Cloud-Enabled Privacy-Preserving Truth Discovery in Crowd Sensing Systems

Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin
Houping Xiao, Jing Gao, Kui Ren
Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, New York 14260
{cmiao, wenjunji, lusu, yaliangl, suxinguo, zhanqin, houpingx, jing, kuiren}@buffalo.edu

ABSTRACT

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 identify truthful values from the crowd sensing data, the topic of truth discovery, whose goal is to estimate user quality and infer truths through quality-aware data aggregation, has drawn significant attention. Though able to improve aggregation accuracy, existing truth discovery approaches fail to take into consideration an important issue in their design, i.e., the protection of individual users' private information. In this paper, we propose a novel cloud-enabled privacy-preserving truth discovery (PPTD) framework for crowd sensing systems, which can achieve the protection of 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 homomorphic cryptosystem. In order to deal with large-scale data, we also propose to parallelize PPTD with MapReduce framework. Through extensive experiments on not only synthetic data but also real world crowd sensing systems, we justify the guarantee of strong privacy and high accuracy of our proposed framework.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

General Terms

Design, Human Factors, Security

Keywords

Crowd Sensing; Truth Discovery; Privacy; Cloud

SenSys'15, November 1–4, 2015, Seoul, South Korea.
© 2015 ACM. ISBN 978-1-4503-3631-4/15/11 ...$15.00.
DOI: http://dx.doi.org/10.1145/2809695.2809719.

1. INTRODUCTION

The recent proliferation of increasingly capable and affordable mobile devices (e.g., smartphones, smartwatches, smartglasses, etc.) packed with a plethora of on-board sensors (e.g., GPS, accelerometer, compass, camera, etc.) has given rise to crowd sensing, a newly-emerged sensing paradigm where the collection of sensory data are outsourced to a crowd of users participating in the sensing task. Recently, a large variety of crowd sensing systems [3–6, 12, 18–21, 28, 39–41, 45] 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, health-care, 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* [29–32, 34, 35, 44, 47–50], i.e., to discover truthful facts from unreliable user information, have 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, etc.) 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. Towards this end, we propose a *cloud-enabled privacy-preserving truth discovery* (PPTD) framework. This framework makes use of 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 in the meanwhile, the weight information are not disclosed to any party.

Additionally, in order to deal with massive data, we also design a parallel privacy-preserving truth discovery scheme using the MapReduce framework [10], and thus the privacy-preserving truth discovery procedure can be conducted in a parallel and distributed manner.

In summary, our contributions in this paper are:

- We propose a novel cloud-enabled privacy-preserving truth discovery framework for crowd sensing systems, which can accurately aggregate sensory data while protecting both user observation and user reliability from being disclosed.
- 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.
- We conduct extensive experiments on both real crowd sensing systems and synthetic data set. The results validate the claim that our proposed framework can generate accurate aggregation results while protecting the privacy of user data and weight.

In the remaining parts of this paper, we first define the problem in Section 2 and give preliminary in Section 3. Then the proposed privacy-preserving truth discovery framework and the parallel mechanism are presented in Section 4. We analyze the privacy of our proposed framework and discuss it in Section 5 and 6 respectively. In Section 7, we conduct a series of experiments to demonstrate the claims given in this paper. We discuss related work in Section 8 and conclude the paper in Section 9.

2. PROBLEM DEFINITION

In this section, we describe the problem settings of our proposed privacy-preserving truth discovery 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 which collects user data and conduct 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 to maliciously manipulate the data price in the scenarios where crowd users trade their data with the could 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 6.

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

Suppose there are K users, denoted as $\mathcal{K} = \{1, 2, \cdots, K\}$, and a cloud server $\mathcal{S}$ that released M objects represented as $\mathcal{M} = \{1, 2, \cdots, M\}$. 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. Our goal is to let server $\mathcal{S}$ accurately estimate the ground truths $\{x_m^*\}_{m=1}^M$ 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 privacy-preserving truth discovery 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

Towards 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 which is close to the estimated ground truths. Typically, the user weights are calculated as follows:

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

where f is a monotonically decreasing function, and $d(\cdot)$ is the distance function which can measure the difference between users' observation values and the estimated ground truths. In this paper, we adopt the weight calculation function of CRH [30] 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 paper, 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:

