A Sybil-Resistant Truth Discovery Framework for Mobile Crowdsensing

Jian Lin* Dejun Yang*, Kun Wu†, Jian Tang† and Guoliang Xue‡
*Colorado School of Mines,†Syracuse University,‡Arizona State University
{jilin, djyang}@mines.edu, {kwu102, jtang02}@syr.edu, xue@asu.edu

***Abstract*—The rapid proliferation of sensor-embedded devices has enabled the mobile crowdsensing (MCS), a new paradigm which effectively collects sensing data from pervasive users. In order to identify the true information from the noisy data submitted by unreliable users, truth discovery algorithms have been proposed for the MCS systems to aggregate data. However, the power of truth discovery algorithms will be undermined by the Sybil attack, in which an attacker can benefit from using multiple accounts. In addition, an MCS system will be jeopardized unless it is resistant to the Sybil attack. In this paper, we proposed a Sybil-resistant truth discovery framework for MCS, which ensures high accuracy under the Sybil attack. To diminish the impact of the Sybil attack, we design three account grouping methods for the framework, which are used in pair with a truth discovery algorithm. We evaluate the proposed framework through a real-world experiment. The results show that existing truth discovery algorithms are vulnerable to the Sybil attack, and the proposed framework can effectively diminish the impact of the Sybil attack.**

I. Introduction

With the rapid proliferation of mobile devices equipped with rich on-board sensors, mobile crowdsensing (MCS) emerges as a new sensing paradigm, which outsources sensing tasks to a crowd of ubiquitous participants. The effectiveness of crowdsensing to collect data enabled numerous MCS applications, which facilitate our lives in various aspects including transportation [18], environmental monitoring [23], social networks [16], and etc.

A typical MCS system consists of a cloud-based platform and a crowd of participates. The success of an MCS application relies on two factors: 1) a sufficient number of users and 2) the quality of sensing data contributed by individual users. In recent years, many incentive mechanisms [6, 32, 33, 35] have been proposed to stimulate users to participate in MCS. In practice, the quality of users' sensing data varies due to many factors, e.g., insufficient skill, poor sensor quality, environmental noise, and etc. Therefore, the platform needs to properly aggregate noisy sensing data collected from a variety of sources to identify the true information (i.e., the *truths*). It is ideal for the platform to use a weighted aggregation method, which assigns higher weights to reliable users. As a result, the aggregated result towards the data provided by reliable users. However, the reliability of users is usually unknown to the platform. Therefore, truth discovery [34] has been proposed to address this issue. Without any prior knowledge about users' reliability, a truth discovery algorithm iteratively assigns different weights to users according to the quality of their data and computes the estimated truth as the weighted average of all data. This process repeats until it converges.

Although the benefits of truth discovery are well understood, the aggregation accuracy highly depends on the quality of input data also raises security concerns. Existing truth discovery algorithms [9–11, 34] assume that most users are reliable. However, this assumption does not hold in practice, especially when an MCS system is under the Sybil attack [3]. In an MCS system, a user could conduct the Sybil attack by submitting data using multiple accounts for various motives. For example, a *rapacious* user may want to receive more rewards without contributing extra efforts (e.g., by submitting duplicated data multiple times using different accounts). Whereas, a *malicious* user may aim to manipulate the results of the system (e.g., by submitting multiple fake data using different accounts as in [28]). The Sybil attack is easy to conduct (e.g., creating multiple accounts) but difficult to detect. Therefore, the power of truth discovery algorithms will be undermined, and the performance of an MCS system will be jeopardized unless it is resistant to the Sybil attack.

The impact of the Sybil attack in MCS has been firstly investigated by Lin *et al.* [12] from an incentive perspective. They proposed Sybil-proof incentive mechanisms for both online [13] and off-line [12] scenarios. In their models, the objective of the Sybil attackers is to maximize its utility which is the difference between the payment and the cost. These incentive mechanisms can prevent rapacious users since they can fundamentally eliminate these users' motivation to conduct the Sybil attack. However, they cannot address malicious attackers since the objective of malicious attackers is to manipulate the results of the system. In addition, existing truth discovery algorithms are vulnerable to the Sybil attack (to be demonstrated in III-C). Although many Sybil attack defense mechanisms have been proposed in recent years [26]. These mechanisms do not consider the behaviors of mobile users in MCS and cannot be applied to MCS directly. Thus, *the problem of designing Sybil-resistant truth discovery algorithms for MCS remains open.*

In this paper, we focus on designing a Sybil-resistant truth discovery framework for MCS, which diminishes the impact of the Sybil attack on the aggregated results. We consider two types of Sybil attacks based on whether a Sybil attacker uses

This research was supported in part by NSF grants 1525920, 1704092, 1717197, and 1717315. The information reported here does not reflect the position or the policy of the federal government.

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00091

multiple devices. These two types of attacks are sufficient to represent the general Sybil attack in MCS. We propose three account grouping methods to cluster suspicious users. The first method is based on users' device fingerprints. This method can defend against the first type of attacks. To defend against the second type of attacks, we propose another two methods. These two methods handle different scenarios. The second method is based on users' accomplished task sets. This method can be used in the scenario where accounts have diverse accomplished task sets. The third method is based on users' trajectories. This method can handle the scenario where most users perform similar tasks. These account grouping methods are used in pair with a truth discovery algorithm and ensure high aggregation accuracy.

The main contributions of this paper are as follows:

- To the best of our knowledge, we are the first to study the problem of the truth discovery under the Sybil attack for MCS. We characterize two types of Sybil attacks and demonstrate that existing truth discovery algorithms are vulnerable to the Sybil attack.
- To the diminish the impact of the Sybil attack, we propose a Sybil-resistant truth discovery framework to ensure high aggregation accuracy in MCS systems. Note that compared with existing truth discovery algorithms, we do not assume that most users are reliable. As an important component of the framework, we design three effective account grouping methods. Each method groups the accounts that are likely from the same Sybil attacker.
- We evaluate the proposed framework through a real-world experiment. The results show that existing truth discovery algorithms are vulnerable to the Sybil attack, and the proposed framework can effectively diminish the impact of the Sybil attack.

The remainder of this paper is organized as follows. In Section II, we review the related work. In Section III, we introduce the preliminaries. In Section IV, we present the details of the proposed Sybil-resistant truth discovery framework. Performance evaluations are presented in Section V. We conclude this paper in Section VI.

II. Related Work

In MCS systems, sensing data are collected from individual users with different qualities. Truth discovery, which aims to identify the true information (i.e., the truth) out of all collected data, has received considerable attention in both industry and academia. Truth discovery refers to a family of algorithms [9–11, 34] that aim to discover the truth from a crowd of users' noisy data. Recently, Jin *et al.* [7] took into consideration the users' strategic behaviors and proposed a payment mechanism to incentivize high-effort sensing from users. Tang *et al.* [25] considered the privacy issue in truth discovery and proposed a non-interactive privacy-preserving truth discovery system for MCS. Zhu *et al.* [36] proposed to use interactive filtering truth discovery to provide reliable crowdsensing services in connected vehicular cloud computing. However, none of these works considered the Sybil attack, and thus may yield unsatisfactory results due to the vulnerability to the Sybil attack.

