Similarly, for $\log Dist_k$, user k privately selects another random $r_{k2} \in \mathbb{Z}_n^*$, and encrypts it as

$$E_{pk}(\widetilde{\log Dist_k}) = g^{\widetilde{\log Dist_k}} r_{k2}^n \ mod \ n^2 \tag{14}$$

Step W6. After the encryption in the above step, user k submits both $E_{pk}(\widetilde{Dist_k})$ and $E_{pk}(\widetilde{\log Dist_k})$ to the cloud server $\mathcal{S}$. Upon receiving the ciphertexts from all users, $\mathcal{S}$ calculates $sum_D = \sum_{k=1}^{K} \widetilde{Dist_k}/L$ and $\log sum_D$ based on the secure sum protocol. Then, $\mathcal{S}$ encrypts $\log sum_D$ according to Equation (8) as

$$E_{pk}(\widetilde{\log sum_D}) = g^{\widetilde{\log sum_D}} r_{s1}^n \ mod \ n^2, \tag{15}$$

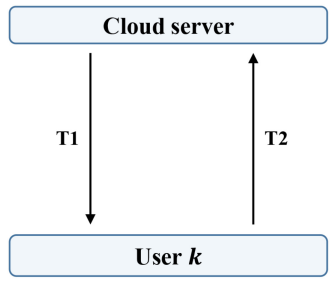


Fig. 3. Secure truth estimation.

where $r_{s1} \in \mathbb{Z}_n^*$ is privately and randomly selected by server $\mathcal{S}$. With above ciphertexts, $\mathcal{S}$ can update the encrypted weight for user k as follows based on Equations (9), (10), and (12).

$$\begin{aligned} E_{pk}(\widetilde{w}_k) &= E_{pk}(\widetilde{\log sum_D}) \cdot E_{pk}(-\widetilde{\log Dist_k}) \\ &= E_{pk}(\widetilde{\log sum_D}) \cdot E_{pk}(\widetilde{\log Dist_k})^{-1} \end{aligned} \tag{16}$$

As for continuous data, as discussed previously, the standard deviation std_m should be first calculated. The calculation steps are described in detail as below (these steps only need to be performed once throughout the whole truth discovery procedure).

Step W2. According to Equation (8), user k encrypts his observation value for object m as $E_{pk}(\widetilde{x}_m^k)$ and sends the ciphertext to the cloud server $\mathcal{S}$.

Step W3. After receiving the ciphertexts from all users, server $\mathcal{S}$ calculates $sum_x = \sum_{k=1}^{K} \widetilde{x}_m^k / L$ and $\bar{x}_m = sum_k / K$ based on the secure sum protocol, and then sends $\bar{x}_m$ to users.

Step W4. User k calculates $d_m^k = (x_m^k - \bar{x}_m)^2$ and encrypts d_m^k as $E_{pk}(\widetilde{d}_m^k)$. Then k sends $E_{pk}(\widetilde{d}_m^k)$ to server $\mathcal{S}$.

Step W5. When server $\mathcal{S}$ receives $E_{pk}(\widetilde{d}_m^k)$ from all users, $\mathcal{S}$ calculates $sum_d = \sum_{k=1}^{K} \widetilde{d}_m^k / L$ and std_m (equals to $\sqrt{sum_d / K}$) through the secure sum protocol. Then, $\mathcal{S}$ sends std_m to users.

4.2.3 Secure Truth Estimation. After updating user weights, the next thing is to estimate the ground truth for each object. As shown in Figure 3, there are two major steps in this phase, which are detailed as follows.

Step T1. The cloud server sends the encrypted weight $E_{pk}(\widetilde{w}_k)$ (updated in the secure weight update phase) to user k. Then, user k calculates the ciphertexts of weighted observation values based on the encrypted weight. For continuous data, user k calculates the ciphertexts according to Equation (10) using the following formula:

$$E_{pk}\left(\widetilde{w}_k \cdot \widetilde{x}_m^k\right) = E_{pk}(\widetilde{w}_k)^{\widetilde{x}_m^k} \tag{17}$$

For categorical data, x_m^k is a vector as described in Section 3.1, so user k needs to calculate the ciphertext for each element in this vector as follows:

$$E_{pk}(\widetilde{w}_k \cdot x_m^k(i)) = \begin{cases} E_{pk}(0) & if\ x_m^k(i) = 0 \\ E_{pk}(\widetilde{w}_k) \cdot E_{pk}(0) & if\ x_m^k(i) = 1, \end{cases} \tag{18}$$

where $x_m^k(i)$ denotes the i-th element in vector x_m^k. Please note that $E_{pk}(0)$ can be dynamically changing because, every time, the encryption procedure is conducted with a different random $r_k \in \mathbb{Z}_n^*$.

Step T2. After the calculation in the above step, user k submits the ciphertexts of weighted data for all the objects to the cloud server $\mathcal{S}$. When receiving ciphertexts from all the users, $\mathcal{S}$ will first calculate the numerator of Equation (5) as follows.

For *continuous data*, server $\mathcal{S}$ calculates the summation of weighted data (i.e., $\sum_{k=1}^{K}(\widetilde{w}_k \cdot \widetilde{x}_m^k)$) with the help of the secure sum protocol, and then derives the approximation of $\sum_{k=1}^{K}(w_k \cdot x_m^k)$ (i.e., the numerator) via dividing the summation by L^2.

For *categorical data*, we need to consider each element in the vector separately. Specifically, for the i-th element, server $\mathcal{S}$ calculates the summation of the weighted data (i.e., $\sum_{k=1}^{K}(\widetilde{w}_k \cdot x_m^k(i))$ via the secure sum protocol, and then gets the approximation of $\sum_{k=1}^{K}(w_k \cdot x_m^k(i))$ (i.e., the numerator). The summations of other elements are calculated in the same way.

As the denominator of Equation (5), the summation of weights is also needed to estimate the ground truths. This can be easily calculated through the secure sum protocol because $\mathcal{S}$ has already stored encrypted weights in the weight update phase. Then, the ground truth for each object $m \in \mathcal{M}$ can be estimated by the cloud server based on Equation (5).

Please note that the ground truths estimated in this step for categorical data are probability values that are used for updating user weights in the next iteration. The final estimation for object m should be the choice with the largest probability in vector x_m^* obtained in the final iteration.

Combining the secure weight update and secure truth estimation phases, we summarize the proposed PPTD procedure in Protocol 2. This protocol repeats the aforementioned two phases iteratively until some convergence criterion is satisfied. Then, the cloud server can output the final estimated ground truth for each object.

PROTOCOL 2: Privacy-Preserving Truth Discovery Protocol

Input: K users, M objects, observation values $\{x_m^k\}_{m,k=1}^{M,K}$ and rounding parameter L

Output: Estimated ground truths $\{x_m^*\}_{m=1}^{M}$

1. The cloud server $\mathcal{S}$ randomly initializes the ground truth for each object;
2. The cloud server $\mathcal{S}$ sends the estimated ground truths (i.e., $\{x_m^*\}_{m=1}^{M}$) and the rounding parameter L to users;
3. Each user $k \in \mathcal{K}$ calculates $Dist_k = \sum_{m=1}^{M} d(x_m^k, x_m^*)$ and gets the rounded values (i.e., $\widetilde{Dist_k}$ and $\widetilde{\log Dist_k}$) with parameter L. Then, user k encrypts them as $E_{pk}(\widetilde{Dist_k})$, $E_{pk}(\widetilde{\log Dist_k})$ and sends the ciphertexts to the cloud server;
4. After receiving ciphertexts from all the users, the server $\mathcal{S}$ calculates $sum_D = \sum_{k=1}^{K} \widetilde{Dist_k}/L$ based on the secure sum protocol, then updates the encrypted weight of each user according to Equations (15) and (16). Also, the updated ciphertext of weight is sent to each corresponding user;
5. When user $k \in \mathcal{K}$ receives encrypted weight from the cloud server, the user calculates ciphertexts of weighted data for continuous data and categorical data, respectively, according to Equations (17) and (18). Then, these ciphertexts are sent to the cloud server;
6. After receiving ciphertexts from all the users, the cloud server $\mathcal{S}$ estimates the ground truths $\{x_m^*\}_{m=1}^{M}$ based on step **T2**;
7. Repeat steps 2~6 until the convergence criterion is satisfied, and then output $\{x_m^*\}_{m=1}^{M}$;

5 PARALLEL PPTD

With the proliferation of human-carried sensing devices, an explosive increase of crowd sensing data is expected in the near future. In order to deal with such massive data, we extend our proposed scheme in a parallel way using the MapReduce framework, which contains two major functions: the Map function that processes input values to generate a set of intermediate key/value pairs, and

the Reduce function that merges all intermediate values associated with the same intermediate key. Here, we just borrow the existing MapReduce framework, in which we do not make research contribution.

We only adapt the truth estimation phase to MapReduce framework, and there is no change in the weight update procedure. In the Map function for estimating ground truths, the input is a list of records: $(m, E_{pk}(\widetilde{w}_k \cdot \widetilde{x}_m^k), k)$, where $m \in \mathcal{M}$, $k \in \mathcal{K}$, and $E_{pk}(\widetilde{w}_k \cdot \widetilde{x}_m^k)$ is the encrypted weighted data. As shown in Algorithm 2, during the mapping process, all the input records are reorganized into key/value pairs, where the key is the ID of each object (i.e., m), and the value is the rest of the information. Before these key/value pairs are fed to Reducers, they will be sorted by Hadoop so that the pairs that have the same key (i.e., the same object ID m) will go to the same Reducer. In the Reducers, as seen in Algorithm 3, the truth value for each object is estimated based on step **T2** described in Section 4.2.3. Since users' weight information is also needed, we use an external file to store the encrypted weights, and all the Reducer nodes can read it. Finally, for each object, a key/value pair is outputted, where the key is object ID m and the value is the estimated ground truth. The two procedures (i.e., distributed weight update and parallel truth estimation) are iteratively conducted until the whole procedure converges.