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

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 $x_m^k = (0, ..., \underset{q}{1}, ..., 0)^T$ 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 truth vector x_m^*:

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

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., Eqn. (2));
    end
    for each object $m$ do
        Update the estimated ground truth based on
        current weights (e.g., Eqn. (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 privacy-preserving truth discovery framework, an additive homomorphic asymmetric cryptosystem is adopted. As 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 $\otimes$ which 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 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, an 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 Equation (6), (7) and (8), the homomorphic properties of this cryptosystem can be described as

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

$$E_{pk}(a \cdot m_1) = E_{pk}(m_1)^a = g^{am_1} {r_1}^{an} mod\ n^2 \tag{10}$$

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

In this paper, the (p, t)-threshold Paillier cryptosystem is adopted, in which the private key sk is divided (denoted as sk_1, sk_2, $\cdots$, 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 [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 privacy-preserving truth discovery (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).

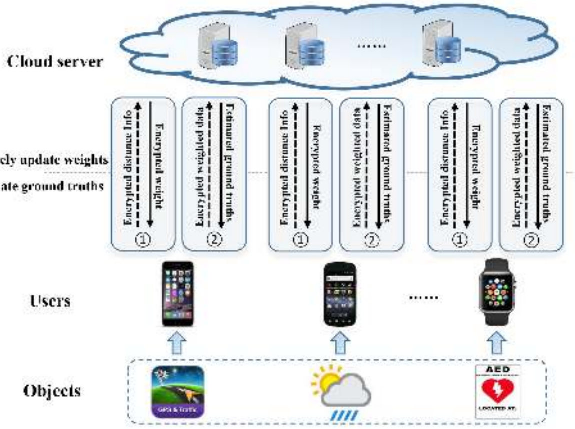


Figure 1: Privacy-preserving truth discovery framework

As shown in Fig. 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 firstly 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 Eqn. (2) and Eqn. (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 Eqn. (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 $\mathcal{C} = E_{pk}(\sum_{k=1}^{K} v_k) = \prod_{k=1}^{K} E_{pk}(v_k)$
  based on Eqn. (9);
3 Server $\mathcal{S}$ randomly selects $t-1$ users and sends $\mathcal{C}$ to
  them;
4 Each selected user $k'$ calculates the partial decryption
  $\mathcal{C}_{k'}$ of $\mathcal{C}$ based on Eqn. (11) and sends $\mathcal{C}_{k'}$ to the cloud
  server;
5 Server $\mathcal{S}$ calculates its partial decryption $\mathcal{C}_{\mathcal{S}}$ 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 use $\widetilde{h}$ to denote the rounded integer of h and other values in this paper 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 Eqn.(2), the encrypted weight can be updated as follows

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

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 Fig. 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 date are different, we need to consider them separately when calculating distances. For categorical data, user k can easily calculate distances based on Eqn. (4). But for continuous data, we need to know the standard deviation std_m according to Eqn. (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 Fig. 2) for all the data types to update user's weight, and then specifically discuss the calculation of std_m for continuous data (**W2**, **W3**, **W4** and **W5** in Fig. 2).

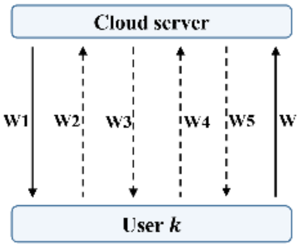


Figure 2: Secure weight update for user k

Step W1. 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 Eqn. (8).

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

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

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

Step W6. After the encryption in 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 Eqn. (8) as

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

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

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

As for continuous data, as discussed previously, the standard deviation std_m should be firstly 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 Eqn. (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 Fig. 3, there are two major steps in this phase, which are detailed as follows.

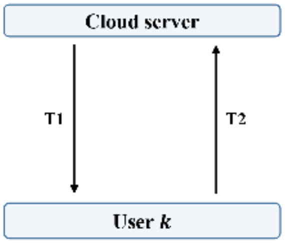


Figure 3: Secure Truth Estimation

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 Eqn. (10) using the following formula:

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

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 Eqn. (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 get 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 Eqn. (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 Eqn. (5).

