TOWARDS PRIVACY-PRESERVING AND SECURE CROWD SENSING IN THE INTERNET OF THINGS

by

Chenglin Miao

June 1st, 2020

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, The State University of New York
in partial fulfilment of the requirements for the
degree of

Doctor of Philosophy

Department of Computer Science and Engineering

The thesis of Chenglin Miao was reviewed by the following:

Dr. Lu Su
Associate Professor of Computer Science and Engineering
Thesis Advisor, Chair of Committee

Dr. Kui Ren
Professor of Computer Science and Engineering
Committee Member

Dr. Jing Gao
Associate Professor of Computer Science and Engineering
Committee Member

Dedication

To Mengdi and my parents.

Acknowledgments

I would like to thank all the people who gave me tremendous support and help during my Ph.D. study. Without their support and help, this thesis would be impossible.

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Lu Su, for his continuous support, invaluable guidance, and persistent encouragement. I feel so fortunate to work with Professor Su, and I have learned a lot from him over the past several years. He taught me not only the way of doing excellent research, but also how to give presentations, write proposals, and mentor students. His passion in research, teaching, and student supervision has motivated me to pursue a career in academia. I wish I could become a great advisor like him.

Next, I sincerely thank my other committee members, Professor Kui Ren and Professor Jing Gao, for their invaluable inputs. They have provided constructive comments and suggestions for my research projects. Their powerful knowledge and vision guided me to think deeper about my research. I am truly honored to have them in my doctoral committee. In addition, I would like to extend my gratitude to Professor Chunming Qiao, Professor Aidong Zhang, Professor Jinhui Xu, Professor Zhi Sun, and Professor Wenyao Xu for valuable discussions, suggestions, and collaborations.

I would also like to express my great appreciation to all my colleagues and friends who supported me during these years. I have enjoyed the time we spent together, and they make my Ph.D. journey a pleasant and exciting one. In particular, I would like to thank Wenjun Jiang, Xin Ma, Weida Zhong, Shiyang Wang, Yi Zhu, Yan Ju, Qi Li, Yaliang Li, Houping Xiao, Chuishi Meng, Fenglong Ma, Hengtong Zhang, Yaqing Wang, Tianqi Wang, Rui Li, Guangxu Xun, Hongfei Xue, Qiuling Suo, Liuyi Yao, Kishlay Jha, Ye Yuan, Zhan Qin, Chaowen Guan, Sixu Piao, Zihao Shan, Tianhang Zheng, Di Wang, Aosen Wang, Chen Song, Zhengxiong Li, Yan Sun, Chen Yuan, Haochen Hu, Rudra Prasad Baksi, and those I have not included their names here, for the collaborations and helpful advice.

Last but not least, I would like to thank my parents and my girlfriend for their continuous love, support and understanding. During the past years, they are with me all the time, sharing my pain and happiness. My degree is at the expense of their sacrifices.

Table of Contents

List of Tables

List of Figures

Abstract

Recent years have witnessed the rise of Internet of Things (IoT), a newly emergent networking paradigm that connects humans and the physical-world through ubiquitous sensing, computing, and communicating devices. Driven by the ubiquitous and interconnected sensing devices in the Internet of Things, crowd sensing has emerged as a new way of collecting information from the physical world. Recently, a large variety of crowd sensing systems 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, and many others. However, the sensory data provided by the participating users are usually not reliable, due to various reasons such as poor sensor quality, incomplete observations, and background noise.

To identify truthful values from the crowd sensing data, a lot of reliability-aware data aggregation mechanisms have been developed and they can automatically capture user reliability in the data aggregation process. Though able to improve the aggregation accuracy, existing reliability-aware data aggregation mechanisms fail to take into consideration the privacy and security issues in their design. On one hand, the sensory data provided by each individual user may contain sensitive information, which may be disclosed to others during the data aggregation process, resulting in the leakage of users' privacy. On the other hand, there may exist malicious users in crowd sensing systems who conduct the data poisoning attacks for the purpose of sabotage or financial rewards, and the effectiveness of the crowd sensing systems can be largely degraded by these malicious users.

In this thesis, we take steps to study and address the privacy and security issues when conducting reliability-aware data aggregation in crowd sensing systems. Specifically, we first consider a widely adopted reliability-aware data aggregation mechanism named truth discovery and propose a series of privacy-preserving truth discovery frameworks for crowd sensing systems. These frameworks can not only accurately calculate the final aggregated results but also provide strong privacy protection for the users' sensitive information. Then, we investigate crowd sensing in adversarial environments and study the data poisoning attacks against the crowd sensing systems employing the truth

discovery mechanism. We develop an optimal attack framework in which the attacker can not only maximize his attack utility but also disguise the introduced malicious users as normal ones such that they cannot be detected easily. Following a similar design philosophy, we also successfully attack the crowd sensing systems empowered with the Dawid-Skene model, another widely adopted reliability-aware data aggregation algorithm. The desirable performance of the proposed frameworks is verified through extensive experiments conducted on real-world crowd sensing systems.

Chapter 1

Introduction

Today, we are living in an interconnected world. Through ubiquitous sensing, computing, and communicating devices, now everything in the world is linked to each other, forming an Internet of Things (IoT) [2, 37]. With more than 30 billion[1] connected devices that pervade every corner of the world, IoT is able to facilitate a whole spectrum of civilian and military applications with enormous societal and economic impacts.

Driven by the ubiquitous and interconnected sensing devices in the Internet of Things, crowd sensing has emerged as a new way of collecting information from the physical world. In crowd sensing systems, humans work as sensor carriers or even the sensors, and the collection of sensory data is outsourced to a large crowd of participating users carrying sensing devices. Recently, a large variety of crowd sensing systems [14–17, 29, 32, 38–41, 47, 49, 55, 68, 81, 86, 87, 90, 107–109, 115] have been developed and they serve 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. The crowd-contributed sensory data in these applications have fundamentally changed the ways in

[1] http://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

which we learn about our world. However, they also pose great challenges on the design and development of crowd sensing systems.

1.1 Motivation

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 user reliability 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, a lot of reliability-aware data aggregation mechanisms [22, 48, 60–65, 70, 72, 94, 98, 102, 103, 112] have been developed and they are widely adopted in many crowd sensing applications. These mechanisms can automatically capture user reliability in the data aggregation process and bring significant improvement to the aggregation accuracy. However, they fail to consider an important issue in the design of crowd sensing systems, i.e., the privacy and security concerns related to the sensory data.

All the parties in a crowd sensing system, including not only the participating users who contribute sensory data but also the cloud server that collects the user-contributed

data, may be malicious. On one hand, in many crowd sensing applications, the data collected from each individual user may contain private personal information. The server may want to infer a user's private information from his sensory data, which makes users reluctant to contribute their data due to privacy concerns. On the other hand, the openness of the crowd sensing systems and the potential value of the collected sensory data offer both opportunities and incentives for malicious users to launch attacks. The participating users may launch malicious attacks through submitting malicious sensory data for the purpose of sabotage or financial rewards. Both the privacy and security issues can largely degrade the effectiveness of the crowd sensing systems in practice. Therefore, there is a great need for privacy-preserving and security mechanisms to protect users' private information from being disclosed as well as defend against malicious attacks.

1.2 Thesis Overview

In this thesis, towards the objective of enabling privacy-preserving and secure crowd sensing in the Internet of Things, we take steps to study and address the privacy and security issues when conducting reliability-aware data aggregation in crowd sensing systems. Specifically, we first consider a widely adopted reliability-aware data aggregation mechanism named *truth discovery* [60–65, 70, 72, 94, 98, 102, 103, 112], and present a series of privacy-preserving truth discovery frameworks for crowd sensing systems. Then we investigate the security vulnerability of truth discovery and study the data poisoning attacks against the crowd sensing systems empowered with this kind of data aggregation mechanism. Following a similar design philosophy, we also successfully attack the *Dawid-Skene model* [22], another widely adopted data aggregation algorithm in crowd sensing systems. In this section, we provide an overview of each work.

Privacy-Preserving Truth Discovery in Crowd Sensing Systems

As a sophisticated reliability-aware data aggregation mechanism, truth discovery has drawn significant attention recently. It involves the probability of a user providing accu-

rate data in the form of user weight when aggregating sensory data, and thus can make the aggregated results close to the information provided by reliable users. 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. However, these approaches fail to take into account the protection of user privacy.

In this work [73, 74], we propose a novel privacy-preserving truth discovery (PPTD) framework, which can protect not only users' sensory data but also their reliability scores derived by the truth discovery approaches. The key idea of the proposed framework is to perform weighted aggregation on users' encrypted data using homomorphic cryptosystem, which can guarantee both high aggregation accuracy and strong privacy protection. In order to deal with large-scale data, we also propose to parallelize PPTD with MapReduce framework. In addition, we design an incremental PPTD scheme for the scenarios where the sensory data are collected in a streaming manner. Extensive experiments based on two real-world crowd sensing systems demonstrate that the proposed framework can generate accurate aggregated results while protecting users' private information.

A Lightweight Privacy-Preserving Truth Discovery Framework

The PPTD framework can achieve strong privacy guarantee, however, at a cost of significant computation and communication overhead. The reason is that each user in this framework has to conduct considerable amount of ciphertext-based calculations and communication with the cloud server during the truth discovery procedure. In crowd sensing systems, the sensing device carried by each participating user usually has limited energy resources. Therefore, there is a great need to design a privacy-preserving truth discovery scheme which can not only guarantee high accuracy and strong privacy protection but also introduce little overhead to the participating users.

In the light of this need, in this work [77], we propose a lightweight privacy preserving truth discovery framework, L-PPTD, which is implemented by involving two non-colluding cloud platforms and adopting additively homomorphic cryptosystem. This framework not only achieves the protection of each user's sensory data and reliability information but also introduces little overhead to the users. In order to further reduce each user's overhead in the scenarios where only the sensory data need to be protected, we propose another more lightweight framework named L^2-PPTD. The desirable performance of the proposed frameworks is verified through extensive experiments conducted on real world crowd sensing systems.

Data Poisoning Attacks in Crowd Sensing Systems

In the aforementioned works, we consider the scenarios where the server that collects the user-contributed data is malicious, and it may want to infer a user's private information from his sensory data. In fact, in some cases, the participating users who contribute the sensory data may also be malicious. They may launch malicious attacks for the purpose of sabotage or financial rewards. One important form of attacks in crowd sensing systems is called *data poisoning*, where an attacker tries to degrade the effectiveness of the crowd sensing systems through creating or recruiting a group of malicious users and letting them submit manipulated sensory data.

Since truth discovery incorporates users' reliability into the aggregation procedure, it shows robustness to the data poisoning attacks. However, truth discovery is not perfect in all cases. In this work [76], we study how to effectively conduct two types of data poisoning attacks, i.e., the availability attack and the target attack, against a crowd sensing system empowered with the truth discovery mechanism. We develop an optimal attack framework in which the attacker can not only maximize his attack utility but also disguise the introduced malicious users as normal ones such that they cannot be detected easily. The desirable performance of the proposed framework is verified through extensive experiments conducted on a real-world crowd sensing system.

Security Vulnerability Analysis for the Dawid-Skene Model

Besides the truth discovery mechanism discussed in above sections, the Dawid-Skene model [22] is another widely adopted reliability-aware data aggregation algorithm in crowd sensing systems. By conducting maximum likelihood estimation (MLE) using the expectation maximization (EM) algorithm, the Dawid-Skene model can jointly estimate each user's reliability and conduct weighted aggregation. So it has the capability of identifying truthful information from noisy or conflicting crowd-contributed sensory data. To well understand the security vulnerability of the Dawid-Skene model to data poisoning attacks in crowd sensing systems, in this work [75], we investigate how to successfully attack this type of data aggregation method. Specifically, we follow the attacking philosophy for truth discovery and design an intelligent data poisoning attack mechanism, based on which the attacker can not only achieve maximum attack utility but also disguise the attacking behaviors. Extensive experiments based on real-world crowd-contributed data are conducted to verify the desirable properties of the proposed mechanism.

1.3 Thesis Organization

In the next four chapters, I will elaborate on the aforementioned privacy-preserving frameworks and attack mechanisms, shedding light on their design philosophy and desirable properties. Specifically,

- In Chapter 2, we propose a novel privacy-preserving truth discovery (PPTD) framework for crowd sensing systems. This framework can achieve the protection of not only users' sensory data but also their reliability scores derived by the truth discovery approaches. In addition, in order to deal with large-scale data and the scenarios where the sensory data are collected in a streaming manner, we also design a parallel PPTD scheme and an incremental PPTD scheme, respectively.