ALGORITHM 2: Map function for estimating truths

Input: A list of records: $(m, E_{pk}(\widetilde{w}_k \cdot \widetilde{x}_m^k), k), m \in \mathcal{M}, k \in \mathcal{K}$

Output: A list with each element in format of $[m, [E_{pk}(\widetilde{w}_k \cdot \widetilde{x}_m^k), k]], m \in \mathcal{M}, k \in \mathcal{K}$

```
output_list ← [ ];
for each record from input do
    Parse the record;
    Append output_list with the new record [m, [E_pk(w̃_k · x̃_m^k), k]];
end
return output_list;
```

ALGORITHM 3: Reduce function for estimating truths

Input: A list of records (sorted by object ID m): $[m, [E_{pk}(\widetilde{w}_k \cdot \widetilde{x}_m^k), k]], m \in \mathcal{M}, k \in \mathcal{K}$

Output: A list with each element in the format of $[m, x_m^*]$

```
output_list ← [ ];
Read encrypted weights of crowd users from file;
Calculate the summation of weights based on the secure sum protocol;
for all the records with the same objectID m do
    Calculate the summation of weighted data through the secure sum protocol;
    Estimate ground truth x*_m based on step T2;
    Append output_list with the new record [m, x*_m];
end
return output_list;
```

6 INCREMENTAL PPTD

In many real-world crowd sensing applications, the sensing tasks may last several days or months and the observation values of different objects are usually collected from users in a "streaming"

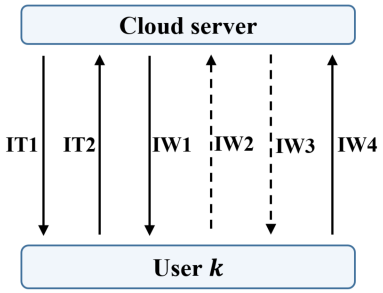


Fig. 4. Incremental PPTD.

manner. In such scenarios, it is inefficient to infer the ground truth for each object until all the objects are observed. For example, in transportation monitoring applications [54], the traffic information is reported by multiple users in real time; it is inefficient to evaluate the route conditions until all the traffic information at different time periods is observed by the users. In healthcare applications [18], patients' health data are usually collected day by day; it is inefficient to analyze these data after several weeks or several months. In order to address this challenge, a possible way is to conduct the PPTD scheme once again on the whole dataset whenever some new objects are observed. However, tremendous unnecessary calculations would be involved to iteratively update the estimated truths of the previous observed objects. To tackle this problem, we design an incremental PPTD scheme that can timely estimate the ground truth of the newly observed object without disclosing the private information of each user and revisiting the old data.

Different from PPTD, we design the incremental PPTD by first conducting the truth estimation and then updating each user's weight when a new object is observed. For each user k, we use $D_k = \sum_{m=1}^{l-1} d(x_m^k, x_m^*)$ to denote his "old information" with respect to the previously observed $l-1$ objects. Here, D_k represents the summation of the distances between user k's observations and the estimated truths for the $l-1$ objects observed in the past. When the l-th object is observed, the secure truth estimation phase and secure weight update phase are conducted as Figure 4.

Secure Truth Estimation. In this phase, the ground truth of the newly observed object is estimated based on user weights, which are calculated based on the historical data. This phase contains two steps, i.e., IT1 and IT2 in Figure 4.

Step IT1. Server $\mathcal{S}$ sends user k the rounding parameter L and the encrypted weight $E_{pk}(\widetilde{w}_k)$, which is calculated by server $\mathcal{S}$ in the *secure weight update* phase for the $(l-1)$-th object. Then, each user k calculates the ciphertext of the weighted data (i.e., $E_{pk}(\widetilde{w}_k \cdot \widetilde{x}_l^k)$ or $E_{pk}(\widetilde{w}_k \cdot x_l^k(i))$) for the l-th object according to Equations (17) or (18). Here, we use x_l^k to denote the observation value of user k for the l-th object, and $\widetilde{x}_l^k$ denotes its approximate value. For continuous data, user k also needs to encrypt his observation value as $E_{pk}(\widetilde{x}_l^k)$ in order to calculate the standard deviation std_l.

Step IT2. User k submits the ciphertext of weighted data to server $\mathcal{S}$. Similar to **Step T2** in Section 4.2.3, server $\mathcal{S}$ calculates the estimated truth (i.e., x_l^*) for the l-th object based on the secure sum protocol and Equation (5). For continuous data, user k also needs to upload the encrypted data $E_{pk}(\widetilde{x}_l^k)$, and then the average observation value $\bar{x}_l$ is calculated based on the secure sum protocol.

Secure Weight Update. After the truth for the l-th object is estimated, each user's weight is securely updated in this phase. Similar to the PPTD scheme, here, we also need to consider continuous data and categorical data separately when calculating distances. We first introduce the common steps (IW1 and IW4 in Figure 4) for the two types of data and then discuss the specific steps (IW2 and IW3 in Figure 4) to calculate the standard deviation std_l for continuous data.

Step IW1. Server $\mathcal{S}$ sends the estimated truth x_l^* to each user. For continuous data, the average value $\bar{x}_l$ is also sent to users. After receiving the value x_l^*, user k first updates D_k as $D_k = D_k + d(x_l^k, x_l^*)$, then calculates the ciphertexts $E_{pk}(\widetilde{D_k})$ and $E_{pk}(\widetilde{\log D_k})$ according to Equations (19) and (20):

$$E_{pk}(\widetilde{D_k}) = g^{\widetilde{D_k}} r'^n_{k1} \ mod \ n^2 \tag{19}$$

$$E_{pk}(\widetilde{\log D_k}) = g^{\widetilde{\log D_k}} r'^n_{k2} \ mod \ n^2, \tag{20}$$

where $r'_{k1}, r'_{k2} \in \mathbb{Z}_n^*$ are privately selected by user k.

Step IW4. User k submits both $E_{pk}(\widetilde{D_k})$ and $E_{pk}(\widetilde{\log D_k})$ to server $\mathcal{S}$. After receiving the ciphertexts from all users, $\mathcal{S}$ updates the encrypted weight $E_{pk}(\widetilde{w}_k)$ for user k based on **Step W6** in Section 4.2.2.

For continuous data, the standard deviation std_l is calculated as follows.

Step IW2. After receiving the average value $\bar{x}_l$, user k calculates $d_l^k = (x_l^k - \bar{x}_l)^2$. Then, the ciphertext $E_{pk}(\widetilde{d_l^k})$ is sent to server $\mathcal{S}$.

Step IW3. When server $\mathcal{S}$ receives $E_{pk}(\widetilde{d_l^k})$ from all users, $\mathcal{S}$ calculates std_l based on the secure sum protocol. Then, the standard deviation std_l is sent to each user.

The incremental privacy-preserving protocol is summarized as Protocol 3.

PROTOCOL 3: Incremental Privacy-Preserving Truth Discovery Protocol

Input: K users, each user's encrypted weight (i.e., $\{E_{pk}(\widetilde{w}_k)\}_{k=1}^K$) after the $(l-1)$-th object is observed, each user's observation value $\{x_l^k\}_{k=1}^K$ for the l-th object, each user's "old information" D_k, and the rounding parameter L.

Output: Estimated ground truth x_l^* of the l-th object.

1. The cloud server $\mathcal{S}$ sends the encrypted weight $E_{pk}(\widetilde{w}_k)$ and the rounding parameter L to user k;
2. Each user $k \in \mathcal{K}$ calculates the ciphertexts of the weighted data according to Equations (17) and (18). For continuous data, user k also needs to calculate the encrypted data $E_{pk}(\widetilde{x_l^k})$. Then, the ciphertexts of the weighted data and the encrypted data are submitted to server $\mathcal{S}$.
3. After receiving the ciphertexts from all users, server $\mathcal{S}$ estimates the ground truth x_l^* based on *Step IT2* and sends it to each user. For continuous data, the average value $\bar{x}_l$ should also be calculated and sent to users.
4. Each user $k \in \mathcal{K}$ updates D_k with $D_k + d(x_l^k, x_l^*)$, then calculates the ciphertexts $E_{pk}(\widetilde{D_k})$ and $E_{pk}(\widetilde{\log D_k})$ according to Equations (19) and (20). Here, the standard deviation std_l for continuous data is calculated based on *Step IW2* and *Step IW3*.
5. User k submits both $E_{pk}(\widetilde{D_k})$ and $E_{pk}(\widetilde{\log D_k})$ to server $\mathcal{S}$. Then, $\mathcal{S}$ updates the encrypted weight $E_{pk}(\widetilde{w}_k)$ for user k based on **Step W6** in Section 4.2.2.

From Protocol 3, we can see when a new object is observed; the ground truth of this object can be estimated in real time without disclosing each user's private information. Since the "old information" of user k has been integrated in the value D_k, there is no need to revisit the observation values of the past objects when estimating the ground truth of the new observed object. Although the iterative procedure is not involved in this scheme, and the two phases (i.e., secure truth estimation and secure weight update) are conducted only once, the weight of each user will converge to stabilization when the number of objects increases. Additionally, this incremental scheme can be easily modified to fit the scenario where the objects are grouped into sequential chunks, of which, each may contain multiple objects. In such scenario, we repeat the two phases for each chunk and

update D_k of user k as $D_k = D_k + \sum_{m=1}^{C_\epsilon} d(x_m^k, x_m^*)$, where C_ϵ is the number of objects in the ϵ-th chunk.

7 PRIVACY ANALYSIS

As previously discussed, the security threats mainly come from the parties themselves in practical crowd sensing systems. Thus, the goal of PPTD is to protect the observation values of each user from being disclosed to other parties, and at the same time, the weight of each user should not be known by any party. Since our framework is built upon the proposed secure sum protocol, we start with the privacy analysis of this protocol.

In the secure sum protocol, the data are exchanged only between cloud server and users, and all the exchanged data are ciphertexts. Although some users obtain the ciphertext of summation $E_{pk}(\sum_{k=1}^{K} v_k)$, they cannot decrypt it because of the (p, t)-threshold Paillier cryptosystem we used, and there is no collusion among users. Thus, the users will learn nothing after the execution of the protocol. Similarly, the ciphertext $E_{pk}(v_k)$ cannot be decrypted by the cloud server, and what the server can know at last is just the summation $\sum_{k=1}^{K} v_k$, based on which it cannot infer the input value v_k of each user. In this way, the privacy of each user's input value is guaranteed by this protocol.