Please note that the ground truths estimated in this step for categorical data are probability values, which 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 privacy-preserving truth discovery 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 $\log \widetilde{Dist_k}$)
  with parameter $L$. Then user $k$ encrypts them as
  $E_{pk}(\widetilde{Dist_k})$, $E_{pk}(\log \widetilde{Dist_k})$ 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 Eqn. (15) and Eqn. (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 Eqn. (17) and Eqn. (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 step 2∼6 until the convergence criterion is
  satisfied and then output $\{x_m^*\}_{m=1}^{M}$;
```

4.3 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 kind of 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 re-organized into key/value pairs, where the key is the ID of each object (i.e., m), and the value is the rest 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
    $[m, [E_{pk}(\widetilde{w}_k \cdot \widetilde{x}_m^k), k]]$;
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;
```

5. PRIVACY ANALYSIS

In this paper, our proposed framework is based on two assumptions: all the parties are semi-honest [33] and there is no collusion among them. The semi-honest model assumes that all the parties are honest but curious, which means they 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. The non-collusion model assumes the parties in our framework have no collusions, which means they will not collude with each other outside the 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 they know each other they are probably not willing to disclose private information to others.

As previously discussed, the security threats mainly comes 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 in the meanwhile, 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 1, followed by the proof.

Theorem 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 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. Firstly, 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 privacy-preserving truth discovery 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 privacy-preserving truth discovery 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 privacy-preserving truth discovery 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. □

6. 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 appropriate L, which is shown in Section 7.

Another issue we are concerned 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 paper, as long as at least $t - 1$ users could upload their data in time, the privacy-preserving truth discovery 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 Eqn. (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**).

7. PERFORMANCE EVALUATION

In this section, we evaluate the proposed privacy-preserving truth discovery (PPTD) framework. Experiment results on both real world crowd sensing systems and synthetic data set are presented and discussed.

7.1 Experiment Setup

In this paper, 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) to measure the mean of absolute distance between the estimated results and ground truths.
- $RMSE$: For continuous data, we also use the root of mean squared error ($RMSE$) 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 [30], 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 privacy-preserving truth discovery framework, we use a Dell Hadoop cluster with Intel Xeon E5-2403 processor (4x 1.80 GHz, 48 GB RAM) as the "cloud".

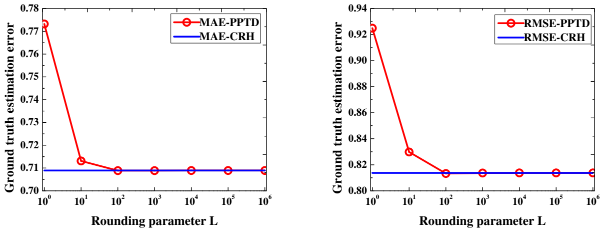
7.2 Experiment on Crowdsourced Indoor Floorplan Construction System

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 indoor floorplan from sensory data (e.g., the readings of compass, accelerometer, gyroscope, etc.) 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 location points in the hallway. We develop an Android App which 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.

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

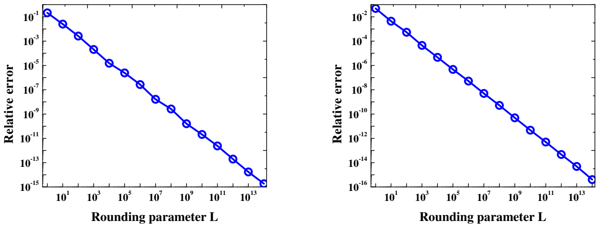
Accuracy. We first compare the accuracy of the final estimated ground truths between PPTD and the baseline approach (i.e., CRH). Since the estimation errors of PPTD are introduced by the rounding parameter L, we vary L from 10^0 to 10^6 and measure the corresponding accuracy. In the experiment, we randomly 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.



(a) MAE (b) RMSE

Figure 4: Ground truth estimation errors under different values of the parameter L

Figure 4 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 Fig. 5.