Lin *et al.* [12] firstly formalized the Sybil attack in MCS and demonstrated that previous incentive mechanisms for MCS are all vulnerable to the Sybil attack. Two Sybil-proof incentive mechanisms are proposed for both on-line [13] and off-line [12] scenarios, respectively. Jiang *et al.* [5] considered the time-sensitive and proposed time-sensitive and Sybil-proof incentive mechanisms for MCS. Although these Sybil-proof incentive mechanisms can fundamentally eliminate rapacious users' motivation to conduct the Sybil attack according to their model, malicious users who aim to manipulate the aggregated data through the Sybil attack cannot be addressed by these mechanisms. This is because they only considered rapacious users in their model.

In recent years, the impact of the Sybil attack has been widely analyzed in a variety of domains, e.g., social networks [27], crowdsourced mobile apps [28], virtual machine instance allocation [29], and Wireless sensor networks [22]. However, the problem of designing Sybil-resistant truth discovery algorithms for MCS is still open. All existing truth discovery algorithms are vulnerable to the Sybil attack. We will show their vulnerabilities to the Sybil attack in Section III-C.

III. Preliminaries

In this section, we first introduce the system model, truth discovery, and the adversary models. At last, we introduce the device fingerprinting, which is used to design our Sybil-resistant framework.

A. System Model

We consider an MCS system consisting of a cloud-based platform and a crowd of n mobile users $\mathcal{U} = \{1, 2, \cdots, n\}$. The platform first publicizes a set $\mathcal{T} = \{\tau_1, \tau_2, \cdots, \tau_m\}$ sensing tasks. Each sensing task can be a task to sense a particular object, event or local phenomenon in a specific sensing region. Then, each user performs sensing tasks $\mathcal{T}_i \subseteq \mathcal{T}$. Let $\mathcal{D}_i = \{(d_j^i, t_j^i) | \tau_j \in \mathcal{T}_i\}$ denote user i's accomplished task set, where d_j^i is the sensing data for task τ_j in the form of numerical values, e.g., cellular signal strength [19], noise level measurements [23], and Wi-Fi signal strength [30], and t_j^i is the corresponding timestamp. Each user i submits $\mathcal{D}_i$ to the platform. Meanwhile, the platform collects the sensor data from i's device for device fingerprinting (to be elaborated in III-D). Note that, this sensor data is independent of the sensing data d_j^i. Once the sensing data $\mathbb{D} = \{\mathcal{D}_i | i \in \mathcal{U}\}$ from all users has been collected, the platform calculates an aggregated result d_j for each task τ_j as an estimate for the truth, which is unknown to either the platform or the users. In practice, the quality of sensing data from different users varies. In addition, the users' data quality is unknown to the platform. Therefore, it is common for a platform to utilize a truth discovery algorithm to aggregate data, which calculates users' weights and estimates the truths in a joint manner.

B. Truth Discovery

A general truth discovery algorithm involves iterative estimations of weights and truth in a joint manner as summarized in Algorithm 1, which can be divided into two phases: weight estimation and truth estimation.

At first, the algorithm randomly guesses each task's truth, and then iteratively updates each user's weight and the estimated truth until it convergences. Note that the convergence criterion is based on applications. For example, the convergence criterion is the maximum number of iterations in [10].

Weight estimation: Let $\mathcal{U}_j = \{i | i \in \mathcal{U}, \tau_j \in \mathcal{T}_i\}$ denote the set of users who submit sensing data for task τ_j. In this step, given the estimated truth d_j for any task τ_j, and the weight w_i of each user $i \in \mathcal{U}_j$ is calculated as

$$w_i = \mathcal{W}(\sum_{\tau_j \in \mathcal{T}_i} \mathcal{D}(d_j^i, d_j)), \tag{1}$$

where $\mathcal{W}(\cdot)$ is a monotonically decreasing function, and $\mathcal{D}(\cdot)$ is the distance function measuring the difference between the user's data and the estimated truth.

Truth estimation: In this step, given users' weights for task τ_j, the estimated truth of task τ_j is calculated as

$$d_j = \frac{\sum_{i \in \mathcal{U}_j} w_i d_j^i}{\sum_{i \in \mathcal{U}_j} w_i}. \tag{2}$$

Although existing truth discovery algorithms [9–11, 34] differ in their ways to update users' weights and the estimated truth, they all follow the same principles: 1) the users whose data are closer to the estimated truth are assigned higher weights; 2) the aggregated result relies more on the users with higher weights.

Algorithm 1: Truth Discovery Algorithm

```
Input: Users' data $\mathbb{D}$;
Output: Estimated truths $\{d_j | \tau_j \in \mathcal{T}\}$;
Randomly initialize the estimated truth for each task;
repeat
    // Weight estimation
    foreach $i \in \mathcal{U}_j$ do
        Update the weight $w_i$ based on (1);
    end
    // Truth estimation
    foreach $\tau_j \in \mathcal{T}$ do
        Update the estimated truth $d_j$ based on (2);
    end
until Convergence criterion is satisfied;
return Estimated truths $\{d_j | \tau_j \in \mathcal{T}\}$.
```

C. Adversary Models

An MCS system can recruit users in various channels such as Aamazon Mechanical Turk. Each user performs a task by submitting timestamped sensing data via an app to the platform. In this paper, we assume that users could conduct the Sybil attack but cannot tamper with the device. Fabricated mobile sensor data through tampering attacks can be identified by evaluating the authenticity of the data [21]. A Sybil attacker is a user who performs a task once but submits data multiple times under different accounts (possibly after simple modification). We also assume that an attacker can submit fake sensing data, but the timestamps cannot be fabricated. Fabricated timestamps can be detected using the method in [31]. We consider the following two scenarios depending on whether a Sybil attacker uses multiple devices. Note that these two scenarios are sufficient to represent the general Sybil attack in MCS.

Attack-I: single device and multiple accounts. In this attack, an attacker i has only one device. Therefore, it can only conduct the Sybil attack by creating multiple accounts. For example, attacker i performs the sensing task τ_j and submits the sensing data $d_j^{i'}$ at $t_j^{i'}$ using account i'. Then it switches to account i'' and submits $d_j^{i''}$ (could be fabricated) at $t_j^{i''}$, which could be a copy of $d_j^{i'}$ without sensing effort. Note that, $t_j^{i'}$ and $t_j^{i''}$ are different due to switching accounts, and the device fingerprints associated with i' and i'' are supposed to be the same since they are from the same device.