Then, we can summarize the privacy-preserving goal of our framework as Theorem 7.1, followed by the proof.

Theorem 7.1. *Suppose $K \geq 3$, and for each object $m \in \mathcal{M}$, there are at least two users $k_1, k_2 \in \mathcal{K}$ giving different observation values (i.e., $x_m^{k_1} \neq x_m^{k_2}$). Also, assume the parties are semi-honest, and there is no collusion among them. Then, after the execution of the PPTD protocol, the observation values of each user will not be disclosed to others and the weight of each user will not be known by any party.*

Proof. First, we prove the observation values of each user will not be disclosed to others in our framework. We can achieve the goal by proving that there is not an attack algorithm, based on which one party can infer the private observation values of the users.

For the cloud server, we assume there exists an attack algorithm based on which the server can infer the observation values of user $k_1 \in \mathcal{K}$. The input of the algorithm should be the plaintexts the server knows during the PPTD procedure. These plaintexts are $\sum_{k=1}^{K} x_m^k$, $\bar{x}_m$, $\sum_{k=1}^{K} d_m^k$, std_m, $\sum_{k=1}^{K} Dist_k$, $\sum_{k=1}^{K} w_k$, $\sum_{k=1}^{K} (w_k \cdot x_m^k)$, and the estimated ground truth x_m^* for each $m \in \mathcal{M}$. Also, the cloud server knows the values K and M. According to our assumption, the server can infer the observation value $x_m^{k_1}$ ($m \in \mathcal{M}$) of user k_1 based on these input values. We also assume another user $k_2 \in \mathcal{K}$ has the observation value $x_m^{k_2}$ ($\neq x_m^{k_1}$) for the object m. Now, we exchange the observation values of k_1 and k_2, which means user k_1 has the observation value $x_m^{k_2}$ and user k_2 has the observation value $x_m^{k_1}$ for the object m after the exchange. Then, we restart the PPTD procedure. However, the plaintexts known by the server will not be changed based on our framework. That is to say, the input values of the attack algorithm will not be changed. So, based on this algorithm, the cloud server would still infer the value $x_m^{k_1}$ for user k_1. However, now the observation value of user k_1 has been changed to $x_m^{k_2}$. Obviously, there is a contradiction. Therefore, such an attack algorithm does not exist and the cloud server cannot infer the observation values of users in our framework.

For each user, he can know the public values $\bar{x}_m$, std_m, x_m^* for each $m \in \mathcal{M}$ besides his private observation values based on our framework. Using the same method above, we can also prove that this user cannot infer the observation values of others.

Next, we prove the weight of each user (i.e., $w_k, k \in \mathcal{K}$) will not be disclosed to any party in our framework.

Based on the PPTD protocol, the cloud server updates the ciphertexts of the weights (i.e., $E_{pk}(w_k)$, $k \in \mathcal{K}$) instead of the plaintexts of them in each iteration. Also, the users calculate the weighted data based on the ciphertexts of weights. Based on the semi-honest and non-collusion assumptions, all the parties cannot decrypt each encrypted weight. The only plaintexts about the weights are the summations $\sum_{k=1}^{K} w_k$ and $\sum_{k=1}^{K} w_k \cdot x_m^k$, which are known by the cloud server. Since x_m^k is only known by the user, the server cannot infer w_k based on the above summations. So the weight information will not be disclosed to any party in this framework. □

As for the incremental PPTD scheme, the private information of each user can also be well protected from being disclosed to others. When the l-th object is observed, the plaintexts known by the cloud server are $\sum_{k=1}^{K} x_l^k$, $\bar{x}_l$, $\sum_{k=1}^{K} d_l^k$, std_l, $\sum_{k=1}^{K} D_k$, $\sum_{k=1}^{K} w_k$, $\sum_{k=1}^{K}(w_k \cdot x_l^k)$, and x_l^*. However, the cloud server can not infer each user's private information from these plaintexts according to the proof described above. Additionally, each user in this scheme knows the public values $\bar{x}_l$, std_l, x_l^*, and his own observation value. However, based on these plaintexts, he can not infer the observation values of others and the weight information.

8 DISCUSSIONS

Since the cryptosystem adopted here is defined over an integer ring, we use parameter L to round the fractional values to integers. During the rounding process, numerical errors are inevitably introduced. However, the accuracy of the final estimated ground truth will not be greatly affected if we select the appropriate L, which is shown in Section 9.

Another issue we are concerned with is missing values, which means not all the objects are observed by all the crowd users. This can be easily handled in our framework. When different users observe different subsets of the objects, we can normalize the aggregate deviations of each user by the number of his observations.

Also, to tackle the issue that some users could not respond timely after the sensing tasks are released, we can set a waiting-time threshold on the cloud server. Based on the (p, t)-threshold Paillier cryptosystem adopted in this article, as long as at least $t - 1$ users could upload their data in time, the PPTD procedure can be completed.

Additionally, our proposed framework can be easily modified to the situation where the user weight is known only to the user himself. In this case, the weight values are updated by users themselves. In particular, during the weight update procedure, user $k \in \mathcal{K}$ just needs to submit the encrypted summation of the distances $E_{pk}(\widetilde{Dist_k})$. Then, the cloud server calculates $\sum_{k=1}^{K} \widetilde{Dist_k}/L$ through the secure sum protocol. Based on this summation, each user can privately update his weight according to Equation (2). In the truth estimation procedure, user $k \in \mathcal{K}$ submits the ciphertexts of weighted data $\{E_{pk}(\widetilde{w_k \cdot x_m^k})\}_{m=1}^{M}$ and the encrypted weight $E_{pk}(\widetilde{w}_k)$. Then, the cloud server can estimate ground truth for each object via the same method used in PPTD (Step **T2**).

Next, we discuss some limitations of PPTD. In this article, we assume that all the parties are semi-honest, and there is no collusion among them. If these assumptions cannot be guaranteed, the observation values and the weight information of individual users may be disclosed. For example, if the parties are not semi-honest, the cloud server can send the ciphertexts of users' private information (e.g., $E_{pk}(\widetilde{x}_m^k)$ or $E_{pk}(\widetilde{w}_k)$) instead of the encrypted summations to $t - 1$ users and then get this private information according to Protocol 1. Additionally, if the cloud server colludes with $t - 1$ users, the private information of other users can also be disclosed. Thus, an interesting direction of future work is how to improve PPTD and enable it to resist collusions or dishonest

behaviors of the parties. Another limitation of this work is the computational and communication overhead introduced by the adopted cryptosystem. Compared with the truth discovery approaches (e.g., CRH) that do not take actions to protect user privacy, PPTD requires each user to conduct ciphertext-based calculations and communication with the cloud server, and this inevitably introduces more overhead to individual users. So, another problem worthy of studying in future work is how to make the proposed framework lightweight and reduce the overhead introduced to the parties.

9 PERFORMANCE EVALUATION

In this section, we evaluate the proposed PPTD framework on both real-world crowd sensing systems and synthetic datasets.

9.1 Experiment Setup

In this article, we consider two different types of data: continuous data and categorical data. To evaluate the estimation accuracy of PPTD, we use following measures for the two data types:

- *MAE*: For continuous data, we use the mean of absolute error (*MAE*), i.e., $\frac{1}{M}\sum_{m=1}^{M}|x_m^* - \hat{x}_m^*|$, to measure the mean of absolute distance between the estimated results and ground truths. Here, $\hat{x}_m^*$ denotes the ground truth of object m.
- *RMSE*: For continuous data, we also use the root of mean squared error (*RMSE*), i.e., $\sqrt{\frac{1}{M}\sum_{m=1}^{M}(x_m^* - \hat{x}_m^*)^2}$, to measure the accuracy. Compared with *MAE*, *RMSE* can penalize more on the large distance and less on the small distance.
- *ErrorRate*: For categorical data, we calculate the percentage of mismatched values between estimated results and ground truths as *ErrorRate*.

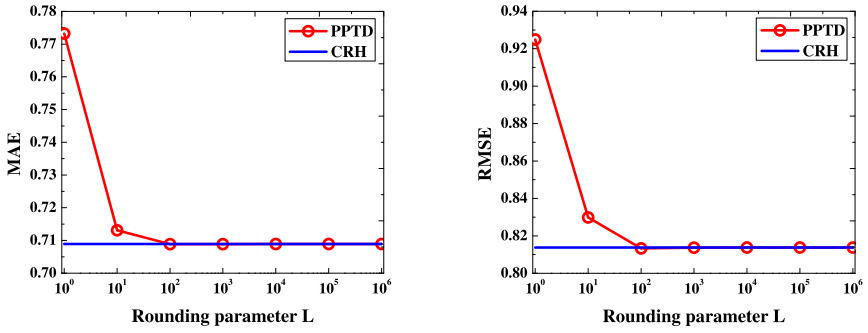
The baseline approach we use in this experiment is the state-of-the-art truth discovery scheme, i.e., CRH [33, 37], which does not take any actions to protect user privacy during the whole procedure.

A $(p, \lfloor \frac{p}{2} \rfloor)$-threshold Paillier cryptosystem is used in our experiment, and, here, we fix the key size as 512 (can also be set as other values according to the practical demand). Our framework was implemented in Java 1.7.0 using the Paillier Threshold Encryption Toolbox.[1] The sensing devices we use are Nexus 4 Android phones. The "cloud" is emulated by a cluster of three Intel(R) Core(TM) 3.40GHz PCs running Ubuntu 14.04, with 8GB RAM. When implementing parallel PPTD framework, we use a 15-node Dell Hadoop cluster with Intel Xeon E5-2403 processor (4 × 1.80 GHz, 48GB RAM) as the "cloud."

9.2 Experiment on Crowdsourced Indoor Floorplan Construction System

In this part, we show the experiment results on continuous data collected from a real-world crowd sensing system to demonstrate the advantages of PPTD. The application is crowdsourced indoor floorplan construction [1, 2, 15], which has recently drawn much attention since many location-based services can be facilitated by it. The goal of such crowd sensing system is to automatically construct an indoor floorplan from sensory data (e.g., the readings of compass, accelerometer, gyroscope) collected from smartphone users. Clearly, these sensor readings encode the private personal activities of the phone user, and thus, the user may not be willing to share such data without the promise of privacy protection. For the sake of illustration, here, we focus on just one task of indoor floorplan construction, namely, to estimate the distance between two particular

[1] http://cs.utdallas.edu/dspl/cgi-bin/pailliertoolbox/.