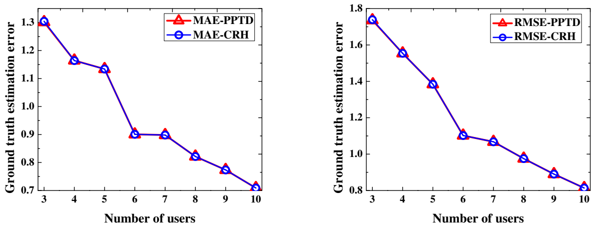


(a) Relative error of weight (b) Relative error of estimated truth

Figure 5: Relative errors under different values of L

As shown in Fig. 5(a) and 5(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 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) MAE (b) RMSE

Figure 6: Ground truth estimation errors under different number of users

Figure 6(a) and 6(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 privacy-preserving truth discovery procedure. In this experiment, the rounding parameter L is still set as 10^{10}. Figure 7 shows the evolution of 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., $\sum_{k=1}^{K} w_k \sum_{m=1}^{M} d(x_m^k, x_m^*)$). Here we repeat the experiment for 5 times with different random initialization values. As we can see, all the objective values, although under different initializations, converge quickly within just a few iterations.

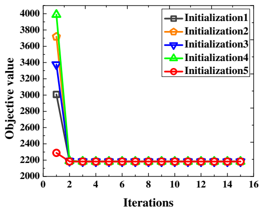


Figure 7: Convergence w.r.t Iterations

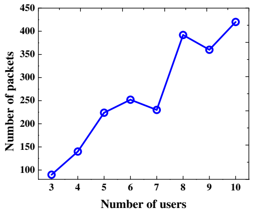
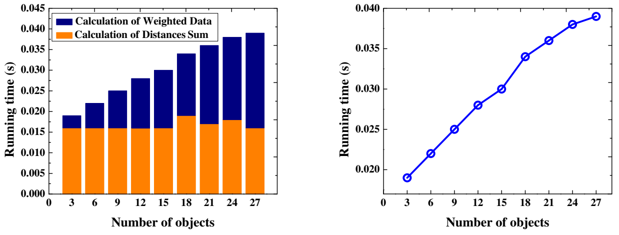


Figure 8: Communication overhead

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 Fig. 9 are averaged values derived from 10 smartphones.

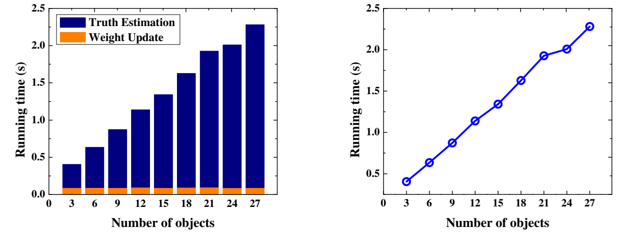


(a) Running time of each procedure

(b) Total running time

Figure 9: Running time w.r.t Number of objects for continuous data on smartphone

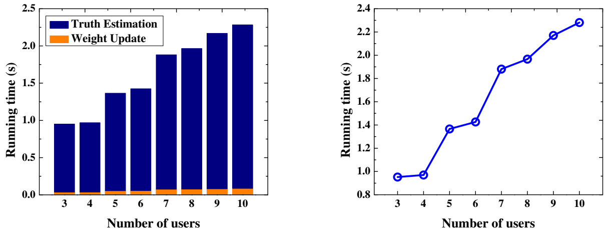
On the cloud server, there are also two major processing procedures in each iteration: 1) updating weights, and 2) estimating ground truths. Here we evaluate the running time of each procedure, and the total running time under different object numbers as well as user numbers, respectively. From Fig. 10(a) and Fig. 11(a), we can see that most of the time is spent in updating truth 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, Fig. 10(b) and Fig. 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

Figure 10: Running time w.r.t Number of objects for continuous data on the cloud server



(a) Running time of each procedure

(b) Total running time

Figure 11: Running time w.r.t Number of users for continuous data on the cloud server

Communication and Energy Overhead. To evaluate the communication overhead in the privacy-preserving truth discovery procedure, 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 8 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 average 6 iterations (much larger than the usual 2 or 3 iterations as shown in Fig. 7), the communication overhead is well within the realm of practicality.

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 12 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 Fig. 12 are averaged values derived from 10 smartphones in WiFi network environment.