- In Chapter 3, to reduce each user's overhead while achieving the protection of both users' sensory data and their reliability information, we propose a lightweight privacy-preserving truth discovery framework, L-PPTD, which is implemented by involving two non-colluding cloud platforms and adopting additively homomorphic cryptosystem. For the scenarios where only the sensory data need to be protected, we propose another more lightweight framework named L^2-PPTD.

- In Chapter 4, we consider the security aspect of the crowd sensing systems, and study how to effectively conduct two types of data poisoning attacks, i.e., the availability attack and the target attack, against a crowd sensing system empowered with the truth discovery mechanism.

- In Chapter 5, to analyze the security vulnerability of the Dawid-Skene model to data poisoning attacks in crowd sensing systems, we design an intelligent data poisoning attack mechanism. Based on this mechanism, the attacker can not only achieve maximum attack utility but also disguise the attacking behaviors.

We review the related work in Chapter 6, and finally conclude the dissertation in Chapter 7.

Chapter 2

Privacy-Preserving Truth Discovery in Crowd Sensing Systems

2.1 Introduction

Although crowd sensing systems can serve a wide spectrum of applications that have significant societal and economic impacts, the sensory data collected by individual users are usually not reliable. The reasons include environment noise, the hardware quality, as well as the ways in which users use the hardware. A possible solution is to aggregate the sensory data of multiple users who observe the same objects (or events). When aggregating crowd sensing data, however, the traditional methods (e.g., average and voting) would not be able to derive accurate aggregated results, since they regard all the users equally. An ideal approach should have the capability to capture the difference in the quality of information among different participating users. However, the challenge here is that the reliability level (referred to as weight) of each user is usually unknown *a priori*. To address this challenge, the problem of truth discovery [60–65, 70, 72, 94, 98, 102, 103, 112], which aims at discovering truthful facts from unreliable data, has recently been widely studied. The common principle of truth discovery approaches is that a user will be assigned a higher weight if his data is closer to the aggregated results,

and the data of a user will be counted more in the aggregation procedure if she has a higher weight.

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. To-

wards this end, we propose a novel privacy-preserving truth discovery (PPTD) framework. This framework makes use of homomorphic cryptosystem [19], and can guarantee both high accuracy and strong privacy. The proposed PPTD framework works as follows. Each participating user will first send the encrypted summation of distances between his own observation values and the estimated aggregated values to the cloud server. Then, the cloud server updates users' weights in encrypted form without decrypting the received distances information, and sends the updated weight to each user. Next, each user calculates the ciphertexts of weighted data using the received encrypted weight. Finally, the final results are estimated by the cloud server based on the ciphertexts received from users. The advantage of our proposed framework is that it can accurately calculate the final aggregated results while protecting the privacy of user data, and at the same time, the weight information are not disclosed to any party.

Additionally, in order to deal with massive data, we design a parallel privacy-preserving truth discovery scheme using the MapReduce framework [23], and thus the privacy-preserving truth discovery procedure can be conducted in a parallel and distributed manner. As for the scenarios where the sensing data are collected in a streaming manner, we also propose an incremental privacy-preserving truth discovery scheme that can aggregate the sensory data in real time and introduce less computational overhead compared with the basic PPTD framework.

2.2 Problem Setting

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 2.6. In this chapter, we assume that all the parties are semi-honest [66], which means all the parties strictly follow the protocol we design, but each party will try to infer the private information of other parties based on the intermediate results he obtains during the execution of the protocol. Additionally, we assume that the parties in our framework have no collusions, which means they will not collude with each other outside the designed protocol. These assumptions are reasonable in most crowd sensing scenarios, since 1) the parties want to get correct results and thus would follow the protocol for their mutual benefits, and 2) crowd users usually do not know each other, and even they know each other they are probably not willing to disclose private information to others.

We formally define the problem targeted in this chapter 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. For each object, there is a ground truth which is not known by all the parties in the framework. Our goal is to let server $\mathcal{S}$ accurately calculate the estimated values

$\{x_m^*\}_{m=1}^M$ of the ground truths for all the objects based on the information collected from users. In this procedure, each observation value (i.e., x_m^k) should not be disclosed to any party except the user who provides this value (i.e., the k-th user). Also, the weight information $\{w_k\}_{k=1}^K$ should not be disclosed to any party in the system.

To solve this problem, we propose a 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.

2.3 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.

2.3.1 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:

$$w_k = f(\sum_{m=1}^{M} d(x_m^k, x_m^*)) \tag{2.1}$$

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 chapter, we adopt the weight calculation function of CRH [61, 65] as f due to its good practical performance:

$$w_k = \log\left(\frac{\sum_{k'=1}^{K}\sum_{m=1}^{M} d(x_m^{k'}, x_m^*)}{\sum_{m=1}^{M} d(x_m^k, x_m^*)}\right) \tag{2.2}$$

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 chapter, 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:

$$d(x_m^k, x_m^*) = \frac{(x_m^k - x_m^*)^2}{std_m} \tag{2.3}$$

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 estimated ground truth vector x_m^*:

$$d(x_m^k, x_m^*) = (x_m^k - x_m^*)^T(x_m^k - x_m^*) \tag{2.4}$$

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

$$x_m^* \leftarrow \frac{\sum_{k=1}^{K} w_k \cdot x_m^k}{\sum_{k=1}^{K} w_k} \tag{2.5}$$