Attack-II: multiple devices and multiple accounts. In this attack, an attacker i has multiple devices. Therefore, it can conduct the Sybil attack by using multiple accounts on different devices. For example, attacker i performs the sensing task τ_j and submits the sensing data $d_j^{i'}$ (could be fabricated) at $t_j^{i'}$ using account i' on one device. Then i uses another device with account i'' and submits data $d_j^{i''}$ at $t_j^{i''}$, which could be a copy of $d_j^{i'}$ without sensing effort. Note that, $t_{i'}^{\tau_j}$ and $t_{i''}^{\tau_j}$ are different due to the switching device, and the device fingerprints associated with i' and i'' could be different since they are from different devices.

Next, we use an example shown in Table I to demonstrate that existing truth discovery algorithms [9–11, 34] are vulnerable to the Sybil attack. Consider an MCS system with 4 tasks and 4 users. Each task is measuring the Wi-Fi signal strength (dBm) at a specific location. Each account is allowed to submit at most one data for one task. For each task, the system aggregates data submitted by users to identify the Wi-Fi signal strength at the associated location. Assume that User 4 is a Sybil attacker who has one device and three accounts (Attack-I), i.e., $4'$, $4''$, and $4'''$. The objective of the attacker is to mislead the system to decide that the Wi-Fi signals in locations of Task 1, 3, and 4 are strong. Therefore, User 4 will use 3 accounts to submit fabricated data (-50 dBm), which represents a strong Wi-Fi signal strength. We use CRH [10], a widely adopted truth discovery (TD) algorithm, to aggregate the data in the cases without and with Sybil attacker 4, respectively. Note that CRH is sufficient to represent existing truth discovery algorithms since they have the same procedure as Algorithm 1. According to the result, we see that the Sybil attack has a significant impact on the aggregated results for Task 1, 3, and 4. This example can be extended to any MCS systems where the sensing data for each task is in the form of numerical values. Therefore, it is urgent to design effective

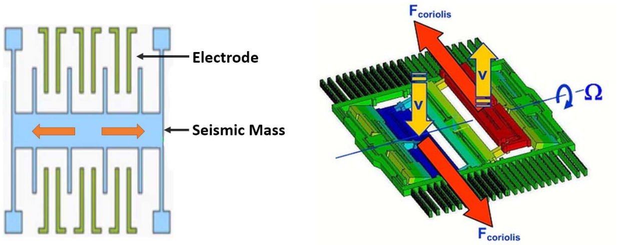
Sybil-resistant truth discovery algorithms for the MCS to diminish the impact of the Sybil attack.

TABLE I
EXAMPLE SHOWING THE SYBIL ATTACK IN MCS

	T1	T2	T3	T4
1	-84.48	-82.11	-75.16	-72.71
2	x	-72.27	-77.21	x
3	-72.41	-91.49	x	-73.55
$4'$	-50	x	-50	-50
$4''$	-50	x	-50	-50
$4'''$	-50	x	-50	-50
TD without the Sybil attack	-84.23	-82.01	-75.22	-72.72
TD with the Sybil attack	-56.06	-86.17	-53.29	-55.35

D. Device Fingerprinting

Device fingerprinting refers to finding characteristics that can help identify an individual device. It has been exploited to track a user across multiple visits to websites [2]. Modern mobile devices (e.g., smartphones) are equipped with rich sensors, e.g., accelerometers and gyroscopes, that are available to apps. In recent years, these sensors are fully invested and have been used to uniquely identify a smartphone by measuring anomalies in their produced signals, which are the result of manufacturing imperfections. The accelerometer and gyroscope sensors in smartphones are based on Micro Electro Mechanical Systems (MEMS) as shown in Fig. 1. When an accelerometer is subjected to a linear acceleration along its sensitive axis, the seismic mass will shift closer to one of the fixed electrodes causing a change in the generated capacitance. During the manufacturing process, there might be a slight gap difference between these electrodes for different chips, which may cause a difference in the generated capacitance for the same acceleration. A gyroscope measures the rate of rotation based on the Coriolis effect. The angular velocity is computed from the Coriolis force, which is sensed by a capacitive sensing structure. A change in the vibration of the proof mass causes a change in capacitance. Similar to the accelerometers, the imperfection in the electro-mechanical structure may cause a difference in the generated capacitance for the same Coriolis force across chips. In this paper, we exploit both accelerometer and gyroscope for device fingerprinting. The fingerprints generated from these two sensors can uniquely identify a mobile device.



(a) Accelerometer (b) Gyroscope

Fig. 1. MEMS-based accelerometer and gyroscope

IV. Sybil-resistant Truth Discovery Framework

In this section, we design a truth discovery framework for MCS, which is resistant to Sybil attack. Note that we use accounts instead of users throughout the rest of this paper since a Sybil attacker can have multiple accounts.

A. Design Rationale

The key idea of the framework is to identify the data submitted by suspicious accounts and diminish the impact of these data on the MCS system. The following three methods can be used to identify these data: 1) In an MCS system, users usually collect and submit sensing data using their mobile devices (e.g., smartphones). Therefore, if multiple data submitted by different accounts can be identified to be submitted by the same device, these data should be assigned less weight in truth discovery since they might be from a Sybil attacker. 2) In order to manipulate the aggregated results for multiple tasks, a Sybil attacker ought to submit data for each task using different accounts (as the example in III-C). Therefore, the data submitted by accounts with highly similar task set are likely from a Sybil attacker. 3) Similarly, the trajectories of a Sybil attacker's accounts should have a similar pattern since they belong to the same user. Therefore, the data submitted by accounts with highly similar trajectories are likely from a Sybil attacker. In order to diminish the impact of the data submitted by Sybil attackers, we group the accounts potentially from Sybil attackers and assign low weights to their data in the truth discovery algorithm. Note that we do not directly eliminate the data submitted by suspicious accounts since there might be false-positives in the identifying process.

B. Design of Framework

We now describe the details of the framework, which is illustrated in Algorithm 2. It takes as inputs the users' data for each task and their associated device fingerprints. The framework first groups all accounts (to be elaborated in IV-C). Let $\mathcal{G} = \{g_1, g_2, \cdots, g_l\}$ denote the account grouping result, where $g_i \cap g_j = \emptyset$ and $\cup_{g_i \in \mathcal{G}} g_i = \mathcal{U}$. Each group $g_k \in \mathcal{G}$ represents a set of accounts likely used by the same user. Let $\tilde{\mathcal{T}}_k = \cup_{i \in g_k} \mathcal{T}_i$ denote the set of tasks performed by the accounts in group g_k. Then the framework groups data as follows. For each task τ_j, we first aggregate the data within each group $g_k \in \mathcal{G}_j$, where $\mathcal{G}_j = \{g_k | g_k \in \mathcal{G}, \tau_j \in \tilde{\mathcal{T}}_k\}$, according to

$$\tilde{d}_j^k = \frac{\sum_{i \in g_k} (d_j^i - \bar{d}_j^k) d_j^i}{\sum_{i \in g_k} (d_j^i - \bar{d}_j^k)}, \tag{3}$$

where $\bar{d}_j^k$ is the arithmetic mean of data submitted by accounts in g_k. The weight of each group $g_k \in \mathcal{G}_j$ is calculated as

$$\tilde{w}_k = 1 - \frac{|g_k|}{|\mathcal{U}_j|}, \tag{4}$$

where $|g_k|$ denotes the number of accounts in group g_k, and $|\mathcal{U}_j|$ is the number of accounts who submit data for task τ_j. Note that using one data for each group could diminish the impact of the Sybil attack.