(a) MAE (b) RMSE

Fig. 5. Ground truth estimation errors under different values of the parameter L.

location points in the hallway. We develop an Android App that can estimate the walking distances of a smartphone user through multiplying the user's step size by step count inferred using the in-phone accelerometer.

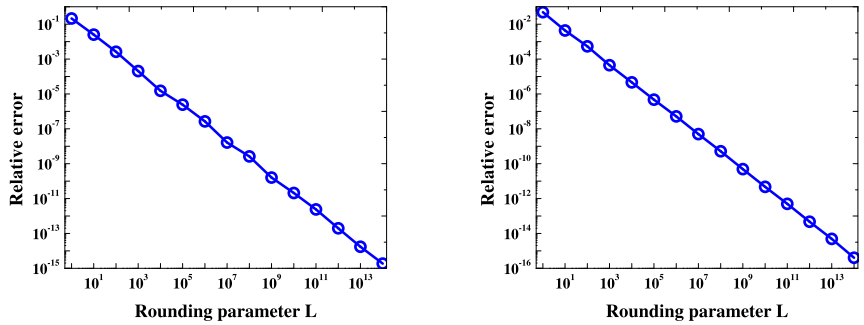
In our experiment, 10 volunteers are employed as smartphone users and we select 27 hallway segments in a building as the *objects*. Each party (including the cloud server and smartphone users) in this experiment holds the public key and the corresponding private key share, which are produced by the cryptosystem. The ground truths of these hallway segments are obtained by measuring them manually.

Accuracy. We first compare the accuracy of the final estimated ground truths between PPTD and the baseline approach (i.e., CRH). Since the estimation errors of PPTD are introduced by the rounding parameter L, we vary L from 10^0 to 10^6 and measure the corresponding accuracy. In the experiment, we randomly initialize the estimated ground truths and use a threshold of the change in the estimated ground truths in two consecutive iterations as the convergence criterion. The experiment is repeated for 20 times, and we report the averaged results.

Figure 5 shows the ground truth estimation errors of our proposed framework and CRH under different values of the parameter L. The estimation error is measured in terms of *MAE* and *RMSE*, respectively. As seen in the figure, the estimation error of PPTD is almost the same as that of CRH unless the rounding parameter L is too small (i.e., 10^0 or 10^1). This is because during the rounding procedure, the fractional part (i.e., decimal digits) of the original value (e.g., $L \cdot \log Dist_k$) is dropped. In this sense, the smaller the parameter L, the more decimal digits of the original value will be lost. To measure the information loss degree, we calculate the relative estimation errors of PPTD and CRH in both object truth and user weight. Here, we manually decrypt user weights for the analysis purpose. In particular, we define the relative error of user weight as $||\log \mathbf{w}_c - \log \mathbf{w}_p||/||\log \mathbf{w}_c||$, where $\mathbf{w}_c$ and $\mathbf{w}_p$ are the weight vectors of all the users obtained from CRH and PPTD, respectively. Similarly, we define the relative error of the estimated ground truths as $||\log \mathbf{x}_c^* - \log \mathbf{x}_p^*||/||\log \mathbf{x}_c^*||$, where $\mathbf{x}_c^*$ and $\mathbf{x}_p^*$ are obtained from CRH and PPTD respectively. The results are shown in Figure 6.

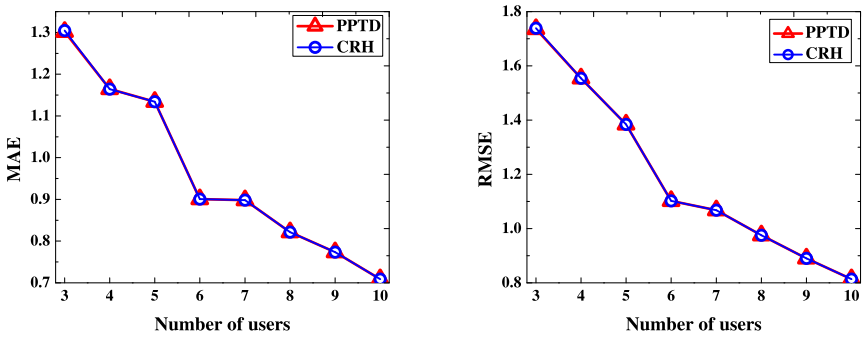
As shown in Figure 6(a) and (b), the relative errors in both truth and weight drop as the parameter L increases. That is to say, we do not need to worry about the estimation errors produced during the rounding procedure as long as we select a large enough parameter L.

Additionally, we evaluate the performance of PPTD under varying the number of users. The number of objects is still 27, while the number of users varies from 3 to 10. We also fix the parameter L as 10^{10} and use the same convergence criterion as before. Then, the experiment is repeated 20 times, and the averaged results are reported.



(a) Relative error of weight (b) Relative error of estimated truth

Fig. 6. Relative errors under different values of L.

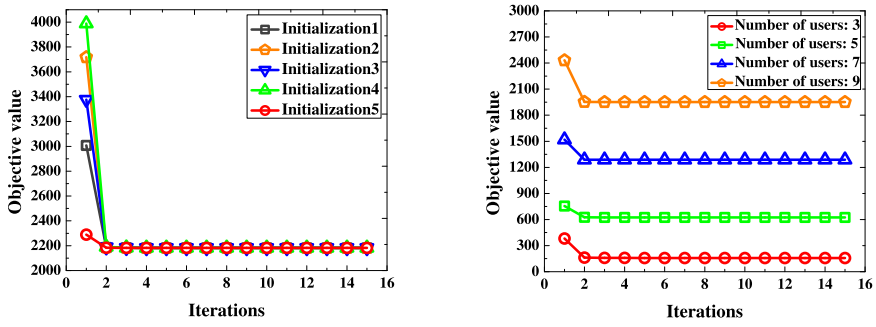


(a) MAE (b) RMSE

Fig. 7. Ground truth estimation errors under different numbers of users.

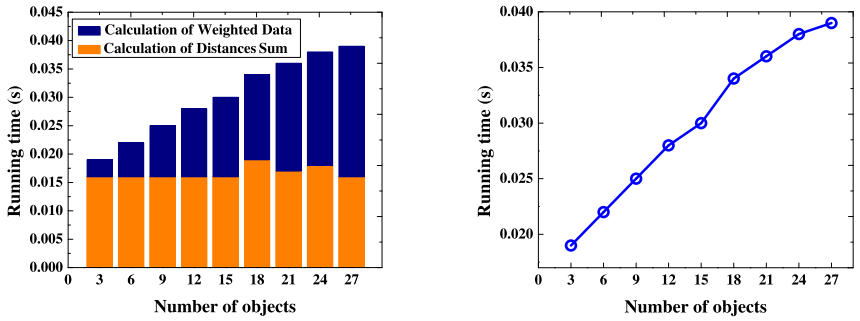
Figure 7(a) and (b) show that PPTD almost has the same estimation errors as CRH, while the number of users is varying, which means that our proposed framework is robust against the change of user numbers. Also, we can see that the estimation errors decrease with the increase of the number of users. This makes sense because it is hard to improve upon the users' individual poor observation values when the number of users that observe the same objects is small. When the number of users increases, each object is observed by more and more diversified crowd users; thus, it is more and more likely to cancel out individual users' biases and errors so as to reach higher accuracy.

Convergence. Next, we show the convergence of the PPTD procedure. In this experiment, the rounding parameter L is still set as 10^{10}. We first set the number of users as 10 and calculate the *objective value* of the truth discovery problem, which is defined as the weighted summation of the distances between individual observations and the estimated ground truths (i.e., $\sum_{k=1}^{K} w_k \sum_{m=1}^{M} d(x_m^k, x_m^*)$). We repeat the experiment five times with different random initialization values and report the evolution of the objective values in Figure 8(a). As we can see, all the objective values, although under different initializations, converge quickly within just a few iterations. Then, we evaluate the convergence of the PPTD procedure when the number of users varies. Here, we consider four cases where the number of users is set as 3, 5, 7, and 9, respectively, and for each case, the initialization values are randomly selected. The evolution of the objective values for the four cases are shown in Figure 8(b), from which we can see all the objective values can converge within just a few iterations.



(a) For different initializations (b) For different numbers of users

Fig. 8. Convergence w.r.t. iterations.



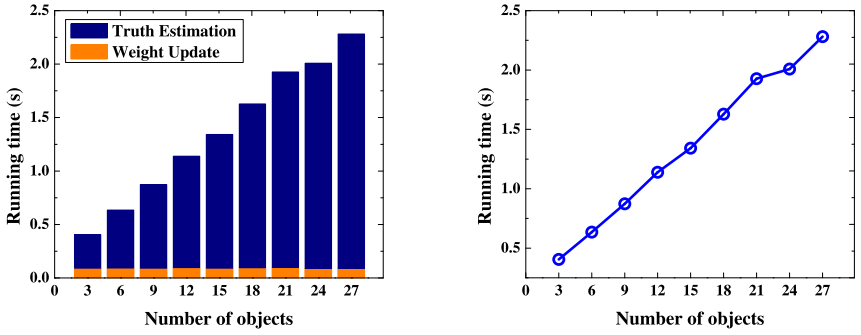
(a) Running time of each procedure (b) Total running time

Fig. 9. Running time w.r.t. the number of objects for continuous data on smartphones.

Computational Cost. In this part, we take a look at PPTD's computational cost, which is composed of the cost on the smartphone of each user and the cost on the cloud server. Here, we also fix the rounding parameter as 10^{10}, which actually has little effect on the computational time compared with user numbers and object numbers.