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., Eq. (2.2));
    end
    for each object m do
        Update the estimated ground truth based on current weights (e.g.,
         Eq. (2.5));
    end
until Convergence criterion is satisfied;
return The estimated ground truths {x_m^*}_{m=1}^M;
```

2.3.2 Cryptographic Tools

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:

$$E_{pk}(m_1 + m_2) = E_{pk}(m_1) \oplus E_{pk}(m_2) \tag{2.6}$$

$$E_{pk}(a \cdot m_1) = a \otimes E_{pk}(m_1) \tag{2.7}$$

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 $\otimes$.

Threshold Paillier Cryptosystem

Although there are several additive homomorphic cryptographic schemes, we use the threshold variant of Paillier scheme [21] 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

$$c = E_{pk}(m) = g^m r^n mod\ n^2 \tag{2.8}$$

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 (2.6), (2.7) and (2.8), the homomorphic properties of this cryptosystem can be described as

$$\begin{aligned} E_{pk}(m_1 + m_2) &= E_{pk}(m_1) \cdot E_{pk}(m_2) \\ &= g^{m_1+m_2}(r_1 r_2)^n mod\ n^2 \end{aligned} \tag{2.9}$$

$$E_{pk}(a \cdot m_1) = E_{pk}(m_1)^a = g^{am_1} {r_1}^{an} mod\ n^2 \tag{2.10}$$

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 chapter, 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

$$c_i = c^{2\Delta sk_i} \tag{2.11}$$

where $\Delta = p!$. Then based on the combining algorithm in [21], at least t partial decryptions can be combined together to get the plaintext m.

2.4 Privacy-Preserving Truth Discovery Framework

In this section, we discuss the details of our novel privacy-preserving truth discovery (PPTD) framework.

2.4.1 PPTD Overview

Figure 2.1 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 pri-

vate decryption key has been divided and distributed to all parties (i.e., party i has got his private key share sk_i).

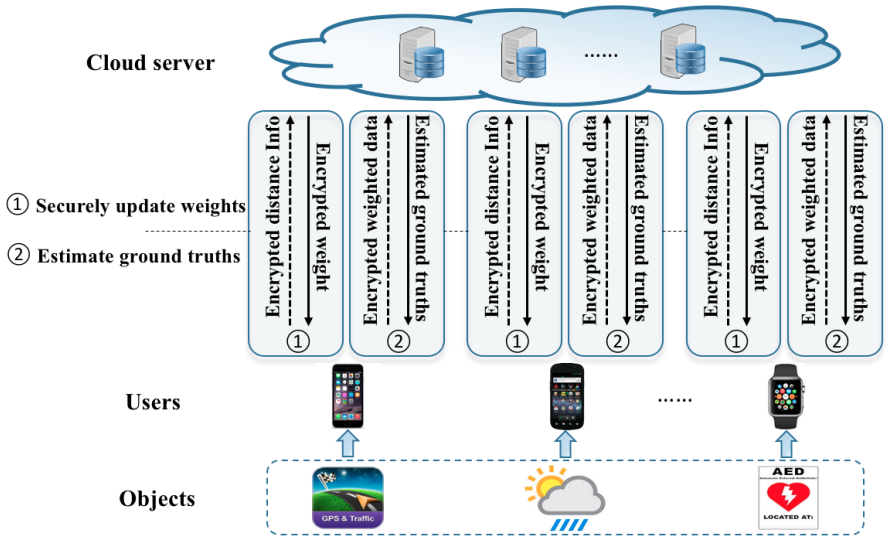


Figure 2.1: Privacy-preserving truth discovery framework

As shown in Figure 2.1, 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.

2.4.2 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.

Secure Sum Protocol

According to Eq. (2.2) and Eq. (2.5), 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 [19]. 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 Eq. (2.8), 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 Eq. (2.9);
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 Eq. (2.11) 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.

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 chapter 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 Eq.(2.2), the encrypted weight can be updated as follows

$$E_{pk}(\widetilde{w}_k) = E_{pk}(\lfloor L \cdot (log(\sum_{k'=1}^{K} Dist_{k'}) - log(Dist_k)) \rfloor) \tag{2.12}$$

where $Dist_k = \sum_{m=1}^{M} d(x_m^k, x_m^*)$ is the summation of distances between the k-th user's observation values $\{x_m^k\}_{m=1}^M$ and the estimated ground truths $\{x_m^*\}_{m=1}^M$. As we can see, in order for the cloud server to update $E_{pk}(\widetilde{w}_k)$, it needs to collect the information about $Dist_k$ from users. This procedure can be shown in Figure 2.2. 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 Eq. (2.4). But for continuous data, we need to know the standard deviation std_m according to Eq. (2.3), which is difficult to

derive without knowing the observation values of other users. Next, we first introduce the common steps (**W1** and **W6** in Figure 2.2) 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 Figure 2.2).

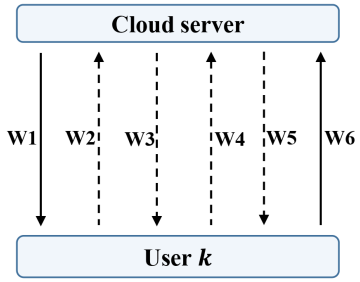


Figure 2.2: Secure weight update for user k

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 Eq. (2.8).

$$E_{pk}(\widetilde{Dist_k}) = g^{\widetilde{Dist_k}} r_{k1}^n \ mod \ n^2 \tag{2.13}$$

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

$$E_{pk}(\widetilde{\log Dist_k}) = g^{\widetilde{\log Dist_k}} r_{k2}^n \ mod \ n^2 \tag{2.14}$$

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 Eq. (2.8) as

$$E_{pk}(\widetilde{\log sum_D}) = g^{\widetilde{\log sum_D}} r_{s1}^n \ mod \ n^2 \tag{2.15}$$

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 Eq. (2.9), Eq. (2.10) and Eq. (2.12).

$$\begin{aligned} E_{pk}(\widetilde{w}_k) &= E_{pk}(\widetilde{\log sum_D}) \cdot E_{pk}(-\widetilde{\log Dist_k}) \\ &= E_{pk}(\widetilde{\log sum_D}) \cdot E_{pk}(\widetilde{\log Dist_k})^{-1} \end{aligned} \tag{2.16}$$

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 Eq. (2.8), 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.

Secure Truth Estimation

After updating user weights, the next thing is to estimate the ground truth for each object. As shown in Figure 2.3, there are two major steps in this phase, which are detailed as follows.

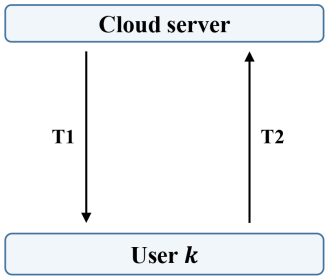


Figure 2.3: Secure truth estimation

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 Eq. (2.10) using the following formula:

$$E_{pk}(\widetilde{w}_k \cdot \widetilde{x}_m^k) = E_{pk}(\widetilde{w}_k)^{\widetilde{x}_m^k} \tag{2.17}$$

For categorical data, x_m^k is a vector as described in Section 2.3.1, so user k needs to calculate the ciphertext for each element in this vector as follows:

$$E_{pk}(\widetilde{w}_k \cdot x_m^k(i)) = \begin{cases} E_{pk}(0) & if\ x_m^k(i) = 0 \\ E_{pk}(\widetilde{w}_k) \cdot E_{pk}(0) & if\ x_m^k(i) = 1 \end{cases} \tag{2.18}$$

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 Eq. (2.5) 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 Eq. (2.5), 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 Eq. (2.5).

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.

2.4.3 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.

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 Eq. (2.15) and Eq. (2.16). 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 Eq. (2.17) and Eq. (2.18). 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 $\{x_m^*\}_{m=1}^{M}$;

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 2.4.2. Since users' weight information is also needed, we use an external file to store the encrypted weights, and all the Reducer nodes can read

it. Finally, for each object a key/value pair is outputted, where the key is object ID m and the value is the estimated ground truth. The two procedures (i.e., distributed weight update and parallel truth estimation) are iteratively conducted until the whole procedure converges.

Algorithm 2: Map function for estimating truths

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

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

Algorithm 3: Reduce function for estimating truths

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

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

2.4.4 Incremental PPTD

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

users in a "streaming" manner. In such scenarios, it is inefficient to infer the ground truth for each object until all the objects are observed. For example, in transportation monitoring applications [96], the traffic information is reported by multiple users in real time, it is inefficient to evaluate the route conditions until all the traffic information at different time periods is observed by the users. In healthcare applications [36], patients' health data are usually collected day by day, it is inefficient to analyze these data after several weeks or several months. In order to address this challenge, a possible way is to conduct the PPTD scheme once again on the whole data set whenever some new objects are observed. However, tremendous unnecessary calculations would be involved to iteratively update the estimated truths of the previous observed objects. To tackle this problem, we design an incremental privacy-preserving truth discovery (i.e., incremental PPTD) scheme that can timely estimate the ground truth of the newly observed object without disclosing the private information of each user and revisiting the old data.

Different from PPTD, we design the incremental PPTD by first conducting the truth estimation and then updating each user's weight when a new object is observed. For each user k, we use $D_k = \sum_{m=1}^{l-1} d(x_m^k, x_m^*)$ to denote his "old information" with respect to the previous observed $l-1$ objects. Here D_k represents the summation of the distances between user k's observations and the estimated truths for the $l-1$ objects observed in the past. When the l-th object is observed, the secure truth estimation phase and secure weight update phase are conducted as Figure 2.4.

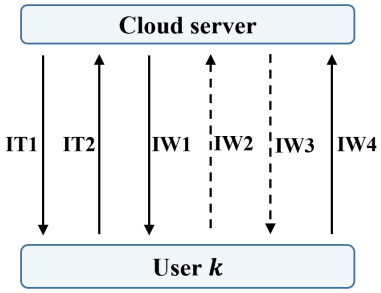


Figure 2.4: Incremental PPTD

Secure Truth Estimation

In this phase, the ground truth of the newly observed object is estimated based on user weights which are calculated based on the historical data. This phase contains two steps, i.e., IT1 and IT2 in Figure 2.4.

Step IT1. Server $\mathcal{S}$ sends user k the rounding parameter L and the encrypted weight $E_{pk}(\widetilde{w}_k)$, which is calculated by server $\mathcal{S}$ in the *secure weight update* phase for the $(l-1)$-th object. Then each user k calculates the ciphertext of the weighted data (i.e., $E_{pk}(\widetilde{w}_k \cdot \widetilde{x}_l^k)$ or $E_{pk}(\widetilde{w}_k \cdot x_l^k(i))$) for the l-th object according to Eq. (2.17) or Eq. (2.18). Here we use x_l^k to denote the observation value of user k for the l-th object, and $\widetilde{x}_l^k$ denotes its approximate value. For continuous data, user k also needs to encrypt his observation value as $E_{pk}(\widetilde{x}_l^k)$ in order to calculate the standard deviation std_l.

Step IT2. User k submits the ciphertext of weighted data to server $\mathcal{S}$. Similar to **Step T2** in Section 2.4.2, server $\mathcal{S}$ calculates the estimated truth (i.e., x_l^*) for the l-th object based on the secure sum protocol and Eq. (2.5). For continuous data, user k also needs to upload the encrypted data $E_{pk}(\widetilde{x}_l^k)$, and then the average observation value $\bar{x}_l$ is calculated based on the secure sum protocol.

Secure Weight Update

After the truth for the l-th object is estimated, each user's weight is securely updated in this phase. Similar to PPTD scheme, here we also need to consider continuous data and categorical data separately when calculating distances. We first introduce the common steps (IW1 and IW4 in Figure 2.4) for the two types of data, and then discuss the specific steps (IW2 and IW3 in Figure 2.4) to calculate the standard deviation std_l for continuous data.

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

$E_{pk}(\widetilde{\log D_k})$ according to Eq. (2.19) and Eq. (2.20):

$$E_{pk}(\widetilde{D_k}) = g^{\widetilde{D_k}} r'^{n}_{k1} \; mod \; n^2 \tag{2.19}$$

$$E_{pk}(\widetilde{\log D_k}) = g^{\widetilde{\log D_k}} r'^{n}_{k2} \; mod \; n^2 \tag{2.20}$$

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