Algorithm 2: Sybil-resistant Truth Discovery Framework

```
Input: Users' data $\mathbb{D}$ and device fingerprints $\mathbb{F}$;
Output: Estimated truths $\{d_j|\tau_j \in \mathcal{T}\}$;
// Account grouping
$\mathcal{G} \leftarrow$ AG($\mathbb{D}$, $\mathbb{F}$);
// Data grouping
foreach $\tau_j \in \mathcal{T}$ do
    Group sensing data for $\tau_j$ based on $\mathcal{G}$;
    Aggregate the data in each group $g_k \in \mathcal{G}_j$ by (3);
    Calculate the weight $\tilde{w}_k$ of each group by (4);
end
Initialize the estimated truth for each task by (5);
repeat
    // Group weight estimation
    foreach $g_k \in \mathcal{G}$ do
        Update the weight $\tilde{w}_k$ using
        $\tilde{w}_k = \mathcal{W}(\sum_{\tau_j \in \tilde{\mathcal{T}}_k} \mathcal{D}(\tilde{d}_j^k, d_j))$, where
        $\tilde{\mathcal{T}}_k = \cup_{i \in g_k} \mathcal{T}_i$;
    end
    // Truth estimation
    foreach $\tau_j \in \mathcal{T}$ do
        Update the estimated truth using
        $d_j = \frac{\sum_{g_k \in \mathcal{G}_j} \tilde{w}_k \tilde{d}_j^k}{\sum_{g_k \in \mathcal{G}_j} \tilde{w}_k}$;
    end
until Convergence criterion is satisfied;
return Estimated truths $\{d_j|\tau_j \in \mathcal{T}\}$.
```

At last, the framework estimates the truth for each task similar to Algorithm 1. Different from Algorithm 1, we treat the accounts in one group as a whole, and thus we use $\tilde{d}_j^k$ for group g_k. In addition, instead of randomly initializing the estimated truth for each task, we initialize the estimated truth of each task τ_j as follows

$$d_j = \frac{\sum_{g_k \in G_j} \tilde{w}_k \tilde{d}_j^k}{\sum_{g_k \in G_j} \tilde{w}_k}. \tag{5}$$

C. Account Grouping

As an important component in the framework, account grouping is used to group accounts that are likely from the same Sybil attacker. Specifically, we design three account grouping methods.

Account Grouping by Device Fingerprint (AG-FP). Inspired by a recent work on device fingerprinting [2], we design an account grouping method that exploits both accelerometer and gyroscope to produce device fingerprints. The device fingerprint can be used to identify the sensing data from different accounts but the same device.

We treat the data from accelerometer and gyroscope as two streams of timestamped real values. Given a timestamp t, the platform collects values from accelerometer and gyroscope along three axes in the form of $\overrightarrow{a}(t) = (a_x, a_y, a_z)$ and $\overrightarrow{w}(t) = (w_x, w_y, w_z)$, respectively. The platform starts collecting these sensor data from the moment an account i signs in the system for T seconds and stores these data in $F_i = ((\overrightarrow{a}(1), \overrightarrow{a}(2), \cdots, \overrightarrow{a}(T)), (\overrightarrow{w}(1), \overrightarrow{w}(2), \cdots, \overrightarrow{w}(T)))$. Once all accounts' device fingerprints $\mathbb{F} = \{F_i | i \in \mathcal{U}\}$ have been collected, the platform converts the acceleration data at each timestamp into a scalar by taking its magnitude (i.e., $|\overrightarrow{a}(t)| = \sqrt{a_x^2 + a_y^2 + a_z^2}$) such that the accelerometer data is independent of the device orientation. For the gyroscope, we consider data from each axis as a separate stream. Therefore, each account i's device fingerprint will be considered as four sensor data streams in the form of $\{|\overrightarrow{a}(t)|, w_x(t), w_y(t), w_z(t)\}$. To characterize a sensor data stream, we use both temporal and spectral features as summarized in Table II, including 9 temporal and 11 spectral features. All of these features are widely used and have been well analyzed in the literature [2, 20].

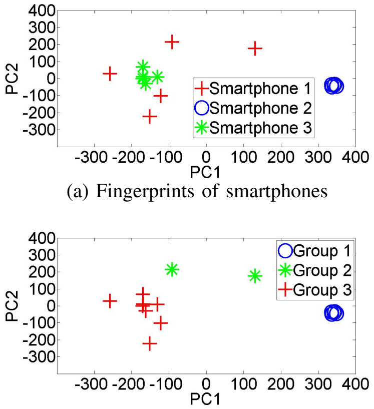
After extracting features from the sensor data, the platform groups device fingerprints using k-Means [15], which is a widely used clustering method in machine learning. The cluster number k is the number of devices in our case. Note that the platform does not know the exact number of devices in practice. Therefore, we use the elbow method [8] to estimate the value of k. The idea of the elbow method is to run k-Means on all the device fingerprints for a range of values of k (e.g., k from 1 to n) and calculate the sum of squared errors (SSE) for each value of k. Note that the SSE tends to decrease toward 0 with the increase of k. At last, we choose the value of k at which SSE starts to diminish. It is worth mentioning that the time complexity of k-Means is $O(nkdi)$, where n is the number of d-dimensional vectors, k is the number of clusters and i is the number of iterations needed until convergence. In addition, we also see that the running time of the elbow method is linear in the number of users. In an MCS system, the number of selected users for each task is usually limited and smaller than the total number of users [14]. Therefore, AG-FP is efficient in practice. Note that the device fingerprint grouping result is the result of account grouping since each account is associated with one fingerprint. As an example, we use 3 smartphones of different models, each collecting 5 fingerprint data. Fig. 2(a) shows the distribution of the fingerprints of these 3 smartphones in the first two principal components' feature space (denoted by PC1 and PC2, respectively). Fig. 2(b) shows the grouping result by k-Means when $k = 3$. We see that Smartphone 2 has a stable fingerprint (i.e., its 5 fingerprints are close together), and thus it can be easily differentiated from the other two smartphones. However, there are three fingerprints from Smartphone 1 that have been wrongly grouped with Smartphone 3. These are the false-positives of this grouping method.

AG-FP can be used to defend against Attack-I since it can diminish the impact of multiple data from different accounts but the same Sybil attacker using the same device. However, a Sybil attacker can use multiple devices in Attack-II. To effectively defend against Attack-II, we propose the following two account grouping methods.