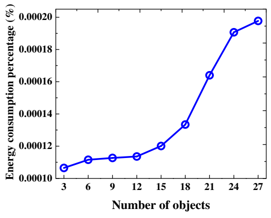


Figure 12: Energy consumption percentage on smartphone

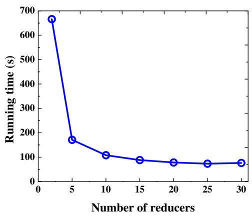
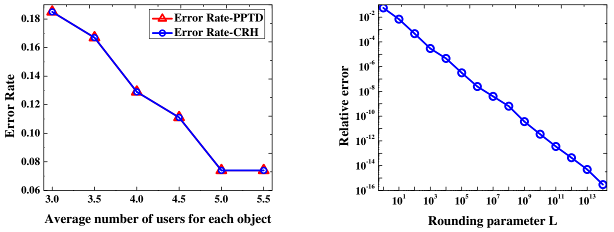


Figure 13: Running time w.r.t Number of reducers

7.3 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 which can integrate the crowd answers and opinions towards 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. The cloud server can then infer the true answer for each question through aggregating the answers from different users. In order 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. We use *Error Rate* as the evaluation metric and for the sake of evaluation, we have got 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 Fig. 14(a), from which we can see that PPTD produces the same error rates as CRH at all time. 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 error of user weights, we manually decrypt each user's weight derived by PPTD. Here we still use the relative error defined in section 7.2 to measure the errors introduced by L. The results are reported in Fig. 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.



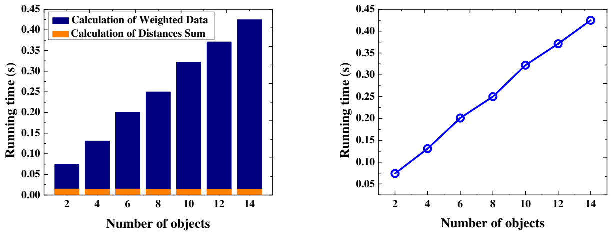
(a) Error rate

(b) Relative error of weights

Figure 14: Accuracy of PPTD for categorical data

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 user's smartphone, and then give the total running time. In this experiment, the number of the objects observed by each user varies from 1 to 14. The results are shown in Fig. 15, from which we can see the second procedure (i.e., calculating weighted data) costs more time than the first procedure (Fig. 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, Fig. 15(b) shows that the largest total running time of the two procedures on user smartphone is no more than 0.45s in each iteration, which verifies the practicality of our proposed framework.

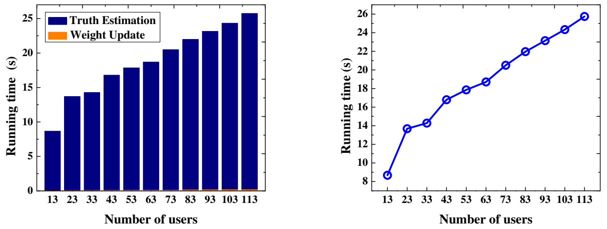


(a) Running time of each procedure

(b) Total running time

Figure 15: Running time w.r.t Number of objects for categorical data on smartphone

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 total running time in each iteration. From Fig. 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 Fig. 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 ten times larger than that in the experiment for continuous data.



(a) Running time of each procedure

(b) Total running time

Figure 16: Running time w.r.t Number of users for categorical data on the cloud server

7.4 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 1000 users and 1000 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.

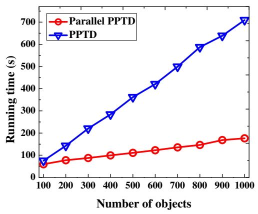


Figure 17: Running time w.r.t Number of objects for parallel PPTD

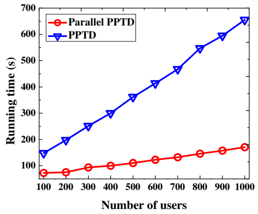


Figure 18: Running time w.r.t Number of users for parallel PPTD

When evaluating running time under different user numbers and object numbers, we adopt 10 Reducer nodes for parallel PPTD. Firstly, we fix the user number as 500 and change the object number from 100 to 1000. The running time of parallel PPTD and the basic PPTD in each iteration are shown in Fig. 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 1000, 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 1000. Figure 18 reports the results in this case. Similar patterns can be seen. When user number reaches 1000, 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 13 shows the running time under different number 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.