Step IW4. User k submits both $E_{pk}(\widetilde{D_k})$ and $E_{pk}(\widetilde{\log D_k})$ to server $\mathcal{S}$. After receiving the ciphertexts from all users, $\mathcal{S}$ updates the encrypted weight $E_{pk}(\widetilde{w}_k)$ for user k based on **Step W6** in Section 2.4.2.

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

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

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

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

From Protocol 3, we can see when a new object is observed, the ground truth of this object can be estimated in real time without disclosing each user's private information. Since the "old information" of user k has been integrated in the value D_k, there is no need to revisit the observation values of the past objects when estimating the ground truth of the new observed object. Although the iterative procedure is not involved in this scheme and the two phases (i.e., secure truth estimation and secure weight update) are conducted only once, the weight of each user will converge to stabilization when the number of objects increases. Additionally, this incremental scheme can be easily modified to fit the scenario where the objects are grouped into sequential chunks, of which each may contains multiple objects. In such scenario, we repeat the two phases for each chunk and update D_k of user k as $D_k = D_k + \sum_{m=1}^{C_\epsilon} d(x_m^k, x_m^*)$, where C_ϵ is the number of objects in the ϵ-th chunk.

Protocol 3: Incremental Privacy-Preserving Truth Discovery Protocol

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

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

1 The cloud server $\mathcal{S}$ sends the encrypted weight $E_{pk}(\widetilde{w}_k)$ and the rounding parameter L to user k;

2 Each user $k \in \mathcal{K}$ calculates the ciphertexts of the weighted data according to Eq. (2.17) and Eq. (2.18). For continuous data, user k also needs to calculate the encrypted data $E_{pk}(\widetilde{x}_l^k)$. Then the ciphertexts of the weighted data and the encrypted data are submitted to server $\mathcal{S}$.

3 After receiving the ciphertexts from all users, server $\mathcal{S}$ estimates the ground truth x_l^* based on *Step IT2* and sends it to each user. For continuous data, the average value $\bar{x}_l$ should also be calculated and sent to users.

4 Each user $k \in \mathcal{K}$ updates D_k with $D_k + d(x_l^k, x_l^*)$, then calculates the ciphertexts $E_{pk}(\widetilde{D_k})$ and $E_{pk}(\widetilde{\log D_k})$ according to Eq. (2.19) and Eq. (2.20). Here the standard deviation std_l for continuous data is calculated based on *Step IW2* and *Step IW3*.

5 User k submits both $E_{pk}(\widetilde{D_k})$ and $E_{pk}(\widetilde{\log D_k})$ to server $\mathcal{S}$. Then $\mathcal{S}$ updates the encrypted weight $E_{pk}(\widetilde{w}_k)$ for user k based on **Step W6** in Section 2.4.2.

2.5 Privacy Analysis

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 at the same time, the weight of each user should not be known by any party. Since our framework is built upon the proposed secure sum protocol, we start with the privacy analysis of this protocol.

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

summation $\sum_{k=1}^{K} v_k$, based on which it cannot infer the input value v_k of each user. In this way, the privacy of each user's input value is guaranteed by this protocol.

Then we can summarize the privacy-preserving goal of our framework as Theorem 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 $\sum_{k=1}^{K} w_k \cdot x^k_m$, which are known by the cloud server. Since x^k_m is only known by the user, the server cannot infer w_k based on the above summations. So the weight information will not be disclosed to any party in this framework. □

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

2.6 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 2.7.

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 chapter, 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 Eq. (2.2). In the truth estimation procedure, user $k \in \mathcal{K}$ submits the ciphertexts of weighted data $\{E_{pk}(\widetilde{w_k \cdot x_m^k})\}_{m=1}^{M}$ and the encrypted weight $E_{pk}(\widetilde{w}_k)$. Then the cloud server can estimate ground truth for each object via the same method used in PPTD (step **T2**).

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

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

2.7 Performance Evaluation

In this section, we evaluate the proposed privacy-preserving truth discovery (PPTD) framework on both real-world crowd sensing systems and synthetic dataset.

2.7.1 Experiment Setup

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

- MAE: For continuous data, we use the mean of absolute error (MAE), i.e., $\frac{1}{M}\sum_{m=1}^{M}|x_m^* - \hat{x}_m^*|$, to measure the mean of absolute distance between the estimated results and ground truths. Here $\hat{x}_m^*$ denotes the ground truth of object m.

- $RMSE$: For continuous data, we also use the root of mean squared error ($RMSE$), i.e., $\sqrt{\frac{1}{M}\sum_{m=1}^{M}(x_m^* - \hat{x}_m^*)^2}$, to measure the accuracy. Compared with MAE, $RMSE$ can penalize more on the large distance and less on the small distance.

- $ErrorRate$: For categorical data, we calculate the percentage of mismatched values between estimated results and ground truths as $ErrorRate$.

The baseline approach we use in this experiment is the state-of-the-art truth discovery scheme, i.e., CRH [61, 65], 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 15-node Dell Hadoop cluster with Intel Xeon E5-2403 processor (4 $\times$ 1.80 GHz, 48GB RAM) as the "cloud".

2.7.2 Experiment on Crowdsourced Indoor Floorplan Construction System

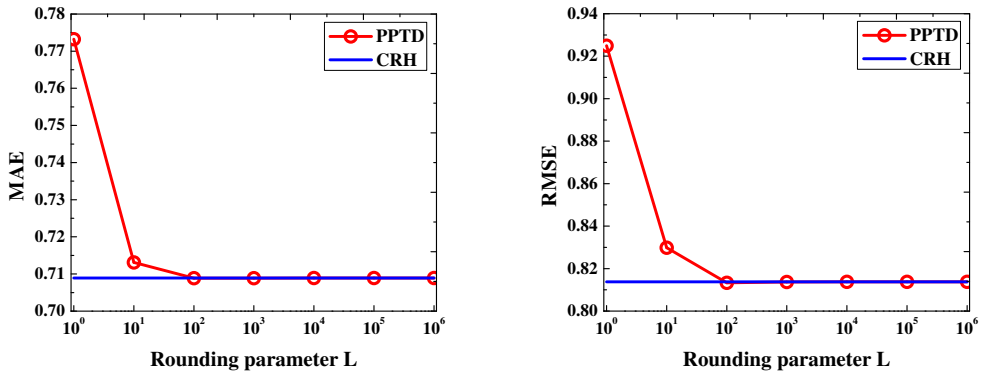
In this part, we show the experiment results on continuous data collected from a real-world crowd sensing system to demonstrate the advantages of PPTD. The application is crowdsourced indoor floorplan construction [10, 12, 33], which has recently drawn much attention since many location-based services 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 par-

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

ticular 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.

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.

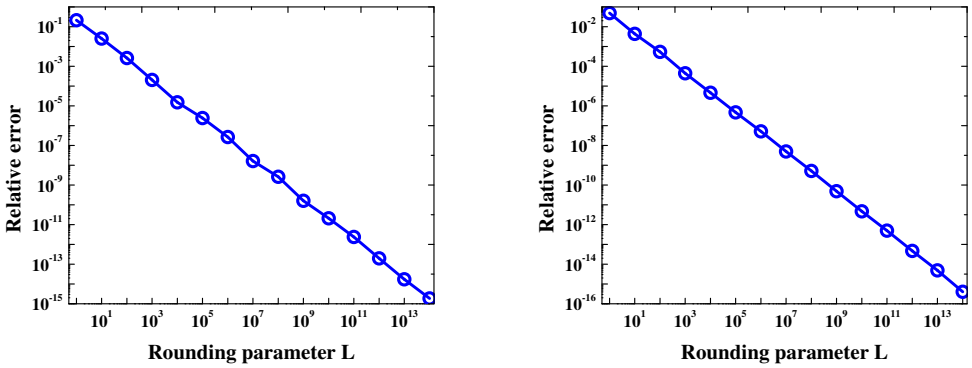


(a) MAE (b) RMSE

Figure 2.5: Ground truth estimation errors under different values of the parameter L

Figure 2.5 shows the ground truth estimation errors of our proposed framework and CRH under different values of the parameter L. The estimation error is measured in terms of MAE and $RMSE$, respectively. As seen in the figure, the estimation error of

PPTD is almost the same as that of CRH unless the rounding parameter L is too small (i.e., 10^0 or 10^1). This is because during the rounding procedure the fractional part (i.e., decimal digits) of the original value (e.g., $L \cdot \log Dist_k$) is dropped. In this sense, the smaller the parameter L, the more decimal digits of the original value will be lost. To measure the information loss degree, we calculate the relative estimation errors of PPTD and CRH in both object truth and user weight. Here we manually decrypt user weights for the analysis purpose. In particular, we define the relative error of user weight as $||\log \mathbf{w}_c - \log \mathbf{w}_p||/||\log \mathbf{w}_c||$, where $\mathbf{w}_c$ and $\mathbf{w}_p$ are the weight vectors of all the users obtained from CRH and PPTD, respectively. Similarly, we define the relative error of the estimated ground truths as $||\log \mathbf{x}_c^* - \log \mathbf{x}_p^*||/||\log \mathbf{x}_c^*||$, where $\mathbf{x}_c^*$ and $\mathbf{x}_p^*$ are obtained from CRH and PPTD respectively. The results are shown in Figure 2.6.



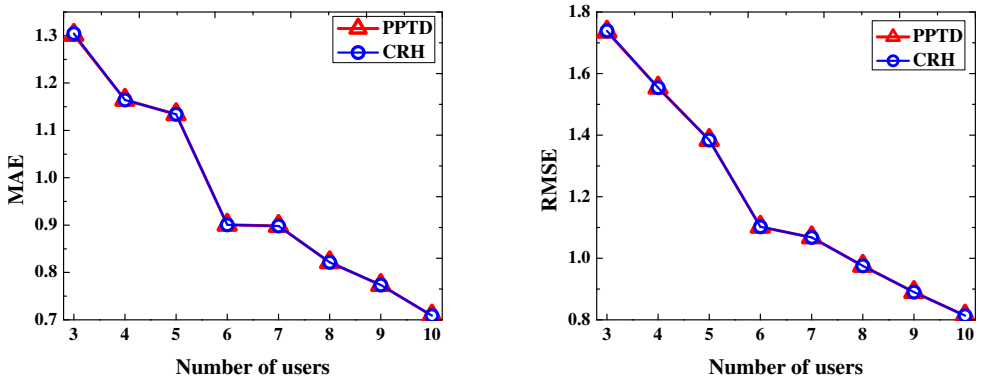
(a) Relative error of weight (b) Relative error of estimated truth

Figure 2.6: Relative errors under different values of L

As shown in Figure 2.6a and 2.6b, 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.

Figure 2.7a and 2.7b 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.

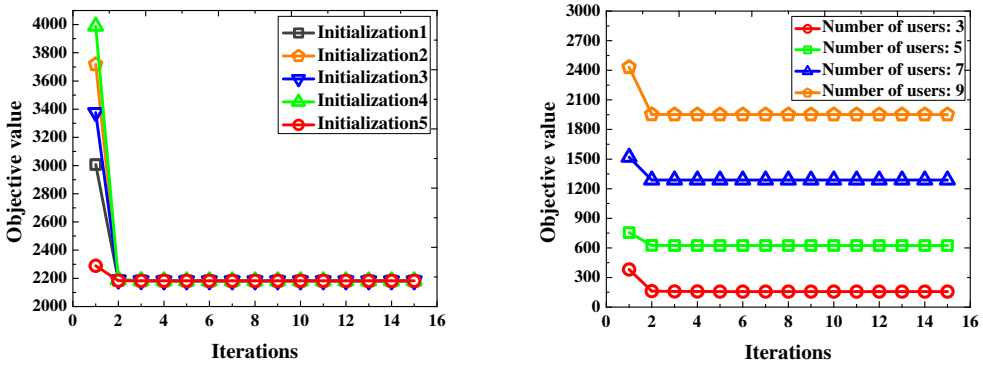


(a) MAE (b) RMSE

Figure 2.7: Ground truth estimation errors under different numbers of users

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}. We first set the number of users as 10 and calculate the *objective value* of the truth discovery problem, which is defined as the weighted summation of the distances between individual observations and the estimated ground truths (i.e., $\sum_{k=1}^{K} w_k \sum_{m=1}^{M} d(x_m^k, x_m^*)$). We repeat the experiment for 5 times with different random initialization values and report the evolution of the objective values in Figure 2.8a. As we can see, all the objective values, although under different initializations, converge quickly within just a few iterations. Then we evaluate the convergence of the privacy-preserving truth discovery procedure when the number of users varies. Here we consider four cases where the

number of users is set as 3, 5, 7 and 9, respectively, and for each case, the initialization values are randomly selected. The evolution of the objective values for the four cases are shown in Figure 2.8b, from which we can see all the objective values can converge within just a few iterations.



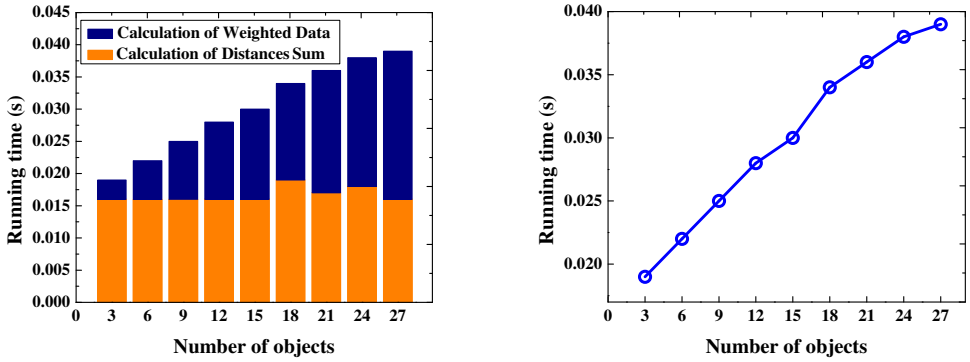
(a) For different initializations

(b) For different numbers of users

Figure 2.8: Convergence w.r.t Iterations

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.

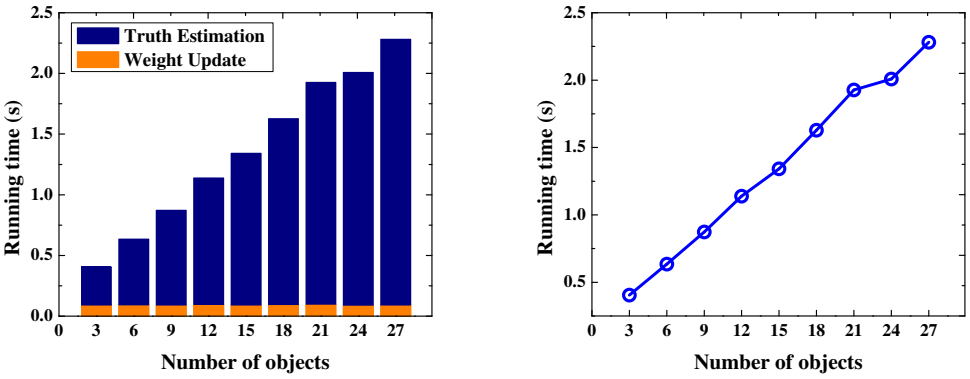
On the smartphone of each user, there are two major processing procedures: 1) calculating the encrypted summation of distances, and 2) calculating the ciphertexts of weighted observation values. In this experiment, we evaluate the running time of each procedure as well as the total running time under different object numbers ranging from 3 to 27. Figure 2.9a 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 2.9b gives the total time of the two procedures in each iteration. When the object number reaches 27, the total running time is only 0.039s, which is sustainable for the phone users. All the results in Figure 2.9 are averaged values derived from 10 smartphones.



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

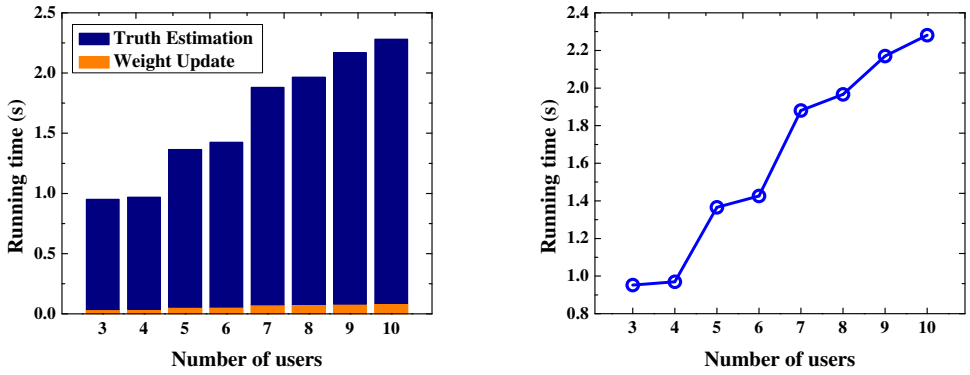
Figure 2.9: Running time w.r.t Number of objects for continuous data on smartphone

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 Figure 2.10a and Figure 2.11a, 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, Figure 2.10b and Figure 2.11b 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 2.10: 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 2.11: Running time w.r.t Number of users for continuous data on the cloud server

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