Account Grouping by Task Set (AG-TS). As mentioned

TABLE II
Temporal and spectral features

#	Domain	Feature	Description
1		Mean	The arithmetic mean of the signal strength at different timestamps
2		Standard Deviation	Standard deviation of the signal strength
3		Skewness	Measure of asymmetry about mean
4		Kurtosis	Measure of the flatness or spikiness of a distribution
5	Time	RMS	Square root of the arithmetic mean of the squares of the signal strength at various timestamps
6		Max	Maximum signal strength
7		Min	Minimum signal strength
8		ZCR	The rate at which the signal changes sign from positive to negative or back
9		Non-Negative count	Number of non-negative values
10		Spectral Centroid	The center of mass of a spectral power distribution
11		Spectral Spread	The dispersion of the spectrum around its centroid
12		Spectral Skewness	The coefficient of skewness of a spectrum
13		Spectral Kurtosis	Measure of the flatness or spikiness of a distribution relative to a normal distribution
14		Spectral Flatness	Measures how energy is spread across the spectrum
15	Frequency	Spectral Irregularity	The degree of variation of the successive peaks of a spectrum
16		Spectral Entropy	The peaks of a spectrum and their locations
17		Spectral Rolloff	The frequency below which 85% of the distribution magnitude is concentrated
18		Spectral Brightness	Amount of spectral energy corresponding to frequencies higher than a given cut-off threshold
19		Spectral RMS	Square root of the arithmetic mean of the squares of the signal strength at various frequencies
20		Spectral Roughness	Average of all the dissonance between all possible pairs of peaks in a spectrum



(a) Fingerprints of smartphones

(b) Grouping result

Fig. 2. Example of AG-FP

before, there should be a consistency in the accomplished task sets of the accounts from the same Sybil attacker. Inspired by [22], we design an account grouping method, which groups accounts by calculating the affinity between two accounts according to their accomplished task sets. The grouping method involves the following steps:

1) Let $T_{i,j}$ denote the number of tasks both accounts i and j have done. Let $L_{i,j}$ denote the number of tasks either i or j has done alone. Then the affinity between i and j, denoted as $A_{i,j}$, is calculated as

$$A_{i,j} = (T_{i,j} - 2L_{i,j})\frac{T_{i,j} + L_{i,j}}{m}, \tag{6}$$

where m is the number of tasks. Note that the larger the affinity value, the more similar the accomplished task sets of two accounts.

2) An undirected graph is constructed, where accounts are the nodes and the undirected edge between i and j is weighted with their affinity values $A_{i,j}$. Note that only edges that are greater than a threshold ρ are included.

3) Connected components are discovered using Depth First Search (DFS) algorithm. Each component represents a set of accounts who have done a similar set of tasks.

4) Each component is a group, and the account that is not in any component will be treated as a separate group.

To illustrate the process of this grouping method, we use the example in Table III. This example has the same setting as the example in Table I, whereas the values in the table are the timestamps for the corresponding tasks.

TABLE III
Example showing Sybil attack in MCS

	T1	T2	T3	T4
1	10:00:35 a.m.	10:02:42 a.m.	10:10:22 a.m.	10:13:41 a.m.
2	x	10:04:15 a.m.	10:06:01 a.m.	x
3	10:01:21 a.m.	10:04:05 a.m.	x	10:08:28 a.m.
4’	10:01:10 a.m.	x	10:15:24 a.m.	10:20:06 a.m.
4”	10:01:34 a.m.	x	10:16:08 a.m.	10:21:25 a.m.
4”’	10:02:35 a.m.	x	10:17:35 a.m.	10:22:02 a.m.

Fig. 3 shows the procedure of AG-TS. The adjacency matrix in Fig. 3(a) shows the number of tasks both i and j have done. The adjacency matrix in Fig. 3(b) shows the number of tasks either i or j has done alone. Fig. 3(c) shows the affinity value of i with respect to j. We set the threshold $\rho = 1$, and thus an undirected graph is drawn as shown in Fig. 3(d). We see that one component is constructed in this example, i.e., $\{1, 4', 4'', 4''\}$. Accounts 2 and 3 are not in the component, and thus each of them is treated as a group. Therefore, the grouping result of this example consists of three groups, i.e., $\{1, 4', 4'', 4''\}$, $\{2\}$, and $\{3\}$. Note that AG-TS groups account 1 with the accounts used by Sybil attacker in the same group, and thus this is a false-positive.

GP-TS can be used in the scenario where accounts have diverse accomplished task sets. To handle the scenario where most accounts have similar accomplished task sets, we propose the following grouping method.

	1	2	3	4'	4''	4'''
1	-	2	2	3	3	3
2	2	-	2	2	2	2
3	2	2	-	2	2	2
4'	3	2	2	-	3	3
4''	3	2	2	3	-	3
4'''	3	2	2	3	3	-

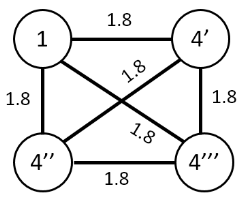
(a) $T_{i,j}$

	1	2	3	4'	4''	4'''
1	-	1	1	0	0	0
2	1	-	1	1	1	1
3	1	1	-	1	1	1
4'	0	1	1	-	0	0
4''	0	1	1	0	-	0
4'''	0	1	1	0	0	-

(b) $L_{i,j}$

	1	2	3	4'	4''	4'''
1	-	0	0	1.8	1.8	1.8
2	0	-	0	0	0	0
3	0	0	-	0	0	0
4'	1.8	0	0	-	1.8	1.8
4''	1.8	0	0	1.8	-	1.8
4'''	1.8	0	0	1.8	1.8	-

(c) $A_{i,j}$



(d) Undirected graph with $A_{i,j} > 1$

Fig. 3. Example of AG-TS

Account Grouping by Trajectory (AG-TR). The sensing data $D_i = \{(d_i^{\tau_j}, t_i^{\tau_j}) | \tau_j \in \mathcal{T}_i\}$ submitted by account i for task τ_j can be regarded as two time series data, i.e., task series X_i and timestamp series Y_i. These two time series data together can be regarded as the trajectory of an account. We design an account grouping method, which groups accounts by calculating the dissimilarity between two accounts according to their task series and timestamp series. This method relies on the principle that the accounts belonging to a Sybil attacker ought to perform similar set of tasks in a similar time pattern.

We use Dynamic Time Warping (DTW) [1] to measure the distance between two time series since it does not require two series to be the same length. Given two time series, $A = a_1, a_2, \cdots, a_m$ and $B = b_1, b_2, \cdots, b_n$, we construct an m-by-n matrix, where each element (i, j) of the matrix is the squared distance, i.e., $(a_i - b_j)^2$, representing the alignment between points a_i and b_j. The warping path $W = \omega_1, \omega_2, \cdots, \omega_K$ is a contiguous set of matrix elements that defines a mapping between A and B, where $\max(m, n) \leq K < m + n - 1$. The basic idea of DTW is to find out the warping path between two time series that minimizes the warping cost. We define the DTW distance as in [24]:

$$DTW(A, B) = \min\{\sqrt{\sum_{k=1}^{K} \omega_k / K}\}. \qquad (7)$$

This can be calculated using dynamic programming. Let $r(i, j)$ denote the cumulative distance, which is calculated as the distance $dist(a_i, b_j)$ found in the current cell and the minimum of the cumulative distances of the adjacent elements: $r(i, j) = dist(a_i, b_j) + \min\{r(i-1, j-1), r(i-1, j), r(i, j-1)\}$.