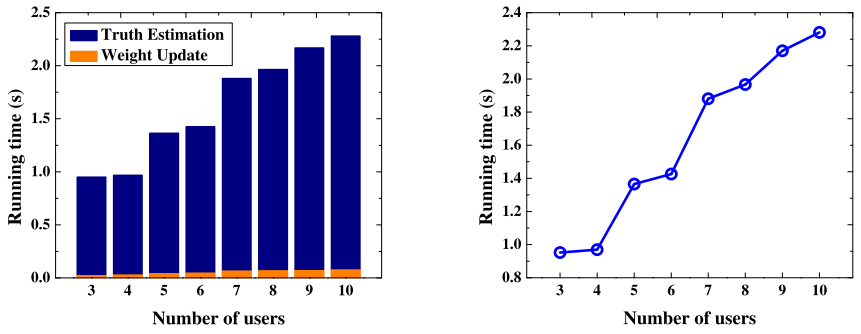
On the smartphone of each user, there are two major processing procedures: (1) calculating the encrypted summation of distances and (2) calculating the ciphertexts of weighted observation values. In this experiment, we evaluate the running time of each procedure as well as the total running time under different object numbers ranging from 3 to 27. Figure 9(a) shows the running time per iteration for the two procedures, respectively. We can see that the second procedure (i.e., calculating the ciphertexts of weighted observation values) varies more when the object number increases. Figure 9(b) gives the total time of the two procedures in each iteration. When the object number reaches 27, the total running time is only 0.039s, which is sustainable for the phone users. All the results in Figure 9 are averaged values derived from 10 smartphones.

On the cloud server, there are also two major processing procedures in each iteration: (1) updating weights and (2) estimating ground truths. Here, we evaluate the running time of each procedure, and the total running time under different object numbers as well as user numbers, respectively. From Figures 10(a) and 11(a), we can see that most of the time is spent in updating truths for each object. That is also the reason why we need to parallelize the truth updating procedure with MapReduce framework when dealing with massive data. On the other hand, Figures 10(b) and 11(b) demonstrate that the total running time is approximately linear with respect to both object number and the user number.



(a) Running time of each procedure (b) Total running time

Fig. 10. Running time w.r.t. the number of objects for continuous data on the cloud server.



(a) Running time of each procedure (b) Total running time

Fig. 11. Running time w.r.t. number of users for continuous data on the cloud server.

As for the baseline approach CRH, each user just needs to upload his sensory data to the cloud server and then the server conducts truth discovery operations on these data. So, there is no computational cost on the user side. Additionally, since all the calculations of CRH are based on plaintexts and the operations of PPTD are conducted on encrypted data, the computational cost of CRH is less than that of PPTD on the cloud server. However, CRH fails to protect users' private data and reliability information, which will raise the privacy concerns of users.

Communication and Energy Overhead. Compared with the baseline approach CRH, in which each user only needs to communicate with the cloud server once for uploading his sensory data, PPTD introduces more communication overhead due to the incorporated privacy-preserving scheme. To evaluate the communication overhead of PPTD, we measure the number of packets exchanged between the cloud server and all the crowd users. In this experiment, we use the change of the aforementioned objective value in two consecutive iterations as the convergence criterion and the threshold is set as 0.001. Figure 12 shows the numbers of exchanged packets over all users during the whole PPTD procedure, under different user numbers (i.e., K) from 3 to 10. As seen, the overall communication overhead is roughly $O(K)$. Actually, for each user, the average number of messages needed to be exchanged with the cloud server can be roughly calculated by $6(i + 1)$, where i is the number of iterations during the PPTD procedure. Considering that here we set a very conservative threshold, which leads to an average of six iterations (much larger than the usual two or three iterations as shown in Figure 8(a)), the communication overhead is well within the realm of practicality.

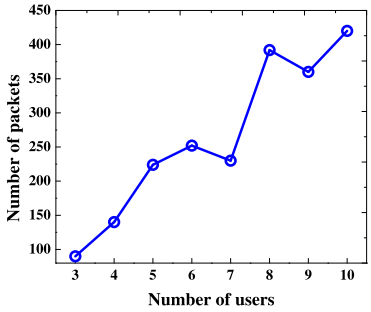


Fig. 12. Communication overhead of PPTD.

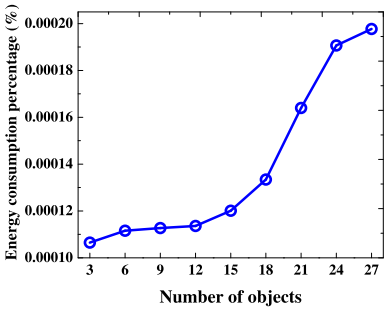


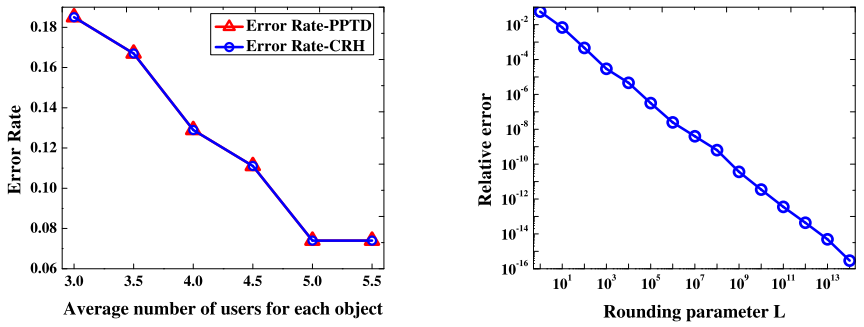
Fig. 13. Energy consumption percentage on smartphones.

The energy overhead on the smartphone of each user is mainly caused by the cipher related operations and data transmissions. For the purpose of evaluating the energy overhead, we measure the average energy consumption percentage (i.e., the energy consumed by PPTD divided by the total energy of the smartphone while it is fully charged) under different object numbers. Figure 13 shows the average energy consumption percentage in one iteration for each user. When the object number reaches 27, the energy consumption percentage is only 0.000198% for each smartphone, which is acceptable for the phone users. The results in Figure 13 are averaged values derived from 10 smartphones in a WiFi network environment.

9.3 Experiment on Crowd Wisdom System

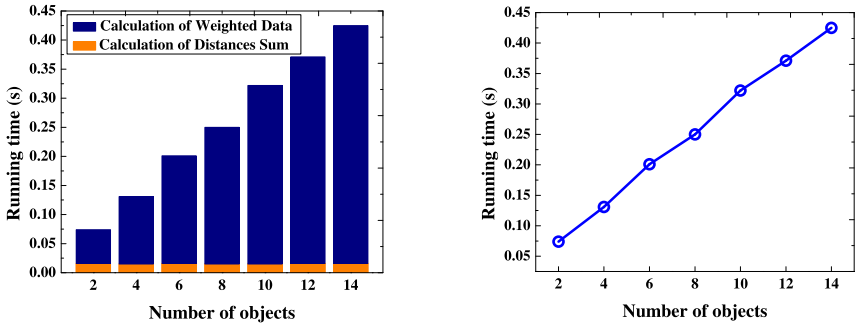
In this part, we evaluate the performance of PPTD on categorical data provided by humans as the sensors. The experiment is conducted on a crowd wisdom system that can integrate the crowd answers and opinions toward a set of questions. We design and implement an Android App through which we can send questions and corresponding candidate answers to the crowd users. Each user who receives the questions can upload his answers to the cloud server. However, the uploaded answers for each question may be conflicted due to various reasons. For example, some users may not have the background knowledge for some specific questions, and different users may have different understandings for the same question. In order to infer the true answer for each question, the cloud server needs to aggregate the answers from different users. To address the concern of some users that their private personal information could be inferred from their answers, we employ PPTD upon this crowd wisdom system, encrypting user answers before they are uploaded to the cloud server. Totally, 113 volunteers are employed as smartphone users and 54 questions are sent to them with candidate answers. For each question, there are four candidate answers and each user who receives this question needs to select one of the candidate answers as the correct answer. We use *Error Rate* as the evaluation metric, and for the sake of evaluation, we have the ground truth answer for each question.

Accuracy and Convergence. Since, in this experiment, each object (i.e., question) is not observed (i.e., answered) by all the users, we use the average number of users observing each object (i.e., the ratio between the number of total answers over the number of total questions) as the tuning variable when evaluating the accuracy of PPTD. The error rates of PPTD and CRH are shown in Figure 14(a), from which we can see that PPTD produces the same error rates as CRH at all times. Moreover, we did not show the error rates with respect to the rounding parameter L, since we find that the final aggregated results are not affected by L. This is because, in this case, the negligible numerical errors introduced by L to the intermediate values are simply not large enough to change the final answers, which are categorical numbers. To evaluate the estimation



(a) Error rate (b) Relative error of weights

Fig. 14. Accuracy of PPTD for categorical data.



(a) Running time of each procedure (b) Total running time

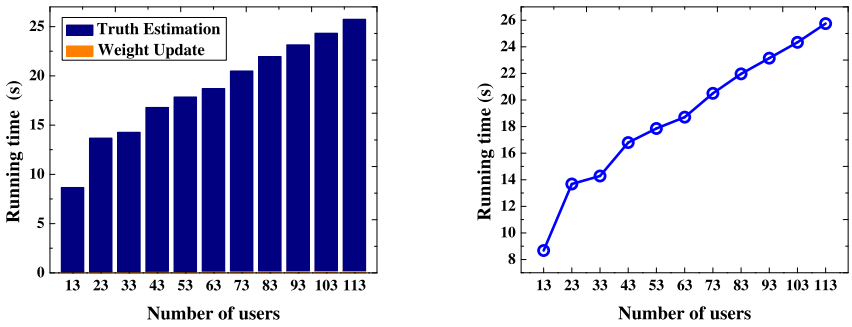
Fig. 15. Running time w.r.t. the number of objects for categorical data on smartphones.

error of user weights, we manually decrypt each user's weight derived by PPTD. Here, we still use the relative error defined in Section 9.2 to measure the errors introduced by L. The results are reported in Figure 14(b), which show that the estimation errors can be ignored if parameter L is large enough. Additionally, we also use the threshold of the change in the estimated ground truths in two consecutive iterations as the convergence criterion, and we find both PPTD and CRH converge within two iterations.

Computational Cost. Next, we evaluate PPTD's computational cost for categorical data. Similar to the experiment on continuous data, the rounding parameter L is also fixed as 10^{10} in this case. Here, we also evaluate the computational cost on user smartphone and the cloud server, respectively.