Communication and Energy Overhead. Compared with the baseline approach CRH, in which each user only needs to communicate with the cloud server once for uploading his sensory data, PPTD introduces more communication overhead due to the incorporated privacy-preserving scheme. To evaluate the communication overhead of PPTD, we measure the number of packets exchanged between the cloud server and all the crowd users. In this experiment, we use the change of the aforementioned objective value in two consecutive iterations as the convergence criterion and the threshold is set as 0.001. Figure 2.12 shows the numbers of exchanged packets over all users during the whole PPTD procedure, under different user numbers (i.e., K) from 3 to 10. As seen, the overall communication overhead is roughly $O(K)$. Actually, for each user, the average number of messages needed to be exchanged with the cloud server can be roughly calculated by $6(i+1)$, where i is the number of iterations during the PPTD

procedure. Considering that here we set a very conservative threshold which leads to average 6 iterations (much larger than the usual 2 or 3 iterations as shown in Figure 2.8a), 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 2.13 shows the average energy consumption percentage in one iteration for each user. When the object number reaches 27, the energy consumption percentage is only 0.000198% for each smartphone, which is acceptable for the phone users. The results in Figure 2.13 are averaged values derived from 10 smartphones in WiFi network environment.

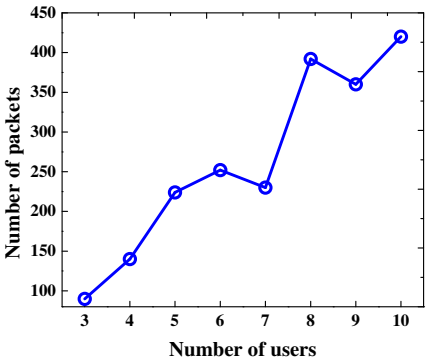


Figure 2.12: Communication overhead of PPTD

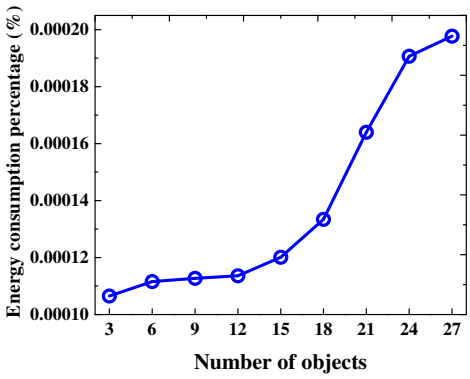


Figure 2.13: Energy consumption percentage on smartphone

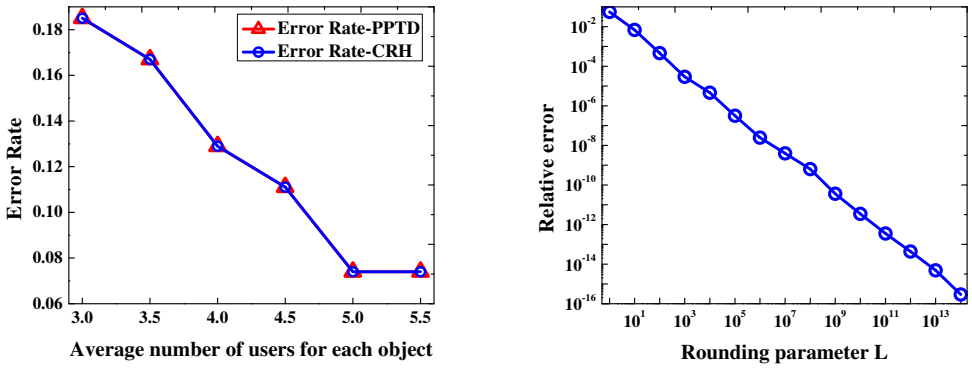
2.7.3 Experiment on Crowd Wisdom System

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. However, the uploaded answers for each question may be conflict due to various reasons. For example, some users may not have the background knowledge for some specific questions, and different users may have different understandings for the same question. In order to infer the true answer for each question, the cloud server needs to aggregate the answers from different users. To address the concern of some users that their private personal information could be inferred from their answers, we employ PPTD upon this crowd wisdom system, encrypting user answers before they are uploaded to the cloud server. Totally, 113 volunteers are employed as smartphone users and 54 questions are sent to them with candidate answers. For each question, there are four candidate answers and each user who receives this question needs to select one of the candidate answers as the correct answer. We use *Error Rate* as the evaluation metric and for the sake of evaluation, we have 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 Figure 2.14a, 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 2.7.2 to measure the errors introduced by L. The results are reported in Figure 2.14b, 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.



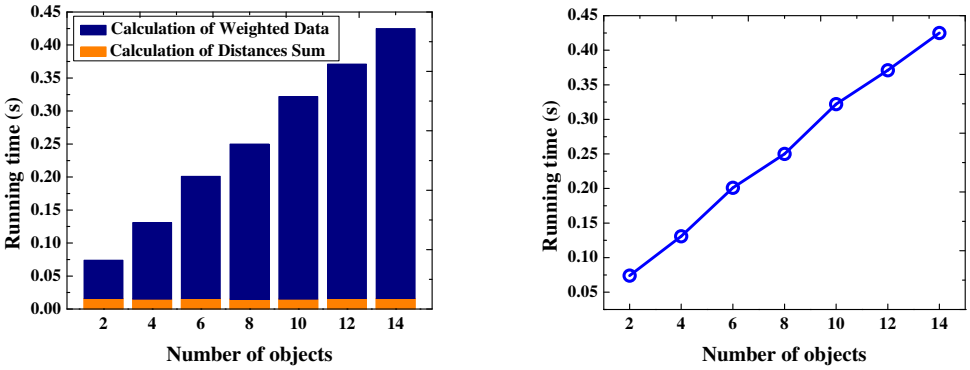
(a) Error rate (b) Relative error of weights

Figure 2.14: 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 2 to 14. The results are shown in Figure 2.15, from which we can see the second procedure (i.e., calculating weighted data) costs more time than the first procedure (Figure 2.15a). This is because most of the operations in the second procedure are conducted on ciphertexts while the first procedure is mainly composed of plaintext based operations. Additionally, Figure 2.15b 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.

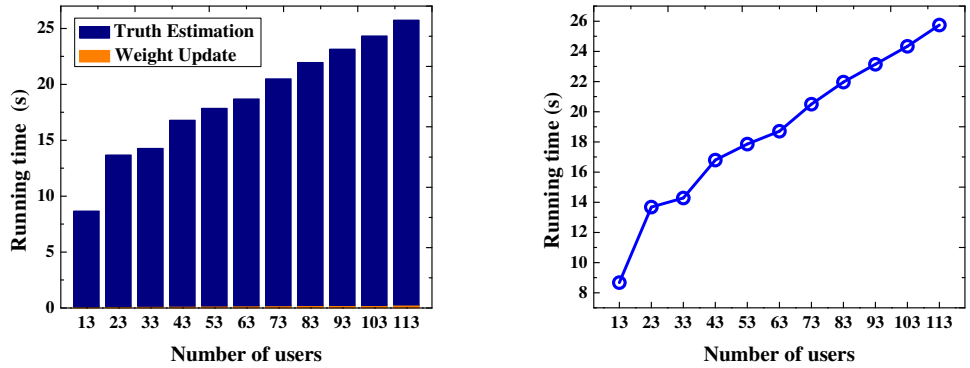
To evaluate the computational cost on the cloud server, we vary the number of users from 13 to 113 (the corresponding number of objects varies from 20 to 54, because each question is only answered by part of the users). Figure 2.16 reports the running time of each procedure and the total running time in each iteration. From Figure 2.16a we can



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

Figure 2.15: Running time w.r.t Number of objects for categorical data on smartphone

see that the computational time of updating truths is far greater than the time of updating weights for all the scenarios, which is similar to that in the experiment for continuous data. The evaluation of total running time in each iteration can be seen in Figure 2.16b. 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 2.16: Running time w.r.t Number of users for categorical data on the cloud server

2.7.4 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.

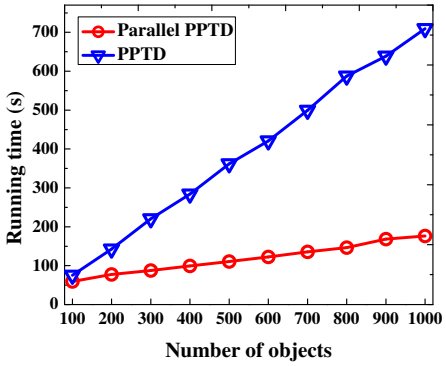


Figure 2.17: Running time w.r.t Number of objects for parallel privacy-preserving truth discovery

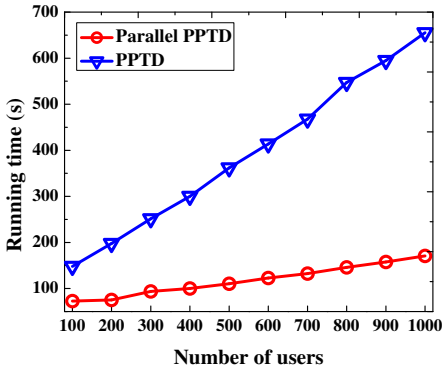


Figure 2.18: Running time w.r.t Number of users for parallel privacy-preserving truth discovery

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 Figure 2.17. 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 2.18 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 2.19 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.

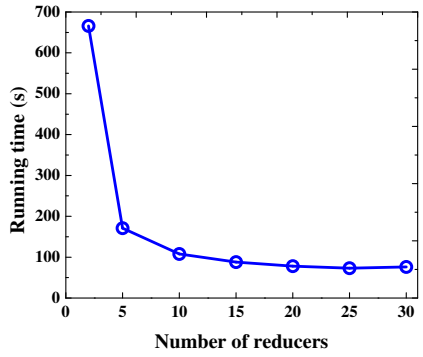


Figure 2.19: Running time w.r.t Number of reducers

2.7.5 Experiment of Incremental PPTD

In this section, we evaluate the performance of incremental PPTD scheme on both continuous data and categorical data which are collected from the crowdsourced indoor floorplan construction system and the crowd wisdom system respectively. Here we assume that the objects (i.e., segments in the hallway or questions) are observed in a "streaming" manner and only one object is observed at each timestamp. The baseline method we adopted is PPTD, i.e., we conduct the PPTD scheme once again on the whole

data set whenever a new object is observed. The rounding parameter L is still set as 10^{10} and we use I-PPTD to denote the proposed incremental PPTD scheme.

Performance on Crowdsourced Indoor Floorplan Construction System

We first compare the accuracy and computational cost of the incremental PPTD scheme with that of the baseline method (i.e., PPTD). In order to measure the estimation error of the proposed scheme, we fix the number of users as 10 and calculate MAE and RMSE for the estimated truths of the 27 objects, and the results are shown in Table 2.1. Although the estimation error of I-PPTD is a little higher than that of PPTD, the computational cost of I-PPTD is much less than that of the baseline method, which can be seen from Figure 2.20 and Figure 2.21.

Table 2.1: Accuracy of I-PPTD V.S. PPTD for continuous data

	MAE	RMSE
I-PPTD	0.7354	0.8342
PPTD	0.7170	0.8218

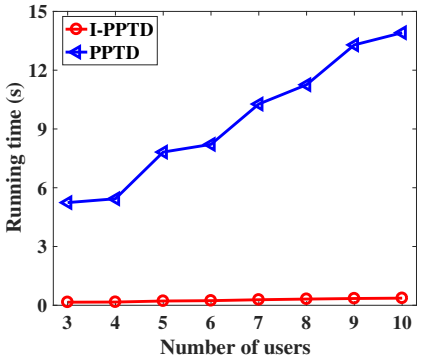


Figure 2.20: Running time of I-PPTD and PPTD w.r.t Number of users for continuous data on the cloud server

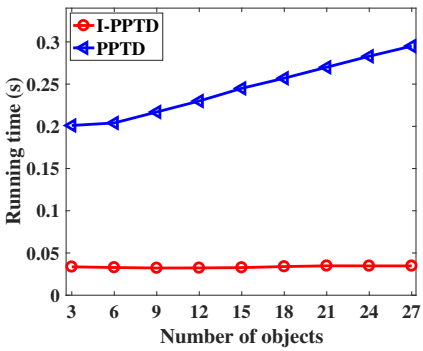


Figure 2.21: Running time of I-PPTD and PPTD w.r.t Number of objects for continuous data on smartphone

Figure 2.20 shows the average running time for each object on the cloud server when the number of users varies from 3 to 10. Here we conduct PPTD with 10 iterations in order to guarantee the convergence. From this figure, we can see the average running

time of I-PPTD is much less than that of PPTD on the cloud server, especially when the number of users becomes large. This is mainly because PPTD needs to revisit the old data and be conducted on all observed objects whenever a new object is observed, while I-PPTD only needs to estimate the truth of the new observed object. Additionally, at each timestamp, PPTD is iteratively conducted until convergence while I-PPTD only needs to be conducted once. Figure 2.21 reports the average running time for each object on smartphone while the number of objects is varying from 3 to 27. Here we also conduct PPTD with 10 iterations. This figure shows that the average running time of I-PPTD on smartphone is much less that of PPTD. The reason is similar with the above. Additionally, the averaging running time of I-PPTD is almost the same while that of PPTD is increasing when the number of objects varies from 3 to 27. The reason is that PPTD needs to estimate the truths of all the observed objects while I-PPTD only needs to estimate the truth for the new observed object at each timestamp.

In this experiment, we also evaluate the convergence of I-PPTD scheme. Here we manually decrypt user weights and report the results at different timestamps. Figure 2.22 shows the weights of 5 randomly selected users when the timestamp is varying from 1 to 27. From this figure we can see users' weights gradually become stable as the number of timestamps increases, which means the proposed I-PPTD scheme can guarantee convergence when sufficient objects are observed. To further illustrate this point, we also compare the weights of all users calculated by I-PPTD with those calculated by PPTD. The results are shown as Figure 2.23. Here we report user weights calculated by I-PPTD at timestamp 5, 15 and 25. We also report the weight values calculated by PPTD scheme after all the objects are observed. From the result we can see although user weights calculated by I-PPTD deviate from the baseline values at the first few timestamps, they gradually converge to the values calculated by PPTD as time goes on. That is to say, I-PPTD can achieve similar accuracy with PPTD when sufficient objects are observed.

Performance on Crowd Wisdom System

In this part, we evaluate the performance of I-PPTD on categorical data. We first eval-

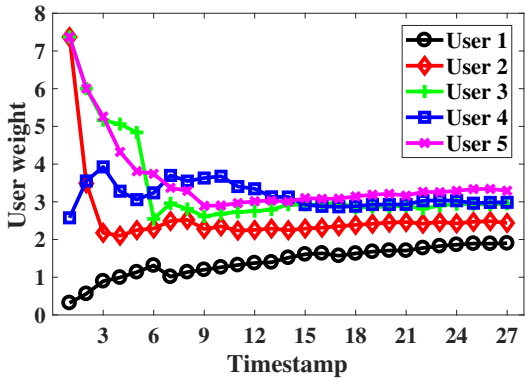


Figure 2.22: Convergence of I-PPTD on continuous data

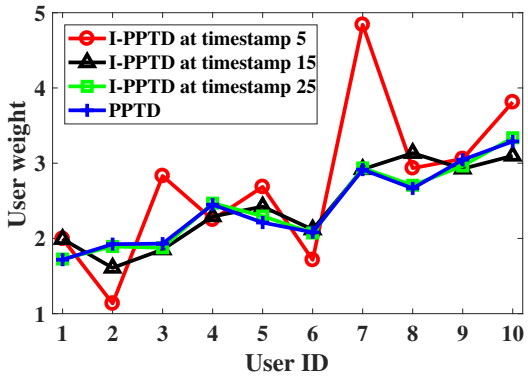


Figure 2.23: User weights calculated by I-PPTD and PPTD

uate the accuracy of I-PPTD by calculating the error rate of the estimated truths. The number of users is fixed as 113. After observing the 54 questions, we find both the error rates of I-PPTD and PPTD are 0.074, which verifies that I-PPTD could guarantee high accuracy while protecting users' privacy on streaming categorical data.

Similar to the experiment on continuous data, we also evaluate the computational cost of I-PPTD on categorical data and compare it with PPTD. In order to guarantee the convergence, we conduct PPTD with 2 iterations whenever a new object is observed in this experiment. We first evaluate the computational cost of I-PPTD on the cloud server. We vary the number of users from 13 to 113 and calculate the average running time for each newly observed object. The results are shown as Figure 2.24, from which we can see the computational cost of I-PPTD is much less than that of PPTD as the number of users increases. The reason is similar to that for continuous data. Then we evaluate