Based on the DTW distance between accounts' task series and time series, we propose an account grouping method, which involves the following steps:

1) The dissimilarity between accounts i and j, denoted as $D_{i,j}$, is calculated as
$$D_{i,j} = DTW(X_i, X_j) + DTW(Y_i, Y_j), \qquad (8)$$
where $DTW(X_i, X_j)$ is DTW distance between the task series of i and j, and $DTW(Y_i, Y_j)$ is the DTW distance between the timestamp series of i and j. Note that the less the dissimilarity value, the more similar the trajectories of two accounts.
2) An undirected graph is constructed, where accounts are the nodes and the undirected edge between i and j is weighted with their dissimilarity values $D_{i,j}$. Only edges that are less than a threshold ϕ are included.
3) Connected components are constructed using Depth First Search (DFS) algorithm. Each component represents a set of accounts with similar trajectories.
4) Each component is a group, and the account that is not in any component will be treated as a separate group.

	1	2	3	4'	4''	4'''
1	0	2	1	1	1	1
2	2	0	2	2	2	2
3	1	2	0	1	1	1
4'	1	2	1	0	0	0
4''	1	2	1	0	0	0
4'''	1	2	1	0	0	0

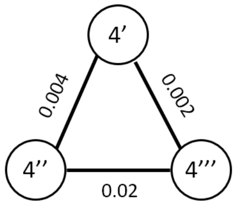
(a) $DTW(X_i, X_j)$

	1	2	3	4'	4''	4'''
1	0	0.01	0.01	0.01	0.01	0.01
2	0.01	0	0.02	0.02	0.02	0.02
3	0.01	0	0	0.02	0.06	0.02
4'	0.01	0.02	0.02	0	0.004	0.002
4''	0.01	0.02	0.015	0.004	0	0.02
4'''	0.01	0.02	0.02	0.002	0.02	0

(b) $DTW(Y_i, Y_j)$

	1	2	3	4'	4''	4'''
1	0	2.01	1.01	1.01	1.01	1.01
2	2.01	0	2.00	2.02	2.02	2.02
3	1.01	2.00	0	1.02	1.06	1.02
4'	1.01	2.02	1.02	0	0.004	0.002
4''	1.01	2.02	1.06	0.004	0	0.02
4'''	1.01	2.02	1.02	0.002	0.02	0

(c) $D_{i,j}$



(d) Undirected graph with $D_{i,j} < 1$

Fig. 4. Example of AG-TR

Next, we still use the example in Table III to illustrate the process of this grouping method. We use two adjacency matrices in Fig. 4(a) and Fig. 4(b) to show the DTW difference between i and j in terms of the task series and timestamp series, respectively. The adjacency matrix in Fig. 4(c) shows the dissimilarity value of i with respect to j. We set the threshold $\phi = 1$, and thus an undirected graph is drawn as shown in Fig. 4(d). We see that one component is constructed in this example, i.e., $\{4', 4'', 4''\}$. Accounts 1, 2, and 3 are not in the component, and thus each of them is treated as a group. Therefore, the grouping result includes four groups, i.e., $\{4', 4'', 4''\}$, $\{1\}$, $\{2\}$, and $\{3\}$. Comparing with the result of AG-TS, AG-TR has less false-positive since it correctly groups all the accounts used by the Sybil attacker in one group. The better performance of AG-TR relies on that it not only considers the similarities in accounts' accomplished tasks as in AG-TS but also considers the similarities in the associated timestamps of these tasks.

Remarks: In this paper, the aforementioned three account grouping methods are used independently in the framework. We leave the combination of them for our future work. The thresholds ρ in AG-TS and ϕ in AG-TR depend on the tasks in an MCS system. A higher value of ρ means that accounts are more likely have common accomplished tasks. A lower value of ϕ means that accounts are more likely have similar trajectories. Note that AG-TS and AG-TR may result in false-positive where two legitimate users with similar accomplished tasks and similar trajectories are considered as accounts from a Sybil attacker. This problem can be alleviated when the system uses existing incentive mechanisms [32, 33, 35] to incentivize and select users. This is because one of them is less likely selected by the incentive mechanism due to its marginal contribution if the other is selected.

V. Experiment

Since there is no public dataset with Sybil attackers' behaviors for MCS, we evaluate our framework by conducting experiments instead of large-scale simulations. In this section, we first describe our experimental setup. Then, we evaluate three account grouping methods. At last, we implement the framework with a truth discovery algorithm that is similar to CRH [10] and compare the aggregation accuracy of our proposed framework with CRH. As in [7], we use the mean absolute error (MAE) as the metric to measure aggregation accuracy, which is defined as $\frac{1}{m}\sum_{j=1}^{m}|d_j - d_j^*|$, where m is the number of tasks, and d_j and d_j^* are the estimated truth and ground truth for task τ_j, respectively. The lower the MAE value, the higher accuracy for the data aggregation. Note that we only compared our framework with CRH since it is sufficient to represent existing truth discovery algorithms as discussed before.

A. Experimental Setup

In our experiment, we consider an MCS system in which the tasks are measuring the Wi-Fi signal strength at 10 Point of Interest (POIs) as shown in Fig 5. We recruited 10 volunteers in our system, among them 8 acted as legitimate users and 2 acted as Sybil attackers. Each legitimate user has one account and uses one smartphone to perform tasks. Each of the two Sybil attackers has 5 accounts. One of the Sybil attacker conduct Attack-I with one smartphone, and the other conduct Attack-II with two smartphones of different models. Each account is only allowed to submit one sensing data at one POI. Therefore, a Sybil attacker can submit at most 5 data for one task using 5 accounts. We assume that the objective of each Sybil attacker is to mislead the system, and thus both Sybil attackers will fabricate the sensing data. Note that although we only have 2 Sybil attackers in our experiment, the experimental results can still represent the scenario when an MCS system is under a large scale of the Sybil attack since the percentage of the Sybil accounts is larger than that of the legitimate users. We collect the Wi-Fi signal strength at each POI multiple times and calculate the average as ground truth. To measure the activeness of each account, we define