8. RELATED WORK

As an effective technique to extract reliable information from crowd sensing systems, truth discovery has drawn more and more attention [29–32, 34, 35, 44, 47–50] in recent years. Representative truth discovery schemes include AccuSim [31], CRH [30], TruthFinder [50], etc. 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, 22, 36]. The representative strategies to tackle various privacy concerns include 1) anonymization [7,38,46], which removes identification information from all the interactions between the participant and other entities, 2) data perturbation [26, 27], 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, 24], 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, privacy-preserving problem is also studied with respect to crowd sensing applications. For example, [23, 42, 43] 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, 37, 51]. 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 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 procedure without significant overhead. Additionally, paper [25] 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, paper [25] mainly focuses on categorical data. In contrast, our scheme can deal with not only categorical data but also other data types.

9. CONCLUSIONS

In this paper, we design a cloud-enabled privacy-preserving truth discovery (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 homomorphic cryptosystem, and iteratively conduct 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.

10. ACKNOWLEDGMENTS

We thank our shepherd Archan Misra and the anonymous reviewers for their valuable comments and suggestions. This work was sponsored in part by US National Science Foundation under grant IIS-1319973 and CNS-1262277.

11. REFERENCES

[1] S. Chen, M. Li, K. Ren, X. Fu, and C. Qiao. 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)*, 2015.

[2] S. Chen, M. Li, K. Ren, and C. Qiao. 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)*, 2015.

[3] Y. Cheng, X. Li, Z. Li, S. Jiang, Y. Li, J. Jia, and X. Jiang. Aircloud: a cloud-based air-quality monitoring system for everyone. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems (Sensys'14)*, 2014.

[4] Y. Chon, Y. Kim, and H. Cha. 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)*, 2013.

[5] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha. 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)*, 2013.

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

[7] C.-Y. Chow, M. F. Mokbel, and T. He. A privacy-preserving location monitoring system for wireless sensor networks. *IEEE Transactions on Mobile Computing*, (1):94–107, 2010.

[8] R. Cramer, I. Damgård, and J. B. Nielsen. *Multiparty computation from threshold homomorphic encryption*. Springer, 2001.

[9] I. Damgård and M. Jurik. A generalisation, a simpli. cation and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*. Springer, 2001.

[10] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[11] C. Fontaine and F. Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007:15, 2007.

[12] R. K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T. F. Abdelzaher. Greengps: a participatory sensing fuel-efficient maps application. In *Proceedings of the 8th ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'10)*, 2010.

[13] R. K. Ganti, N. Pham, Y.-E. Tsai, and T. F. Abdelzaher. Poolview: stream privacy for grassroots participatory sensing. In *Proceedings of the 6th ACM conference on Embedded network sensor systems (Sensys'08)*, 2008.

[14] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[15] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li. Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In *Proceedings of the 20th ACM Annual International Conference on Mobile Computing and Networking (MobiCom'14)*, 2014.

[16] C. Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[17] O. Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 1998.

[18] S. Hu, H. Liu, L. Su, H. Wang, T. F. Abdelzaher, P. Hui, W. Zheng, Z. Xie, J. Stankovic, et al. Towards automatic phone-to-phone communication for vehicular networking applications. In *Proceedings of the 33th Annual IEEE International Conference on Computer Communications (INFOCOM'14)*, 2014.

[19] S. Hu, L. Su, S. Li, S. Wang, C. Pan, S. Gu, T. Amin, H. Liu, S. Nath, R. R. Choudhury, et al. Experiences with enav: A low-power vehicular navigation system. In *Proceedings of the 2015 ACM Conference on Ubiquitous Computing (UbiComp'15)*, 2015.

[20] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher. 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)*, 2013.

[21] S. Hu, L. Su, H. Liu, H. Wang, and T. F. Abdelzaher. Smartroad: Smartphone-based crowd sensing for traffic regulator detection and identification. *ACM Transactions on Sensor Networks (TOSN)*, 11(4):55, 2015.

[22] K. L. Huang, S. S. Kanhere, and W. Hu. Towards privacy-sensitive participatory sensing. In *Proceedings of the 7th IEEE International Conference on Pervasive Computing and Communications (PerCom 2009)*, 2009.

[23] K. L. Huang, S. S. Kanhere, and W. Hu. A privacy-preserving reputation system for participatory sensing. In *LCN*, 2012.