the computational cost of I-PPTD on the user side. We vary the number of objects observed by each user from 2 to 14 and calculate average running time for each newly observed object on smartphone. Figure 2.25 reports the results, from which we can see the computational cost of I-PPTD on the user side is also much less than that of PPTD. The experimental results on categorical data further verify that I-PPTD scheme is much more efficient than PPTD scheme when the data is collected in a streaming manner.

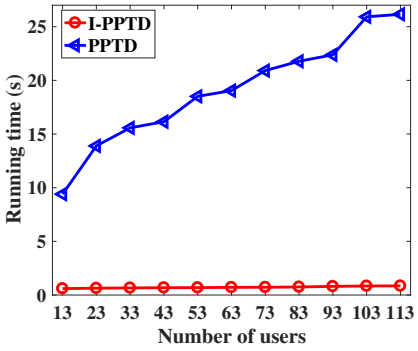


Figure 2.24: Running time of I-PPTD and PPTD w.r.t Number of users for categorical data on the cloud server

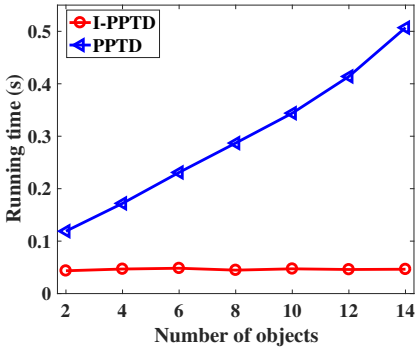


Figure 2.25: Running time of I-PPTD and PPTD w.r.t Number of objects for categorical data on smartphone

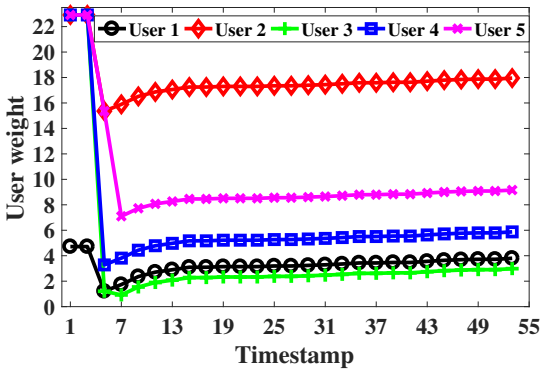


Figure 2.26: Convergence of I-PPTD on categorical data

In order to evaluate the convergence of I-PPTD on categorical data, we randomly select 5 users and manually decrypt their weights at different timestamps. The result is shown as Figure 2.26, from which we can see the weight of each user is gradually converging to stable as the number of observed questions increases. This confirms that

the proposed I-PPTD scheme can guarantee convergence when sufficient object are observed.

2.8 Summary

In this chapter, we design a novel 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. Additionally, we design an incremental PPTD scheme to deal with the scenarios where the sensory data of different objects are collected in a streaming manner. Theoretical analysis demonstrates that the observation values of each user will not be disclosed to others and the weight of each user will not be known by any party based on the proposed framework. Through extensive experiments on both real-world crowd sensing systems and synthetic data, we demonstrate that the proposed framework can not only generate accurate aggregated results but also guarantee the introduced computational and communication overhead are within the realm of practicality.

Chapter 3

A Lightweight Privacy-Preserving Truth Discovery Framework

3.1 Introduction

The privacy-preserving truth discovery (PPTD) framework designed in Chapter 2 can achieve strong privacy guarantee, however, at a cost of significant computation and communication overhead. The reason is that each user in this framework has to conduct considerable amount of ciphertext-based calculations and communication with the cloud server during the truth discovery procedure. In crowd sensing systems, the sensing device carried by each participating user usually has limited energy resources. Therefore, there is a great need to design a privacy-preserving truth discovery scheme which can not only guarantee high accuracy and strong privacy protection but also introduce little overhead to the participating users.

In the light of this need, in this chapter we propose a lightweight privacy-preserving truth discovery framework (L-PPTD) for crowd sensing systems, in which the sensory data and reliability information of each user are both protected from being disclosed to others. The proposed framework is implemented by involving two non-colluding cloud platforms and adopting additively homomorphic cryptosystem. In this framework, the

aggregated results (referred to as *truth*) are cooperatively estimated by the two cloud platforms without disclosing any user's private information. Additionally, instead of directly encrypting the data to be uploaded, each user in this framework preserves her privacy through perturbing the data with some random numbers, and all the ciphertext-based calculations are moved onto the cloud side, which substantially reduce the overhead incurred on each user.

Although L-PPTD can achieve tremendous reduction of the overhead on the user side, each user is still responsible for calculating her weight so as to protect reliability information. To further reduce the workload of users, we propose a more lightweight framework (L^2-PPTD) suiting for the scenarios where only the sensory data of each user need to be protected. In this framework, all each user needs to do is just uploading the perturbed sensory data and random numbers before the truth discovery procedure starts.

3.2 Problem setting

In this section, we formulate the problem addressed by the proposed lightweight privacy-preserving truth discovery framework. This framework consists of two different types of parties: *data requester* and *participating users*. The data requester is an individual or organization who posts sensing tasks which usually require the observations on a collection of *objects* (e.g., the potholes or litters in geotagging campaigns), while the participating users are a group of mobile device users who carry out the sensing tasks and generate sensory data with their mobile devices.

In crowd sensing systems, the sensory data and the reliability information of each participating user may be disclosed to the data requester or other users during the data aggregation process, resulting in the leakage of users' privacy. Here we mainly consider the attacks coming from the inside malicious parties (data requester or participating users). For the sake of curiosity and financial purpose, the data requester may try to infer the sensitive personal information of each participating user. On the other hand,

each participating user may also want to know the private information of other users. In this chapter, we still adopt the *semi-honest threat model*, in which all the parties will strictly follow the designed protocols, but each of them may backup all the data she has sent and received, and then try to learn the private information of other parties. Additionally, we assume there is no collusion in the designed framework, which means the parties will not collude with each other outside the designed protocols.

The problem addressed in this chapter is formulated as follows: Suppose there are M objects in the posted sensing task, denoted as $O = \{o_1, o_2, ..., o_M\}$, and these objects will be observed by K participating users represented as $U = \{u_1, u_2, ..., u_K\}$. We use $W = \{w_1, w_2, ..., w_K\}$ to denote the weights (i.e., reliability) of these users. Let x_m^k denote the sensory data of user u_k for object o_m. For every object $o_m \in O$, there is a ground truth which is not known by all the parties in this framework. Our goal is *to calculate the estimated values* $\{x_m\}_{m=1}^M$ *of the ground truths for all the objects while protecting the sensory data and reliability information of each user from being disclosed to others.*

3.3 Preliminary

In this section, we will review the concepts and general procedure of truth discovery and additively homomorphic encryption, which are the two major techniques adopted in our proposed framework.

3.3.1 Truth Discovery

As described in Section 2.3.1. The truth discovery approaches usually take a two-step iterative procedure:

Weight Estimation

In this step, each user's weight will be estimated based on the difference between its sensory data and the estimated truths. Typically, the weight of a user u_k is calculated

as $w_k = f(\sum_{m=1}^{M} d(x_m^k, x_m))$, where f is a monotonically decreasing function, and $d(x_m^k, x_m)$ is the distance function which is used to measure the difference between users' sensory data and the estimated truths.

In the proposed framework, we consider both cases when the sensory data are continuous or categorical. For continuous data, the squared distance function $d(x_m^k, x_m) = (x_m^k - x_m)^2$ is adopted. For categorical data, we assume each task has multiple candidate results or answers. Then the sensory data $x_m^k = (0, ..., \underset{q}{1}, ..., 0)^T$ represents user k selects the q-th result or answer for object o_m. In this case, the distance function is defined as $d(x_m^k, x_m) = (x_m^k - x_m)^T (x_m^k - x_m)$.

In this chapter, we aim to develop a general framework that is compatible with different types of function f. Without loss of generality, we will first use the following logarithmic weight function adopted in the widely used truth discovery method CRH [61,65] as an example when presenting our framework

$$w_k = \log \left(\frac{\sum_{k'=1}^{K} \sum_{m=1}^{M} d(x_m^{k'}, x_m)}{\sum_{m=1}^{M} d(x_m^k, x_m)} \right), \tag{3.1}$$

and then discuss the generalization of the proposed framework to other weight functions.

Truth Estimation

After user weights are calculated, the ground truth for each object o_m can be estimated as

$$x_m = \frac{\sum_{k=1}^{K} w_k x_m^k}{\sum_{k=1}^{K} w_k}. \tag{3.2}$$

When the sensory data is continuous, this value is actually the weighted average of the users' observations on object o_m. But when the data is categorical, x_m is a vector in which each element represents the probability of a particular candidate result or answer being the truth. The estimated truth of object o_m will be the result or answer with the largest value in vector x_m.

In truth discovery algorithms, the above two steps will be iteratively conducted until some convergence criterion is satisfied. The convergence criterion can be a predefined

iteration number or a threshold of the change in the estimated truths in two consecutive iterations.

3.3.2 Additively Homomorphic Encryption

Let $\mathcal{M}$ denote the message space, an encryption scheme is said to be additively homomorphic if the encryption function E satisfies

$$\forall\, m_1, m_2 \in \mathcal{M}, \quad E[m_1 + m_2] = E[m_1] \oplus E[m_2] \tag{3.3}$$

$$\forall\, m_3 \in \mathcal{M}, \quad E[a \cdot m_3] = a \otimes E[m_3] \tag{3.4}$$

for some operators $\oplus$ and $\otimes$, where a is a constant. In other words, by using the additively homomorphic cyptosystem, the encrypted sum of messages can be directly calculated from the ciphertexts of these messages.

In this chapter, we adopt a widely used additively homomorphic cryptosystem, namely, Paillier's cryptosystem [80], in which the message $m \in \mathbb{Z}_n$ ($\mathbb{Z}_n$ is the set of integers modulo the large positive integer n) can be encrypted as $E[m] = g^m r^n mod\ n^2$ with the public key $pk = (n, g)$, where $r \in \mathbb{Z}_n^*$ ($\mathbb{Z}_n^*$ denotes the multiplicative group of invertible elements of $\mathbb{Z}_n$) is privately and randomly selected by the user who calculates the ciphertexts. Then, $\forall\, m_1, m_2, m_3 \in \mathbb{Z}_n$, the following equations show the additively homomorphic properties of Paillier's cryptosystem:

$$D(E[m_1 + m_2]) = D(E[m_1] \cdot E[m_2]) = m_1 + m_2 \tag{3.5}$$

$$D(E[a \cdot m_3]) = D(E[m_3]^a) = a \cdot m_3 \tag{3.6}$$

where $D(\cdot)$ is the decryption function with the private key sk.

3.4 L-PPTD Framework

In this section, we will first introduce the L-PPTD framework, and then analyze its computational complexity and communication overhead.