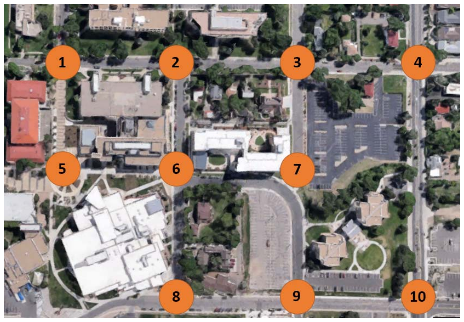


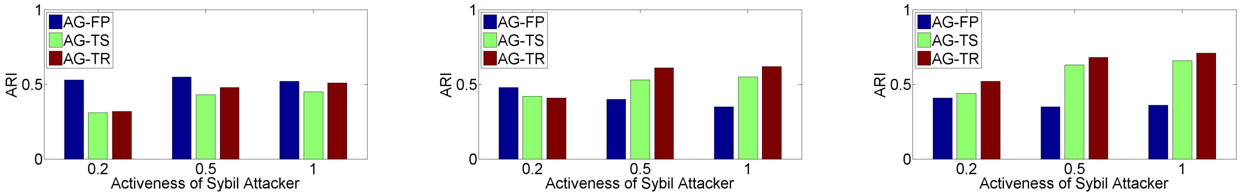
Fig. 5. POIs for Wi-Fi signal strength measurement

$$\alpha_i = \frac{|\mathcal{T}_i|}{m}, \tag{9}$$

where $|\mathcal{T}_i|$ is the number of tasks performed by i and m is the total number of tasks. In our experiment, each account has to perform at least two task, and thus $\alpha_i \in [0.2, 1]$. To some extent, the activeness is a good indicator to measure the contribution of legitimate accounts to the system. However, the more damages can be made by Sybil attackers with higher activeness. In our experiment, each user performs tasks according its own preference with according activeness. At last, we collect 54 walking traces in total. We use 11 smartphones in our experiment, and the distribution of these smartphones is listed in Table IV. One iPhone 6S is used to conduct Attack-I and one iPhone SE and one Nexus 6P are used to conduct Attack-II. As in [2], we collect device fingerprint through the browser by using a Javascript to access accelerometer and gyroscope. We use MIRtoolbox [17] to extract spectral features. Since AG-FP depends on the inherent imperfections of motion sensors to generate fingerprint, we need to keep the smartphone stationary while collecting sensor data. Therefore, we ask users to hold the smartphones in hand for 6 seconds when they sign in the system. Note that the Sybil attackers do this process again when they change accounts.

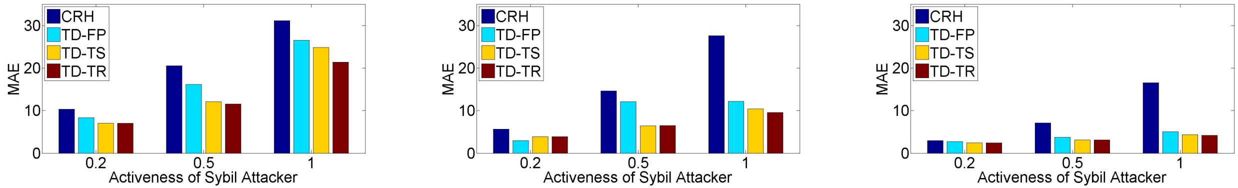
B. Evaluation of Account Grouping

In this part, we first show the performance of AG-FP, and then we compare the performance of the proposed three grouping methods in terms of the Adjusted Rand Index (ARI) [4], a widely used metric in machine learning to evaluate the



(a) Legitimate accounts' $\alpha = 0.2$ (b) Legitimate accounts' $\alpha = 0.5$ (c) Legitimate accounts' $\alpha = 1$

Fig. 6. ARI comparison



(a) Legitimate accounts' $\alpha = 0.2$ (b) Legitimate accounts' $\alpha = 0.5$ (c) Legitimate accounts' $\alpha = 1$

Fig. 7. MAE comparison

TABLE IV
MODELS OF SMARTPHONES USED IN THE EXPERIMENT

OS	Model	Quantity
iOS	iPhone SE**	1
	iPhone 6	1
	iPhone 6S*	2
	iPhone 7	1
	iPhone X	1
Android	Nexus 6P**	3
	LG G5	1
	Nexus 5	1
Total		11

$*$ Used to conduct Attack-I. $**$ Used to conduct Attack-II.

performance of clustering. The value of ARI lies in the range $[-1, 1]$, and the larger the value the better grouping result.

Fig. 8 shows the distribution of the center of all 11 smartphones in the space of the first two principal components. We see that the centers of the smartphones of the same model are very close, and thus it is hard to differentiate them. Actually, the smartphones of the same model are usually grouped together in our experiment. Therefore, the impact of Attack-I can be effectively diminished since multiple data submitted by a Sybil attacker for one task will be treated as a single data. However, there are false-positives in this method. The smartphone used by a legitimate user might be grouped with other smartphones either from legitimate users or from Sybil attackers. For the first case, the truth discovery result will not be effected since the aggregated data for the group is calculated based on data submitted by legitimate users. For the second case, the aggregated data for the group will be closed to the average of the data submitted by both legitimate users and Sybil attackers according to (3), and thus the impact of the Sybil attack will be diminished.

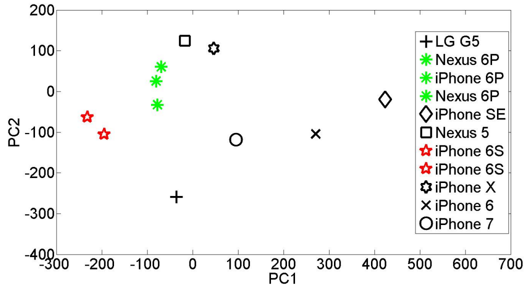


Fig. 8. Smartphone fingerprints in the first two principal components' space

Fig. 6 shows the ARI value of the three proposed grouping methods in different settings. In each setting, we fix the activeness of legitimate users and vary the activeness of Sybil attackers. Specifically, we consider three settings, i.e., $\alpha = 0.2$, 0.5 and 1. In Fig. 6, we see that the ARI of AG-FP decreases as the activeness increases. This is because, with more users in the system, there might be more smartphones of the same model, causing more false-positives. We also see that the ARI value of both AG-TS and AG-TR increase with the increase of activeness. This is because, more information (i.e., more accomplished tasks and longer trajectory) can be used to differentiate accounts when accounts have higher activeness. In addition, we see that the performance of AG-TR is better than AG-TS. This is because, AG-TR can still differentiate accounts according to their timestamp series when they have similar accomplished task sets.

C. Evaluation of Accuracy

We now use MAE as a metric to measure the accuracy of the proposed framework and compare it with CRH. We implement the framework with aforementioned three account grouping methods independently, denoted as TD-FP, TD-TS, and TD-TR, respectively. Fig. 7 shows the MAE of our framework and CRH in different settings. In each setting, we still fix the activeness of legitimate users and vary the activeness of Sybil attackers. We see that the MAE values of the four methods decrease with the increase of the activeness of legitimate users. This is because, with more data from legitimate users, it is harder for a Sybil attacker to manipulate the aggregated results. We also see that, fixing the activeness of legitimate users, the MAE increases with the increase of activeness of Sybil attackers. This demonstrate the impact of the Sybil attack on the aggregated results. The reason for this is that a larger activeness value of a Sybil attacker implies more false data, which may be a majority for a task causing the manipulation of the aggregated result. As shown in Fig. 7(c), the MAE of CRH is still large even the with a high activeness of legitimate users. On the contrary, the MAE of our proposed framework is always lower than CRH no matter which grouping method is used. This is because our framework can diminish the impact of the Sybil attack by grouping data from suspicious accounts. The performance of TD-TR is the better than TD-FP since it can address both Attack-I and Attack-II. Meanwhile, TD-TR is better than TD-TS since it has less false-positive in account grouping as discussed before.