In particular, we evaluate the two major procedures on a user's smartphone and then give the total running time. In this experiment, the number of the objects observed by each user varies from 2 to 14. The results are shown in Figure 15, from which we can see the second procedure (i.e., calculating weighted data) costs more time than the first procedure (Figure 15(a)). This is because most of the operations in the second procedure are conducted on ciphertexts, while the first procedure is mainly composed of plaintext-based operations. Additionally, Figure 15(b) shows that the largest total running time of the two procedures on a user's smartphone is no more than 0.45s in each iteration, which verifies the practicality of our proposed framework.

To evaluate the computational cost on the cloud server, we vary the number of users from 13 to 113 (the corresponding number of objects varies from 20 to 54 because each question is only answered by part of the users). Figure 16 reports the running time of each procedure and the



(a) Running time of each procedure (b) Total running time

Fig. 16. Running time w.r.t. the number of users for categorical data on the cloud server.

total running time in each iteration. From Figure 16(a), we can see that the computational time of updating truths is far greater than the time of updating weights for all the scenarios, which is similar to that in the experiment for continuous data. The evaluation of total running time in each iteration can be seen in Figure 16(b). We can see the total running time is 25.74s when the number of users is 113. This total time is reasonable, considering the number of crowd users in this experiment is 10 times larger than that in the experiment for continuous data.

9.4 Experiment of Parallel PPTD

From above experimental results, we can see most of the computational time on the cloud server is consumed in updating the ground truths, so we improve PPTD by adapting this procedure to MapReduce framework. In this part, the efficiency of parallel PPTD is verified. Here, we use a Hadoop cluster as the cloud server. The crowd sensing system is simulated with 1,000 users and 1,000 objects, and the observation values are generated through adding Gaussian noise of different intensities to the ground truths. For comparison purpose, we also deploy the basic PPTD framework on the same server.

When evaluating running time under different user numbers and object numbers, we adopt 10 Reducer nodes for parallel PPTD. First, we fix the user number as 500 and change the object number from 100 to 1,000. The running time of parallel PPTD and the basic PPTD in each iteration are shown in Figure 17. From this figure, we can see the parallel PPTD is increasingly more efficient than the basic PPTD as the number of objects goes up. When the object number reaches 1,000, it takes parallel PPTD only 176.48s to complete the two procedures while the basic PPTD would have to spend 709.25s to finish the same computations. Then, we fix the object number as 500 and change the user number from 100 to 1,000. Figure 18 reports the results in this case. Similar patterns can be seen. When the user number reaches 1,000, the parallel PPTD only spends 170.78s to finish the job, much less than the 655.38s consumed by PPTD. All the above results confirm the efficiency of parallel PPTD.

Moreover, it is important to study the effect of the node number in the Hadoop system on the performance of the proposed mechanism. In this experiment, we fix both user number and object number as 500. Figure 19 shows the running time under different numbers of Reducer nodes (results will be similar for the Mapper nodes). As we can see, with the increase of Reducer numbers, the running time decreases rapidly at first and gets flattened very soon. This is because including more Reducer nodes, though improving the parallelism, will introduce more overhead (e.g., communication). Therefore, it is not true that more Reducer nodes would always lead to better performance.

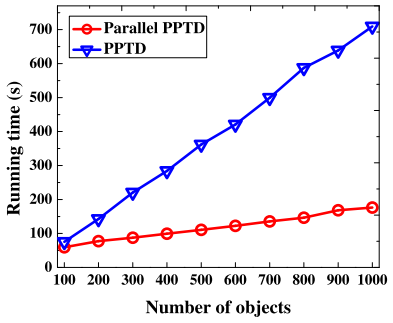


Fig. 17. Running time w.r.t. the number of objects for parallel PPTD.

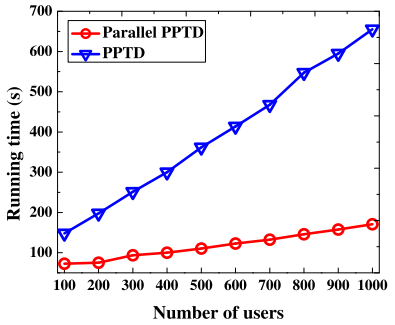


Fig. 18. Running time w.r.t. the number of users for parallel PPTD.

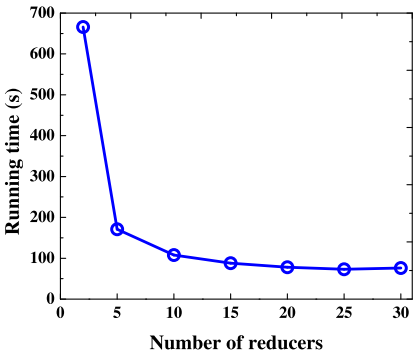


Fig. 19. Running time w.r.t. the number of reducers.

9.5 Experiment of Incremental PPTD

In this section, we evaluate the performance of incremental PPTD scheme on both continuous data and categorical data, which are collected from the crowdsourced indoor floorplan construction system and the crowd wisdom system, respectively. Here, we assume that the objects (i.e., segments in the hallway or questions) are observed in a "streaming" manner, and only one object is observed at each timestamp. The baseline method we adopted is PPTD, i.e., we conduct the PPTD scheme once again on the whole dataset whenever a new object is observed. The rounding parameter L is still set as 10^{10} and we use I-PPTD to denote the proposed incremental PPTD scheme.

9.5.1 Performance on Crowdsourced Indoor Floorplan Construction System. We first compare the accuracy and computational cost of the incremental PPTD scheme with that of the baseline method (i.e., PPTD). In order to measure the estimation error of the proposed scheme, we fix the number of users as 10 and calculate MAE and RMSE for the estimated truths of the 27 objects, and the results are shown in Table 1. Although the estimation error of I-PPTD is a little higher than that of PPTD, the computational cost of I-PPTD is much less than that of the baseline method, which can be seen from Figures 20 and 21.

Figure 20 shows the average running time for each object on the cloud server when the number of users varies from 3 to 10. Here, we conduct PPTD with 10 iterations in order to guarantee the convergence. From this figure, we can see the average running time of I-PPTD is much less than that of PPTD on the cloud server, especially when the number of users becomes large. This is mainly because PPTD needs to revisit the old data and be conducted on all observed objects

Table 1. Accuracy of I-PPTD vs. PPTD for Continuous Data

	MAE	RMSE
I-PPTD	0.7354	0.8342
PPTD	0.7170	0.8218

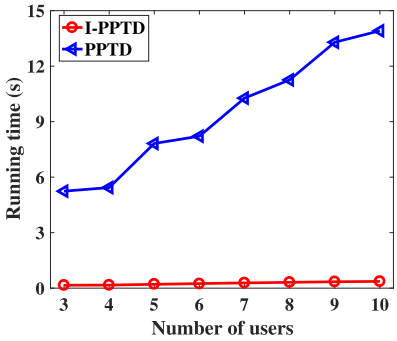


Fig. 20. Running time of I-PPTD and PPTD w.r.t. the number of users for continuous data on the cloud server.

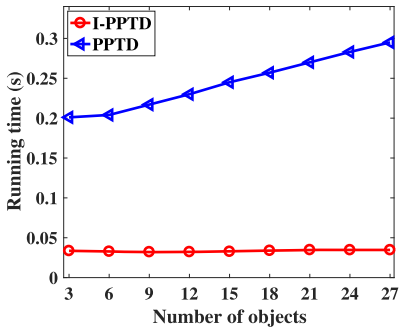


Fig. 21. Running time of I-PPTD and PPTD w.r.t. the number of objects for continuous data on smartphone.

whenever a new object is observed, while I-PPTD only needs to estimate the truth of the new observed object. Additionally, at each timestamp, PPTD is iteratively conducted until convergence while I-PPTD only needs to be conducted once. Figure 21 reports the average running time for each object on a smartphone while the number of objects is varying from 3 to 27. Here, we also conduct PPTD with 10 iterations. This figure shows that the average running time of I-PPTD on smartphones is much less that of PPTD. The reason is similar with the above. Additionally, the averaging running time of I-PPTD is almost the same while that of PPTD is increasing when the number of objects varies from 3 to 27. The reason is that PPTD needs to estimate the truths of all the observed objects while I-PPTD only needs to estimate the truth for the newly observed object at each timestamp.

In this experiment, we also evaluate the convergence of the I-PPTD scheme. Here, we manually decrypt user weights and report the results at different timestamps. Figure 22 shows the weights of five randomly selected users when the timestamp is varying from 1 to 27. From this figure, we can see users' weights gradually become stable as the number of timestamps increases, which means the proposed I-PPTD scheme can guarantee convergence when sufficient objects are observed. To further illustrate this point, we also compare the weights of all users calculated by I-PPTD with those calculated by PPTD. The results are shown as Figure 23. Here, we report user weights calculated by I-PPTD at timestamp 5, 15, and 25. We also report the weight values calculated by the PPTD scheme after all the objects are observed. From the result, we can see that although user weights calculated by I-PPTD deviate from the baseline values at the first few timestamps, they gradually converge to the values calculated by PPTD as time goes on. That is to say, I-PPTD can achieve similar accuracy with PPTD when sufficient objects are observed.

9.5.2 Performance on Crowd Wisdom System. In this part, we evaluate the performance of I-PPTD on categorical data. We first evaluate the accuracy of I-PPTD by calculating the error rate of the estimated truths. The number of users is fixed as 113. After observing the 54 questions, we find

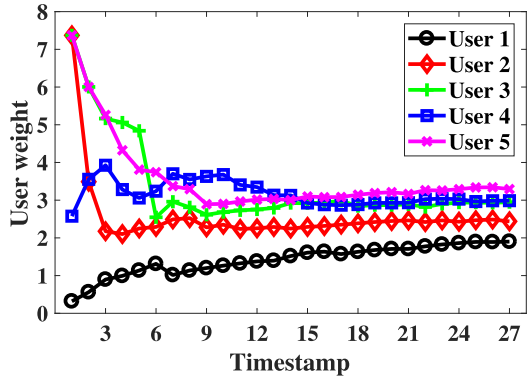


Fig. 22. Convergence of I-PPTD on continuous data.