In this framework, we aim to protect the sensory data and reliability information of each user from being disclosed to others while accurately estimating the object truths. To achieve this goal, two non-colluding cloud platforms are involved as the third parties in L-PPTD. As shown in Figure 3.1, we use S_A and S_B to denote the two cloud platforms respectively. The key idea of L-PPTD is to calculate the summed distances (i.e., $\sum_{k=1}^{K}\sum_{m=1}^{M} d(x_m^k, x_m)$) used to estimate user weights through the cooperation of S_A and S_B without letting S_A or S_B know the raw data of each user. With this distance information, each user will be able to calculate her weight by herself. After the weight of each user is derived, the object truth can be estimated via a similar cooperation between S_A and S_B. The final estimated object truths will be forwarded to the data requester.

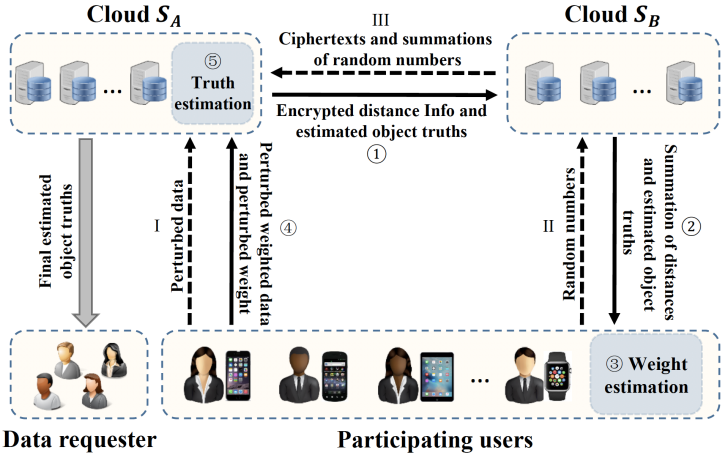


Figure 3.1: The workflow of L-PPTD

3.4.1 The Detailed Procedure of L-PPTD

As shown in Figure 3.1, L-PPTD is mainly composed of two phases: *Initialization Phase* and *Iteration Phase*. Next, we will elaborate on each step in the two phases.

Initialization Phase

This phase will be conducted only once during the whole truth discovery procedure. Firstly, cloud S_B generates the public key pk and private key sk. The public key can be known by all the parties while the private key is only known by S_B. Then the following steps are taken.

Step I**:** Each user u_k generates random numbers $\{\alpha_m^k\}_{m=1}^M$, $\{\beta_m^k\}_{m=1}^M$ and γ_k, which are used to perturb her sensory data, weighted data and weight respectively. For categorical data, α_m^k and β_m^k are two vectors in which each element is a random. Then u_k perturbs the sensory data x_m^k as $\tilde{x}_m^k = x_m^k - \alpha_m^k$, and uploads all the perturbed data $\{\tilde{x}_m^k\}_{m=1}^M$ to cloud S_A.

Step II**:** Each user uploads all the random numbers generated in step I to cloud S_B. Thus, the random numbers of each user are only known by herself and cloud S_B.

Step III**:** After receiving the random numbers from each user, cloud S_B calculates the ciphertexts $\{E[\alpha_m^k]\}_{k,m=1}^{K,M}$ and summations $\{\sum_{k=1}^K \beta_m^k\}_{m=1}^M$, $\sum_{k=1}^K \gamma_k$. For categorical data, $E[\alpha_m^k]$ represents a ciphertext vector in which each element is the ciphertext of the corresponding element in vector α_m^k. Then the ciphertexts and the summations are sent to cloud S_A.

Please note that when the data needed to be encrypted in this chapter is not integer, we will round it by multiplying a parameter R (a magnitude of 10) and at last the original data will be recovered by dividing the same parameter.

Iteration Phase

This phase starts with the random initialization of the object truths on cloud platform S_A.

Step ①**:** Based on the homomorphic property of the Paillier cryptosystem, cloud S_A firstly calculates the ciphertext C_{conti} for continuous data as

$$
\begin{aligned}
C_{conti} =& E[\sum_{k=1}^{K}\sum_{m=1}^{M}(x_m^k - x_m)^2 - \sum_{k=1}^{K}\sum_{m=1}^{M}(\alpha_m^k)^2] \\
=& \prod_{k=1}^{K}\prod_{m=1}^{M} E[(x_m^k - x_m)^2 - (\alpha_m^k)^2] \\
=& \prod_{k=1}^{K}\prod_{m=1}^{M} E[(x_m^k - \alpha_m^k - x_m)^2 + 2\alpha_m^k(x_m^k - \alpha_m^k - x_m)] \\
=& \prod_{k=1}^{K}\prod_{m=1}^{M} \{E[(\tilde{x}_m^k - x_m)^2] \cdot E[\alpha_m^k]^{2(\tilde{x}_m^k - x_m)}\}
\end{aligned}
\tag{3.7}
$$

where $E[\alpha_m^k]$ is received from cloud S_B. Based on the perturbed sensory data $\tilde{x}_m^k$ received from the k-th user, and the estimated truth x_m for each object o_m calculated in the previous iteration (x_m is randomly initialized in the first iteration), $E[(\tilde{x}_m^k - x_m)^2]$ is calculated with public key pk.

For categorical data, we assume x_{mi}^k, α_{mi}^k, $\tilde{x}_{mi}^k$ and x_{mi} represents the i-th element of vector x_m^k, α_m^k, $\tilde{x}_m^k$ and x_m respectively, and the number of elements in each vector is l. Similarly, The ciphertext C_{cate} can be calculated by S_A as

$$
\begin{aligned}
C_{cate} =& E[\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{i=1}^{l}(x_{mi}^k - x_{mi})^2 - \sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{i=1}^{l}(\alpha_{mi}^k)^2] \\
=& \prod_{k=1}^{K}\prod_{m=1}^{M}\prod_{i=1}^{l} E[(x_{mi}^k - x_{mi})^2 - (\alpha_{mi}^k)^2] \\
=& \prod_{k=1}^{K}\prod_{m=1}^{M}\prod_{i=1}^{l} \{E[(\tilde{x}_{mi}^k - x_{mi})^2] \cdot E[\alpha_{mi}^k]^{2(\tilde{x}_{mi}^k - x_{mi})}\}.
\end{aligned}
\tag{3.8}
$$

At last, the ciphertext C_{conti} or C_{cate} together with the estimated object truths $\{x_m\}_{m=1}^M$ are sent to cloud S_B.

Step ②: After receiving the ciphertext, cloud S_B decrypts it with his private key sk and calculates the summation of distances (i.e., $\sum_{k'=1}^{K}\sum_{m=1}^{M} d(x_m^{k'}, x_m)$ in Eq. (3.1)) by adding the value $\sum_{k=1}^{K}\sum_{m=1}^{M}(\alpha_m^k)^2$ (for categorical data, the

value is $\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{i=1}^{l}(\alpha_{mi}^{k})^2)$ to the decrypted data. Then the summation $\sum_{k=1}^{K}\sum_{m=1}^{M} d(x_m^k, x_m)$ together with the estimated object truths $\{x_m\}_{m=1}^{M}$ are sent to each user.

Step ③: In this step, each user u_k first calculates $\sum_{m=1}^{M} d(x_m^k, x_m)$ based on the estimated object truths $\{x_m\}_{m=1}^{M}$ received from cloud S_B, then estimates her weight w_k according to Eq. (3.1).

Step ④: After the weight is estimated, each user u_k calculates the weighted data $w_k x_m^k$ for object o_m and perturbs it as $w_k x_m^k - \beta_m^k$, where β_m^k is the random number (random number vector for categorical data) generated in the initialization phase. Additionally, the weight w_k is perturbed as $w_k - \gamma_k$. Then the perturbed weighted data (i.e., $w_k x_m^k - \beta_m^k$) and perturbed weight (i.e., $w_k - \gamma_k$) are sent to cloud S_A.

Step ⑤: Based on the information received from all the participating users, cloud S_A first calculates the value $\sum_{k=1}^{K}(w_k x_m^k - \beta_m^k)$ for each object o_m, then derives the summation of the weighted data as

$$\sum_{k=1}^{K} w_k x_m^k = \sum_{k=1}^{K}(w_k x_m^k - \beta_m^k) + \sum_{k=1}^{K} \beta_m^k \tag{3.9}$$

where the value $\sum_{k=1}^{K} \beta_m^k$ is received from cloud S_B in the initialization phase. Similarly, the summation of all users' weights is calculated as $\sum_{k=1}^{K} w_k = \sum_{k=1}^{K}(w_k - \gamma_k) + \sum_{k=1}^{K} \gamma_k$. With all the above information, S_A is able to estimate the object truth x_m according to Eq. (3.2).

In this phase, step ① $\sim$ ⑤ are repeated until the convergence criterion is satisfied. The final estimated truth for each object will be sent to the data requester by cloud S_A. Please note that, for categorical data, the final result for each object o_m should be the candidate answer with the largest value in the vector x_m calculated in the final iteration.

In the procedure of L-PPTD, each user only communicates with the two cloud platforms once respectively in each iteration and all the calculations on the user side are

based on plaintexts. Thus, very little overhead will be introduced to each user, which is confirmed by the experimental results presented in Section 3.6.

3.4.2 Security Analysis

The security goal of L-PPTD can be summarized as Theorem 2, followed by the proof.

Theorem 2. *Suppose the number of participating users satisfies $K \geq 3$ and for each object, there are at least two users providing different sensory data. If the parties (including the two cloud platforms) are semi-honest and there is no collusion among them, the sensory data and weight information of each user will not be disclosed to any other party under the L-PPTD framework.*

Proof. Firstly, we prove the security of users' private information on the cloud side. For cloud S_A, besides the ciphertexts $\{E[\alpha_m^k]\}_{k,m=1}^{K,M}$, he knows the plaintexts $\{x_m^k - \alpha_m^k\}_{k,m=1}^{K,M}$, $\{w_k x_m^k - \beta_m^k\}_{k,m=1}^{K,M}$, $\{w_k - \gamma_k\}_{k=1}^{K}$, $\{\sum_{k=1}^{K} \beta_m^k\}_{m=1}^{M}$, $\sum_{k=1}^{K} \gamma_k$ and $\{x_m\}_{m=1}^{M}$. Since the private key sk is only known by cloud S_B, cloud S_A cannot decrypt the ciphertexts. Although the values $\{\sum_{k=1}^{K} \beta_m^k\}_{m=1}^{M}$ and $\sum_{k=1}^{K} \gamma_k$ are known by cloud S_A, he cannot learn anything about the individual random numbers just based on these summations. In this way, as long as the two cloud platforms do not collude with each other, cloud S_A cannot infer the plaintexts of $\{x_m^k\}_{k,m=1}^{K,M}$, $\{w_k x_m^k\}_{k,m=1}^{K,M}$ and $\{w_k\}_{k=1}^{K}$. For cloud S_B, he knows the plaintexts of $\{\alpha_m^k\}_{k,m=1}^{K,M}$, $\{\beta_m^k\}_{k,m=1}^{K,M}$, $\{\gamma_k\}_{k=1}^{K}$, $\{x_m\}_{m=1}^{M}$ and $\sum_{k=1}^{K}\sum_{m=1}^{M} d(x_m^k, x_m)$. However, based on these values, he cannot learn anything about the private information of each user.

On the user side, besides the sensory data $\{x_m^{k'}\}_{m=1}^{M}$ and weight $w_{k'}$, each user $u_{k'}$ also knows the plaintexts of $\{x_m\}_{m=1}^{M}$ and $\sum_{k=1}^{K}\sum_{m=1}^{M} d(x_m^k, x_m)$, based on which the value $\sum_{k=1}^{K}\sum_{m=1}^{M} d(x_m^k, x_m) - \sum_{m=1}^{M} d(x_m^{k'}, x_m)$ can be calculated. However, since the number of all users satisfies $K \geq 3$, user $u_{k'}$ cannot infer the private information of any other individual users. For the data requester, he knows nothing about users' private information except the final aggregated results $\{x_m\}_{m=1}^{M}$.

In summary, the sensory data and weight information of each user will not be disclosed to other parties under the L-PPTD framework. □

3.4.3 Efficiency Analysis

Next, we will discuss the computational complexity and communication overhead of L-PPTD.

Computational Complexity. On the user side, all the computations are conducted on the plaintexts, and thus will introduce less overhead compared with the ciphertext-based calculations. In the initialization phase, each user only needs to generate some random numbers and then perturb her sensory data. The computational complexity is $O(M)$ for each user in this phase. In the iteration phase, each user u_k calculates the weight w_k based on $\sum_{m=1}^{M} d(x_m^k, x_m)$, the perturbed weighted data $\{w_k x_m^k - \beta_m^k\}_{m=1}^{M}$ and the perturbed weight $w_k - \gamma_k$. The computational costs of these calculations are $O(M)$, $O(M)$ and $O(1)$ respectively in each iteration.

On the cloud side, we mainly consider the overhead introduced by the ciphertext-based calculations, which dominate the overall computational cost when the key size is fixed. In each iteration, cloud S_A has to conduct $O(KM)$ encryptions in order to encrypt the values $\{(\tilde{x}_m^k - x_m)^2\}_{k,m=1}^{K,M}$, and $O(KM)$ ciphertext multiplications and exponentiations to calculate C_{conti} or C_{cate}. For cloud S_B, he needs to take $O(KM)$ encryptions to encrypt the random numbers $\{\alpha_m^k\}_{k,m=1}^{K,M}$ in the initialization phase and conduct decryption once in each iteration.