VI. Conclusion

In this paper, we first analyzed the impact of the Sybil attack on existing truth discovery algorithms for MCS. We demonstrated that they are vulnerable to the Sybil attack. Then, we proposed a truth discovery framework, which is resistant to the Sybil attack and ensures high aggregation accuracy. Specifically, we designed three account grouping methods, which are used in pair with a truth discovery algorithm in the framework. These methods can effectively group accounts that are likely from the same Sybil attacker. We evaluated the proposed framework through a real-world experiment. The results show the vulnerability of existing truth discovery algorithm to the Sybil attack and the effectiveness of the proposed framework in diminishing the impact of the Sybil attack.

References

[1] D. J. Berndt and J. Clifford, “Using dynamic time warping to find patterns in time series.” in *KDD workshop*, vol. 10, no. 16, 1994, pp. 359–370.

[2] A. Das, N. Borisov, and M. Caesar, “Tracking mobile web users through motion sensors: Attacks and defenses.” in *NDSS*, 2016.

[3] J. R. Douceur, “The sybil attack,” in *IPTPS*, 2002, pp. 251–260.

[4] L. Hubert and P. Arabie, “Comparing partitions,” *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[5] L. Jiang, X. Niu, J. Xu, Y. Wang, Y. Wu, and L. Xu, “Time-sensitive and Sybil-proof incentive mechanisms for mobile crowdsensing via social network,” *IEEE Access*, vol. 6, pp. 48 156–48 168, 2018.

[6] H. Jin, L. Su, D. Chen, K. Nahrstedt, and J. Xu, “Quality of information aware incentive mechanisms for mobile crowd sensing systems,” in *MobiHoc*, 2015, pp. 167–176.

[7] H. Jin, L. Su, and K. Nahrstedt, “Theseus: Incentivizing truth discovery in mobile crowd sensing systems,” in *MobiHoc*, 2017, pp. 1–10.

[8] T. M. Kodinariya and P. R. Makwana, “Review on determining number of cluster in k-means clustering,” *International Journal*, vol. 1, no. 6, pp. 90–95, 2013.

[9] 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*, vol. 8, no. 4, pp. 425–436, 2014.

[10] 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 *SIGMOD*, 2014, pp. 1187–1198.

[11] Y. Li, Q. Li, J. Gao, L. Su, B. Zhao, W. Fan, and J. Han, “On the discovery of evolving truth,” in *KDD*, 2015, pp. 675–684.

[12] J. Lin, J. Li, J. Yang, G. Xue, and J. Tang, “Sybil-proof incentive mechanisms for crowdsensing,” in *INFOCOM*, 2017, pp. 2088–2096.

[13] J. Lin, J. Li, J. Yang, G. Xue, and J. Tang, “Sybil-proof online incentive mechanisms for crowdsensing,” in *INFOCOM*, 2018.

[14] Y. Liu, B. Guo, Y. Wang, W. Wu, Z. Yu, and D. Zhang, “Taskme: multi-task allocation in mobile crowd sensing,” in *UbiComp*, 2016, pp. 403–414.

[15] J. MacQueen *et al.*, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14, 1967, pp. 281–297.

[16] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, “Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application,” in *SenSys*, 2008, pp. 337–350.

[17] “MIRtoolbox,” https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox.

[18] P. Mohan, V. N. Padmanabhan, and R. Ramjee, “Nericell: rich monitoring of road and traffic conditions using mobile smartphones,” in *SenSys*, 2008, pp. 323–336.

[19] “Opensignal,” http://opensignal.com/.

[20] G. Peeters, “A large set of audio features for sound description (similarity and classification) in the CUIDADO project,” in *IRCAM Technical report*, 2004.

[21] H. Pieterse, M. Olivier, and R. van Heerden, “Smartphone data evaluation model: Identifying authentic smartphone data,” *Digital Investigation*, vol. 24, pp. 11–24, 2018.

[22] C. Piro, C. Shields, and B. N. Levine, “Detecting the sybil attack in mobile ad hoc networks,” in *Securecomm and Workshops, 2006*, 2006, pp. 1–11.

[23] R. K. Rana, C. T. Chou, S. S. Kanhere, N. Bulusu, and W. Hu, “Ear-phone: an end-to-end participatory urban noise mapping system,” in *IPSN*, 2010, pp. 105–116.

[24] C. A. Ratanamahatana and E. Keogh, “Making time-series classification more accurate using learned constraints,” in *Proceedings of the 2004 SIAM International Conference on Data Mining*, 2004, pp. 11–22.

[25] X. Tang, C. Wang, X. Yuan, and Q. Wang, “Non-interactive privacy-preserving truth discovery in crowd sensing application,” in *INFOCOM*, 2018.

[26] A. Vasudeva and M. Sood, “Survey on Sybil attack defense mechanisms in wireless ad hoc networks,” *Journal of Network and Computer Applications*, 2018.

[27] B. Wang, L. Zhang, and N. Z. Gong, “SybilSCAR: Sybil detection in online social networks via local rule based propagation,” in *INFOCOM*, 2017, pp. 1099–1107.

[28] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao, “Defending against Sybil devices in crowdsourced mapping services,” in *MobiSys*, 2016, pp. 179–191.

[29] Q. Wang, B. Ye, B. Tang, S. Guo, and S. Lu, “eBay in the clouds: False-name-proof auctions for cloud resource allocation,” in *ICDCS*, 2015, pp. 153–162.

[30] “WiFi Map,” https://www.wifimap.io/.

[31] K. H. Wong, Y. Zheng, J. Cao, and S. Wang, “A dynamic user authentication scheme for wireless sensor networks,” in *SUTC*, 2006, pp. 244–251.

[32] D. Yang, G. Xue, X. Fang, and J. Tang, “Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing,” in *MobiCom*, 2012, pp. 173–184.

[33] D. Yang, G. Xue, X. Fang, and J. Tang, “Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones,” *IEEE/ACM Trans. Netw.*, vol. 24, no. 3, pp. 1732–1744, 2016.

[34] X. Yin, J. Han, and S. Y. Philip, “Truth discovery with multiple conflicting information providers on the web,” *TKDE*, vol. 20, no. 6, pp. 796–808, 2008.

[35] X. Zhang, G. Xue, R. Yu, D. Yang, and J. Tang, “Truthful incentive mechanisms for crowdsourcing,” in *INFOCOM*, 2015, pp. 2830–2838.

[36] L. Zhu, C. Zhang, C. Xu, and K. Sharif, “Rtsense: Providing reliable trust-based crowdsensing services in cvcc,” *IEEE Network*, vol. 32, no. 3, pp. 20–26, 2018.