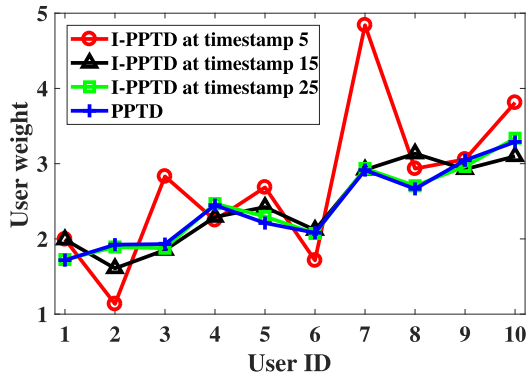


Fig. 23. User weights calculated by I-PPTD and PPTD.

both the error rates of I-PPTD and PPTD are 0.074, which verifies that I-PPTD could guarantee high accuracy while protecting users' privacy on streaming categorical data.

Similar to the experiment on continuous data, we also evaluate the computational cost of I-PPTD on categorical data and compare it with PPTD. In order to guarantee the convergence, we conduct PPTD with *two* iterations whenever a new object is observed in this experiment. We first evaluate the computational cost of I-PPTD on the cloud server. We vary the number of users from 13 to 113 and calculate the average running time for each newly observed object. The results are shown as Figure 24, from which we can see the computational cost of I-PPTD is much less than that of PPTD as the number of users increases. The reason is similar to that for continuous data. Then, we evaluate the computational cost of I-PPTD on the user side. We vary the number of objects observed by each user from 2 to 14 and calculate the average running time for each newly observed object on a smartphone. Figure 25 reports the results, from which we can see the computational cost of I-PPTD on the user side is also much less than that of PPTD. The experimental results on categorical data further verify that the I-PPTD scheme is much more efficient than the PPTD scheme when the data is collected in a streaming manner.

In order to evaluate the convergence of I-PPTD on categorical data, we randomly select five users and manually decrypt their weights at different timestamps. The result is shown as Figure 26, from which we can see the weight of each user is gradually converging to stable as the number

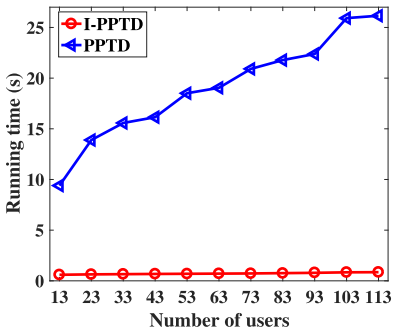


Fig. 24. Running time of I-PPTD and PPTD w.r.t. the number of users for categorical data on the cloud server.

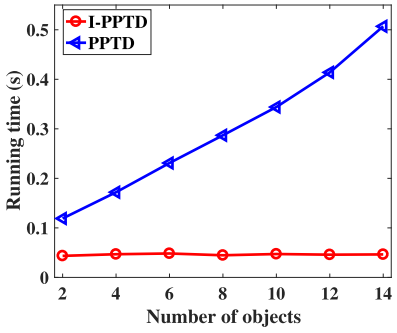


Fig. 25. Running time of I-PPTD and PPTD w.r.t. the number of objects for categorical data on smartphones.

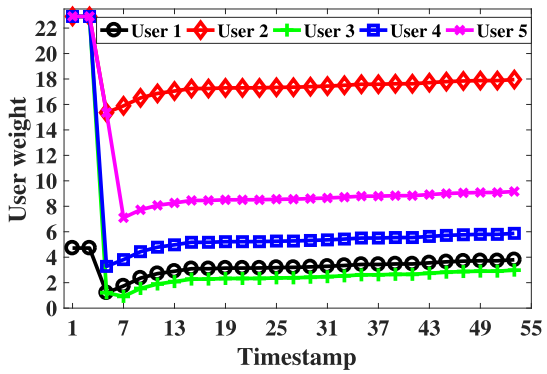


Fig. 26. Convergence of I-PPTD on categorical data.

of observed questions increases. This confirms that the proposed I-PPTD scheme can guarantee convergence when sufficient objects are observed.

10 RELATED WORK

As an effective technique to extract reliable information from crowd sensing systems, truth discovery has drawn more and more attention [25, 32–37, 39, 40, 51, 56–59] in recent years. Representative truth discovery schemes include AccuSim [34], CRH [33], TruthFinder [59], and the like. Compared with the naive averaging or voting approaches, these schemes can provide more reliable aggregated results by estimating and incorporating user reliability into the aggregation process. However, none of these schemes take actions to protect user privacy, which is a key concern in many crowd sensing systems [14].

The importance of privacy protection has long been recognized in many fields [7, 23, 42]. The representative strategies to tackle various privacy concerns include (1) anonymization [7, 44, 53], which removes identification information from all the interactions between the participant and other entities; (2) data perturbation [28, 29], which achieves privacy protection by adding artificial noise to the data before sharing them with others; and, (3) the approaches based on cryptography or secure multi-party computation [17, 26], in which the sensitive data are encrypted and, in many cases, the parties need to cooperate with each other to decrypt the final results.

Recently, the privacy-preserving problem is also studied with respect to crowd sensing applications. For example, Refs. [24, 30, 49, 50] present anonymization-based schemes to protect user's private information from being disclosed. Although these schemes can guarantee the users' privacy in some cases, they are not suitable for truth discovery scenarios, where, instead of the anonymity of each user, what we need to preserve is the confidentiality of his observation values from which sensitive personal information (including user identity) may be inferred. Moreover, some perturbation-based methods are also proposed [13, 43, 45, 55, 60]. However, it is difficult to integrate these schemes with truth discovery approaches because the artificial noise added to each user's data would make it difficult to accurately estimate his reliability. Thus, cryptography based schemes are good choices, as they can guarantee the confidentiality of the observation values without introducing additional noise. Since some computations need to be conducted on encrypted data in a truth discovery procedure, such schemes should have homomorphic properties [11]. Recently, the fully homomorphic encryption scheme [16] has drawn much attention due to the ability of taking arbitrary computations on encrypted data, but the prohibitively high computation cost makes it impractical to be used in crowd sensing applications. Although our proposed scheme is based on the traditional Paillier cryptosystem, which cannot conduct arbitrary computations over encrypted data, we use it in a novel manner that well captures the specific algebra operations in truth discovery procedures without significant overhead. Additionally, Ref. [27] proposes a homomorphic encryption-based approach to protect user privacy in crowdsourcing applications. However, it addresses a different scenario in which users can know their own reliability information. Also, Ref. [27] mainly focuses on categorical data. In contrast, our scheme can deal with not only categorical data but also other data types.

11 CONCLUSIONS

In this article, we design a novel PPTD framework to tackle the issue of privacy protection in crowd sensing systems. The key idea of PPTD is to perform weighted aggregation on the encrypted data of users using a homomorphic cryptosystem and iteratively conducting two phases (i.e., secure weight update and secure truth estimation) until convergence. During this procedure, both user's observation values and his reliability score are protected. In order to process large-scale data efficiently, a parallelized extension of PPTD is also proposed based on the MapReduce framework. Additionally, we design an incremental PPTD scheme to deal with the scenarios where the sensing data of different objects are collected in a streaming manner. Theoretical analysis demonstrates that the observation values of each user will not be disclosed to others and the weight of each user will not be known by any party based on the proposed framework. Through extensive experiments on both real-world crowd sensing systems and synthetic data, we demonstrate that the proposed framework can not only generate accurate aggregated results but also guarantee the introduced computational and communication overhead are within the realm of practicality.

REFERENCES

[1] Si Chen, Muyuan Li, Kui Ren, Xinwen Fu, and Chunming Qiao. 2015. Rise of the indoor crowd: Reconstruction of building interior view via mobile crowdsourcing. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15)*.

[2] Si Chen, Muyuan Li, Kui Ren, and Chunming Qiao. 2015. Crowd map: Accurate reconstruction of indoor floor plans from crowdsourced sensor-rich videos. In *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems (ICDCS'15)*.

[3] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, Yilong Li, Ji Jia, and Xiaofan Jiang. 2014. AirCloud: A cloud-based air-quality monitoring system for everyone. In *Proceedings of the 12th ACM Conference on Embedded Networked Sensor Systems (SenSys'14)*.

[4] Yohan Chon, Yunjong Kim, and Hojung Cha. 2013. Autonomous place naming system using opportunistic crowdsensing and knowledge from crowdsourcing. In *Proceedings of the 12th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'13)*.

[5] Yohan Chon, Nicholas D. Lane, Yunjong Kim, Feng Zhao, and Hojung Cha. 2013. Understanding the coverage and scalability of place-centric crowdsensing. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'13)*.

[6] Yohan Chon, Nicholas D. Lane, Fan Li, Hojung Cha, and Feng Zhao. 2012. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp'12)*.

[7] Chi-Yin Chow, Mohamed F. Mokbel, and Tian He. 2010. A privacy-preserving location monitoring system for wireless sensor networks. *IEEE Transactions on Mobile Computing*1 (2010), 94–107.

[8] Ronald Cramer, Ivan Damgård, and Jesper B. Nielsen. 2001. *Multiparty Computation from Threshold Homomorphic Encryption*. Springer.

[9] Ivan Damgård and Mads Jurik. 2001. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography*. Springer.

[10] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: Simplified data processing on large clusters. *Communicatons of the ACM* 51, 1 (2008), 107–113.

[11] Caroline Fontaine and Fabien Galand. 2007. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security* 1 (2007), 1–10.

[12] Raghu K. Ganti, Nam Pham, Hossein Ahmadi, Saurabh Nangia, and Tarek F. Abdelzaher. 2010. GreenGPS: A participatory sensing fuel-efficient maps application. In *Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'10)*.

[13] Raghu K. Ganti, Nam Pham, Yu-En Tsai, and Tarek F. Abdelzaher. 2008. PoolView: Stream privacy for grassroots participatory sensing. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems (SenSys'08)*.

[14] Raghu K. Ganti, Fan Ye, and Hui Lei. 2011. Mobile crowdsensing: Current state and future challenges. *IEEE Communications Magazine* 49, 11 (2011), 32–39.

[15] Ruipeng Gao, Mingmin Zhao, Tao Ye, Fan Ye, Yizhou Wang, Kaigui Bian, Tao Wang, and Xiaoming Li. 2014. Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In *Proceedings of the 20th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'14)*.