[24] T. Jung, X. Mao, X.-Y. Li, S.-J. Tang, W. Gong, and L. Zhang. Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation. In *Proceedings of the 32st Annual IEEE International Conference on Computer Communications (INFOCOM'13)*, 2013.

[25] H. Kajino, H. Arai, and H. Kashima. Preserving worker privacy in crowdsourcing. *Data Mining and Knowledge Discovery*, 28(5-6):1314–1335, 2014.

[26] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the third IEEE International Conference on Data Mining (ICDM'03)*, 2003.

[27] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *Knowledge and Information Systems*, 7(4):387–414, 2005.

[28] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha. 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), 2013.

[29] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han. A confidence-aware approach for truth discovery on long-tail data. *Proceedings of the VLDB Endowment*, 8(4):425–436, 2014.

[30] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han. 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)*, 2014.

[31] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: is the problem solved? *Proceedings of the VLDB Endowment*, 6(2):97–108, 2012.

[32] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, and J. Han. On the discovery of evolving truth. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'15)*, 2015.

[33] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Proceedings of the 20th Annual International Cryptology Conference (CRYPTO'00)*, 2000.

[34] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, and J. Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(SIGKDD'15)*, 2015.

[35] C. Meng, W. Jiang, Y. Li, J. Gao, L. Su, H. Ding, and Y. Cheng. Truth discovery on crowd sensing of correlated entities. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (Sensys'15)*, 2015.

[36] S. Oh, T. Vu, M. Gruteser, and S. Banerjee. Phantom: Physical layer cooperation for location privacy protection. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM'12)*, 2012.

[37] N. Pham, R. K. Ganti, Y. S. Uddin, S. Nath, and T. Abdelzaher. Privacy-preserving reconstruction of multidimensional data maps in vehicular participatory sensing. In *Wireless Sensor Networks*. Springer, 2010.

[38] L. Pournajaf, L. Xiong, D. A. Garcia-Ulloa, and V. Sunderam. A survey on privacy in mobile crowd sensing task management. Technical report, Technical Report TR-2014-002, Department of Mathe-matics and Computer Science, Emory University, 2014.

[39] M.-R. Ra, B. Liu, T. F. La Porta, and R. Govindan. Medusa: A programming framework for crowd-sensing applications. In *Proceedings of the 10th ACM International Conference on Mobile systems, applications, and services (MobiSys'12)*, 2012.

[40] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow. 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)*, 2011.

[41] F. Saremi, O. Fatemieh, H. Ahmadi, H. Wang, T. Abdelzaher, R. Ganti, H. Liu, S. Hu, S. Li, and L. Su. Experiences with greengps–fuel-efficient navigation using participatory sensing. *IEEE Transactions on Mobile Computing (TMC)*, PP(99):1, 2015.

[42] K. Shilton. Four billion little brothers?: Privacy, mobile phones, and ubiquitous data collection. *Communications of the ACM*, 52(11):48–53, 2009.

[43] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, and N. Triandopoulos. Anonysense: A system for anonymous opportunistic sensing. *Pervasive and Mobile Computing*, 7(1):16–30, 2011.

[44] L. Su, Q. Li, S. Hu, S. Wang, J. Gao, H. Liu, T. F. Abdelzaher, J. Han, X. Liu, Y. Gao, et al. Generalized decision aggregation in distributed sensing systems. In *Proceedings of the 35th IEEE International Conference on Real-Time Systems Symposium (RTSS'14)*, 2014.

[45] V. Subbaraju, A. Kumar, V. Nandakumar, S. Batra, S. Kanhere, P. De, V. Naik, D. Chakraborty, and A. MISRA. 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)*, 2013.

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

[47] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher. 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)*, 2012.

[48] S. Wang, L. Su, S. Li, S. Hu, T. Amin, H. Wang, S. Yao, L. Kaplan, and T. Abdelzaher. Scalable social sensing of interdependent phenomena. In *Proceedings of the 14th ACM International Conference on Information Processing in Sensor Networks (IPSN'15)*, 2015.

[49] S. Wang, D. Wang, L. Su, L. Kaplan, and T. F. Abdelzaher. 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)*, 2014.

[50] X. Yin, J. Han, and P. S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796–808, 2008.

[51] F. Zhang, L. He, W. He, and X. Liu. Data perturbation with state-dependent noise for participatory sensing. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM'12)*, 2012.