[16] Craig Gentry. 2009. *A Fully Homomorphic Encryption Scheme*. Ph.D. Dissertation. Stanford University.

[17] Oded Goldreich. 1998. Secure Multi-party Computation. *Manuscript. Preliminary Version* (1998).

[18] "GSK". 2016. Can an iPhone transform the way we monitor and improve patient health? Retrieved from http://www.gsk.com/en-gb/behind-the-science/innovation/can-an-iphone-transform-the-way-we-monitor-and-improve-patient-health/.

[19] Shaohan Hu, Hengchang Liu, Lu Su, Hongyan Wang, Tarek F. Abdelzaher, Pan Hui, Wei Zheng, Zhiheng Xie, John Stankovic, et al. 2014. Towards automatic phone-to-phone communication for vehicular networking applications. In *Proceedings of the 33rd Annual IEEE International Conference on Computer Communications (INFOCOM'14)*.

[20] Shaohan Hu, Lu Su, Shen Li, Shiguang Wang, Chenji Pan, Siyu Gu, T. Amin, Hengchang Liu, Suman Nath, Romit Roy Choudhury, et al. 2015. Experiences with eNav: A low-power vehicular navigation system. In *Proceedings of the 2015 ACM Conference on Ubiquitous Computing (UbiComp'15)*.

[21] Shaohan Hu, Lu Su, Hengchang Liu, Hongyan Wang, and Tarek F. Abdelzaher. 2013. SmartRoad: A crowd-sourced traffic regulator detection and identification system. In *Proceedings of the 12th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'13)*.

[22] Shaohan Hu, Lu Su, Hengchang Liu, Hongyan Wang, and Tarek F. Abdelzaher. 2015. SmartRoad: Smartphone-based crowd sensing for traffic regulator detection and identification. *ACM Transactions on Sensor Networks (TOSN)* 11, 4 (2015), 55.

[23] Kuan Lun Huang, Salil S. Kanhere, and Wen Hu. 2009. Towards privacy-sensitive participatory sensing. In *Proceedings of the 7th IEEE International Conference on Pervasive Computing and Communications (PerCom 2009)*.

[24] Kuan Lun Huang, Salil S. Kanhere, and Wen Hu. 2012. A privacy-preserving reputation system for participatory sensing. In *LCN*.

[25] Wenjun Jiang, Chenglin Miao, Lu Su, Qi Li, Shaohan Hu, ShiGuang Wang, Jing Gao, Hengchang Liu, Tarek Abdelzaher, Jiawei Han, et al. 2018. Towards quality aware information integration in distributed sensing systems. *IEEE Transactions on Parallel and Distributed Systems* 29, 1 (2018), 198–211.

[26] Taeho Jung, XuFei Mao, Xiang-Yang Li, Shao-Jie Tang, Wei Gong, and Lan Zhang. 2013. Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation. In *Proceedings of the 32nd Annual IEEE International Conference on Computer Communications (INFOCOM'13)*.

[27] Hiroshi Kajino, Hiromi Arai, and Hisashi Kashima. 2014. Preserving worker privacy in crowdsourcing. *Data Mining and Knowledge Discovery* 28, 5–6 (2014), 1314–1335.

[28] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. 2003. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM'03)*.

[29] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. 2005. Random-data perturbation techniques and privacy-preserving data mining. *Knowledge and Information Systems* 7, 4 (2005), 387–414.

[30] Leyla Kazemi and Cyrus Shahabi. 2011. A privacy-aware framework for participatory sensing. *ACM SIGKDD Explorations Newsletter* 13, 1 (2011), 43–51.

[31] Nicholas D. Lane, Yohan Chon, Lin Zhou, Yongzhe Zhang, Fan Li, Dongwon Kim, Guanzhong Ding, Feng Zhao, and Hojung Cha. 2013. Piggyback crowdsensing (PCS): Energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (Sensys'13)*.

[32] Qi Li, Yaliang Li, Jing Gao, Lu Su, Bo Zhao, Murat Demirbas, Wei Fan, and Jiawei Han. 2014. A confidence-aware approach for truth discovery on long-tail data. *Proceedings of the VLDB Endowment* 8, 4 (2014), 425–436.

[33] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei Han. 2014. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD'14)*.

[34] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, and Divesh Srivastava. 2012. Truth finding on the deep web: Is the problem solved? *Proceedings of the VLDB Endowment* 6, 2 (2012), 97–108.

[35] Yaliang Li, Jing Gao, Chuishi Meng, Qi Li, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. A survey on truth discovery. *ACM SIGKDD Explorations Newsletter* 17, 2 (2016), 1–16.

[36] Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2015. On the discovery of evolving truth. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'15)*.

[37] Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. 2016. Conflicts to harmony: A framework for resolving conflicts in heterogeneous data by truth discovery. *IEEE Transactions on Knowledge and Data Engineering* 28, 8 (2016), 1986–1999.

[38] Yehuda Lindell and Benny Pinkas. 2000. Privacy preserving data mining. In *Proceedings of the 20th Annual International Cryptology Conference (CRYPTO'00)*.

[39] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei Han. 2015. FaitCrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'15)*.

[40] Chuishi Meng, Wenjun Jiang, Yaliang Li, Jing Gao, Lu Su, Hu Ding, and Yun Cheng. 2015. Truth discovery on crowd sensing of correlated entities. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15)*.

[41] Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin, Houping Xiao, Jing Gao, and Kui Ren. 2015. Cloud-enabled privacy-preserving truth discovery in crowd sensing systems. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15)*.

[42] Sangho Oh, Tam Vu, Marco Gruteser, and Suman Banerjee. 2012. Phantom: Physical layer cooperation for location privacy protection. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM'12)*.

[43] Nam Pham, Raghu K. Ganti, Yusuf S. Uddin, Suman Nath, and Tarek Abdelzaher. 2010. Privacy-preserving reconstruction of multidimensional data maps in vehicular participatory sensing. In *Wireless Sensor Networks*. Springer.

[44] Layla Pournajaf, Li Xiong, Daniel A. Garcia-Ulloa, and Vaidy Sunderam. 2014. A survey on privacy in mobile crowd sensing task management. *Tech. Rep. TR-2014–002* (2014).

[45] Daniele Quercia, Ilias Leontiadis, Liam McNamara, Cecilia Mascolo, and Jon Crowcroft. 2011. SpotME if you can: Randomized responses for location obfuscation on mobile phones. In *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS'11)*.

[46] Moo-Ryong Ra, Bin Liu, Tom F. La Porta, and Ramesh Govindan. 2012. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'12)*.

[47] Kiran K. Rachuri, Cecilia Mascolo, Mirco Musolesi, and Peter J. Rentfrow. 2011. Sociablesense: Exploring the trade-offs of adaptive sampling and computation offloading for social sensing. In *Proceedings of the 17th Annual ACM International Conference on Mobile Computing and Networking (MobiCom'11)*.

[48] Fatemeh Saremi, Omid Fatemieh, Hossein Ahmadi, Hongyan Wang, Tarek Abdelzaher, Raghu Ganti, Hengchang Liu, Shaohan Hu, Shen Li, and Lu Su. 2016. Experiences with greengps-fuel-efficient navigation using participatory sensing. *IEEE Transactions on Mobile Computing* 15, 3 (2016), 672–689.

[49] Katie Shilton. 2009. Four billion little brothers? Privacy, mobile phones, and ubiquitous data collection. *Commun. ACM* 52, 11 (2009), 48–53.

[50] Minho Shin, Cory Cornelius, Dan Peebles, Apu Kapadia, David Kotz, and Nikos Triandopoulos. 2011. AnonySense: A system for anonymous opportunistic sensing. *Pervasive and Mobile Computing* 7, 1 (2011), 16–30.

[51] Lu Su, Qi Li, Shaohan Hu, Shiguang Wang, Jing Gao, Hengchang Liu, Tarek F. Abdelzaher, Jiawei Han, Xue Liu, Yan Gao, et al. 2014. Generalized decision aggregation in distributed sensing systems. In *Proceedings of the 35th IEEE International Conference on Real-Time Systems Symposium (RTSS'14)*.

[52] Vigneshwaran Subbaraju, Amit Kumar, Vikrant Nandakumar, Sonali Batra, Salil Kanhere, Pradipta De, Vinayak Naik, Dipanjan Chakraborty, and Archan MISRA. 2013. ConferenceSense: A case study of sensing public gatherings using participatory smartphones. In *Proceedings of the International Workshop on Pervasive Urban Crowdsensing Architecture and Applications (PUCAA'13)*.

[53] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 557–570.

[54] "TechCrunch". 2016. Transit app begins crowdsourcing data to provide accurate, real-time info. Retrieved from https://techcrunch.com/2016/12/20/transit-app-begins-crowdsourcing-data-to-provide-accurate-real-time-info/.

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

[56] Dong Wang, Lance Kaplan, Hieu Le, and Tarek Abdelzaher. 2012. On truth discovery in social sensing: A maximum likelihood estimation approach. In *Proceedings of the 11th ACM International Conference on Information Processing in Sensor Networks (IPSN'12)*.

[57] Shiguang Wang, Lu Su, Shen Li, Shaohan Hu, Tanvir Amin, Hongwei Wang, Shuochao Yao, Lance Kaplan, and Tarek Abdelzaher. 2015. Scalable social sensing of interdependent phenomena. In *Proceedings of the 14th ACM International Conference on Information Processing in Sensor Networks (IPSN'15)*.

[58] Shiguang Wang, Dong Wang, Lu Su, Lance Kaplan, and Tarek F. Abdelzaher. 2014. Towards cyber-physical systems in social spaces: The data reliability challenge. In *Proceedings of the 35th IEEE International Conference on Real-Time Systems Symposium (RTSS'14)*.

[59] Xiaoxin Yin, Jiawei Han, and Philip S. Yu. 2008. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering* 20, 6 (2008), 796–808.

[60] Fan Zhang, Li He, Wenbo He, and Xue Liu. 2012. Data perturbation with state-dependent noise for participatory sensing. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM'12)*.

Received September 2017; revised April 2018; accepted September 2018