Communication Overhead. Since the proposed framework is used in crowd sensing systems, in which the mobile devices usually have limited energy resources, we hope that each user communicates with the clouds as little as possible. Thus, here we mainly analyze the amount of communication between the parties in L-PPTD. In addition to the analysis provided here, we also conduct real-world experiments to measure the communication overhead in Section 3.6.

In L-PPTD, each user needs to upload the perturbed data to cloud S_A and the random numbers to cloud S_B, both of which are conducted only once during the whole truth discovery procedure. In every iteration, each user first receives the summation $\sum_{k=1}^{K}\sum_{m=1}^{M} d(x_m^k, x_m)$ and the estimated object truths from cloud S_B, then uploads the perturbed weight and perturbed weighted data to cloud S_A. So the total number of communication times between each user and the two cloud platforms is $2(t+1)$, where t is the number of iterations. For the cloud platforms, cloud S_B needs to send the ciphertexts and summations of the random numbers to cloud S_A in the initialization phase. In each iteration, cloud S_A sends the ciphertext C_{conti} or C_{cate} together with the estimated truths to cloud S_B. So the total number of communication times between cloud S_A and cloud S_B is $t+1$.

3.4.4 Generalization

Although the logarithmic weight function is adopted in this chapter, L-PPTD can also incorporate other types of weight function, such as the reciprocal function $w_k = {d_k}^{-p}$ and the affine function $w_k = 1 - pd_k$, where $d_k = \sum_{m=1}^{M} d(x_m^k, x_m)$ and p is a parameter chosen based on the specific application scenarios. In L-PPTD, besides the summation $\sum_{k=1}^{K} d_k$ received from S_B, each user u_k can calculate the value d_k by herself. So as long as the weight function f (presented in Section 3.3.1) does not involve other information about all users except the summation $\sum_{k=1}^{K} d_k$, it can be calculated on the user side.

3.5 L^2-PPTD Framework

Although little overhead is introduced on the user side, each user in L-PPTD framework is still involved in the calculation of her own weight and weighted data. To make the proposed framework more efficient in the scenarios where only the sensory data of each user need to be protected, we propose an even more lightweight framework, called L^2-PPTD, in which the users need not to be involved in the iterative procedure. Similar

to L-PPTD, L^2-PPTD also contains two phases (i.e., *initialization phase* and *iteration phase*) as shown in Figure 3.2.

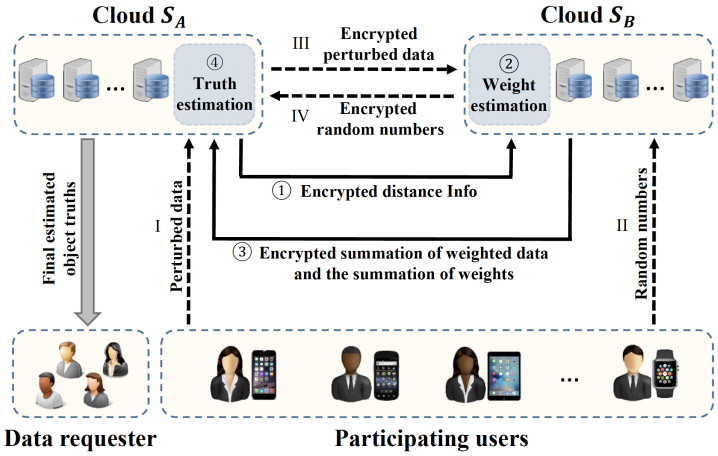


Figure 3.2: The workflow of L^2-PPTD

3.5.1 The Detailed Procedure of L^2-PPTD

In the L^2-PPTD framework, each user only needs to take part in the initialization phase. Both the weight estimation and truth estimation are completed on the cloud side.

Initialization Phase

This phase is also conducted only once during the whole truth discovery procedure. Different from L-PPTD, both of the two cloud platforms S_A and S_B need to generate their own public keys and private keys. We use (pk_A, sk_A) and (pk_B, sk_B) to denote the key pairs of S_A and S_B respectively.

Step I**:** Each user u_k generates a random number α_m^k for each object o_m (for categorical data, α_m^k is a vector in which each element is a random). Then she perturbs the sensory data x_m^k as $\tilde{x}_m^k = x_m^k - \alpha_m^k$, and uploads all the perturbed data $\{\tilde{x}_m^k\}_{m=1}^M$ to cloud S_A.

Step II**:** Each user u_k uploads the random numbers $\{\alpha_m^k\}_{m=1}^M$ to cloud S_B.

Step III**:** Cloud S_A first encrypts each perturbed sensory data $\tilde{x}_m^k$ with his own public key pk_A, then sends all the ciphertexts $\{E_A[\tilde{x}_m^k]\}_{k,m=1}^{K,M}$ (E_A is the encryption function

based on pk_A) to cloud S_B. For categorical data, $E_A[\tilde{x}_m^k]$ is a ciphertext vector in which each element is the ciphertext of the corresponding element in vector $\tilde{x}_m^k$.

Step IV**:** Cloud S_B encrypts each random number α_m^k with his own public key pk_B, and sends all the ciphertexts $\{E_B[\alpha_m^k]\}_{k,m=1}^{K,M}$ (E_B is the encryption function based on pk_B) to cloud S_A.

Iteration Phase

This phase also starts with the random initialization of the object truths on cloud S_A.

Step ①: For continuous data, cloud S_A calculates the ciphertext C_{conti}^k for each user u_k as

$$\begin{aligned} C_{conti}^k =& E_B[\sum_{m=1}^{M}(x_m^k - x_m)^2 - \sum_{m=1}^{M}(\alpha_m^k)^2] \\ =& \prod_{m=1}^{M}\{E_B[(\tilde{x}_m^k - x_m)^2] \cdot E_B[\alpha_m^k]^{2(\tilde{x}_m^k - x_m)}\}. \end{aligned} \tag{3.10}$$

For categorical data, S_A calculates the ciphertext C_{cate}^k for each user u_k as

$$\begin{aligned} C_{cate}^k =& E_B[\sum_{m=1}^{M}\sum_{i=1}^{l}(x_{mi}^k - x_{mi})^2 - \sum_{m=1}^{M}\sum_{i=1}^{l}(\alpha_{mi}^k)^2] \\ =& \prod_{m=1}^{M}\prod_{i=1}^{l}\{E_B[(\tilde{x}_{mi}^k - x_{mi})^2] \cdot E_B[\alpha_{mi}^k]^{2(\tilde{x}_{mi}^k - x_{mi})}\}. \end{aligned} \tag{3.11}$$

Then, all the ciphertexts $\{C_{conti}^k\}_{k=1}^K$ or $\{C_{cate}^k\}_{k=1}^K$ are sent to cloud S_B (here the estimated object truths are not sent).

Step ②: After receiving the ciphertexts from cloud S_A, cloud S_B decrypts them with his private key sk_B and calculates the summation of distances (i.e., $\sum_{m=1}^{M} d(x_m^k, x_m)$) for each user u_k by adding the value $\sum_{m=1}^{M}(\alpha_m^k)^2$ or $\sum_{m=1}^{M}\sum_{i=1}^{l}(\alpha_{mi}^k)^2$ to the decrypted data. Then, cloud S_B estimates the weight of u_k according to Eq. (3.1).

Step ③: When the sensory data is continuous, based on the estimated weights, cloud S_B first calculates the value $\sum_{k=1}^{K} w_k \alpha_m^k$ for each object o_m and encrypts it with S_A's

public key pk_A as $E_A[\sum_{k=1}^{K} w_k \alpha_m^k]$. Then, cloud S_B calculates the encrypted summation of weighted data for each object o_m as

$$\begin{aligned} E_A[\sum_{k=1}^{K} w_k x_m^k] =& E_A[\sum_{k=1}^{K} w_k (x_m^k - \alpha_m^k) + \sum_{k=1}^{K} w_k \alpha_m^k] \\ =& E_A[\sum_{k=1}^{K} w_k \tilde{x}_m^k] \cdot E_A[\sum_{k=1}^{K} w_k \alpha_m^k] \\ =& \prod_{k=1}^{K} \{E_A[\tilde{x}_m^k]^{w_k}\} \cdot E_A[\sum_{k=1}^{K} w_k \alpha_m^k]. \end{aligned} \tag{3.12}$$

When the sensory data is categorical, $E_A[\sum_{k=1}^{K} w_k x_m^k]$ is a ciphertext vector in which each element is calculated similarly with Eq. (3.12). Then, the ciphertexts (ciphertext vectors for categorical data) of all objects together with the summation of all users' weights (i.e., $\sum_{k=1}^{K} w_k$) are sent to cloud S_A.

Step ④: After receiving the data from cloud S_B, cloud S_A decrypts the ciphertexts and gets the summation $\sum_{k=1}^{K} w_k x_m^k$ for each object o_m. Then, the truth of each object is estimated according to Eq. (3.2).

Step ① $\sim$ ④ in this phase will also be iteratively conducted until the convergence criterion is satisfied. The final estimated truths are then sent to the data requester. In this framework, although the reliability information (i.e., weight) of each user is known by cloud S_B, much less overhead will be introduced on the user side since the users only take part in the initialization phase.

3.5.2 Security Analysis

Theorem 3. *Suppose there are at least two users providing different sensory data for each object. If the parties (including the two cloud platforms) are semi-honest and there is no collusion among them, the sensory data of each user will not be disclosed to any other party under L^2-PPTD framework.*

Proof. In L^2-PPTD, since each user does not receive any information about other parties, we just need to prove the sensory data of each user would not be disclosed to the cloud platforms and the data requester.

For cloud S_A, the values he knows include the ciphertexts $\{E_B[\alpha_m^k]\}_{k,m=1}^{K,M}$, $\{C_{conti}^k\}_{k=1}^K$ (or $\{C_{cate}^k\}_{k=1}^K$) and the plaintexts $\{x_m^k - \alpha_m^k\}_{k,m=1}^{K,M}$, $\{\sum_{k=1}^K w_k x_m^k\}_{m=1}^M$, $\sum_{k=1}^K w_k$, $\{x_m\}_{m=1}^M$. Without the private key sk_B, above ciphertexts cannot be decrypted by S_A. Since the weight of each user estimated on cloud S_B is not sent to cloud S_A, S_A cannot infer the individual values w_k, $w_k x_m^k$ just based on the summations $\{\sum_{k=1}^K w_k x_m^k\}_{m=1}^M$ and $\sum_{k=1}^K w_k$. Additionally, as the two cloud platforms do not collude with each other, the sensory data of each user will not be inferred by cloud S_A from the values $\{x_m^k - \alpha_m^k\}_{k,m=1}^{K,M}$.

For cloud S_B, he knows the ciphertexts $\{E_A[x_m^k - \alpha_m^k]\}_{k,m=1}^{K,M}$, $\{E_A[\sum_{k=1}^K w_k x_m^k]\}_{m=1}^M$ and the plaintexts $\{\alpha_m^k\}_{k,m=1}^{K,M}$, $\{\sum_{m=1}^M d(x_m^k, x_m)\}_{k=1}^K$, $\{w_k\}_{k=1}^K$. Since the object truths $\{x_m\}_{m=1}^M$ estimated on cloud S_A are not sent to cloud S_B, S_B can not learn the individual sensory data x_m^k from the values $\{\sum_{m=1}^M d(x_m^k, x_m)\}_{k=1}^K$. Additionally, S_B cannot decrypt the ciphertexts without S_A's private key. So each user's sensory data will not be known to cloud S_B. Additionally, similar to L-PPTD, the data requester can only know the final aggregated results $\{x_m\}_{m=1}^M$ in L^2-PPTD.

In summary, the sensory data of each user will not be disclosed to other parties under the L^2-PPTD framework. □

3.5.3 Efficiency Analysis

Computational Complexity. The participating users in L^2-PPTD are only involved in the initialization phase and the computational cost (based on plaintexts) is $O(M)$ for each user. For cloud S_A and cloud S_B, both of them are involved in the initialization phase and iteration phase. In the initialization phase, they both have to take $O(KM)$ encryptions. In each iteration, cloud S_A has to conduct $O(KM)$ encryptions, $O(M)$

decryptions, $O(KM)$ ciphertext multiplications and exponentiations. On the other hand, Cloud S_B needs to conduct $O(K)$ decryptions, $O(M)$ encryptions, $O(KM)$ ciphertext multiplications and $O(KM)$ ciphertext exponentiations to calculate the weights and ciphertexts $\{E_A[\sum_{k=1}^{K} w_k x_m^k]\}_{m=1}^{M}$.

Communication Overhead. In L^2-PPTD, since the participating users are only involved in the initialization phase, the number of communication times between each user and the cloud platforms is only 2 during the whole truth discovery procedure. As for cloud S_A and S_B, both of them need to send data once to each other in the initialization phase and each iteration, so the communication overhead between the two clouds is $2(t+1)$, where t is the number of iterations.

3.5.4 Generalization

Since the weight of each user in L^2-PPTD is estimated based on plaintexts on cloud S_B, which has all the values $\{\sum_{m=1}^{M} d(x_m^k, x_m)\}_{k=1}^{K}$ needed to estimate the users' weights, any weight function f can be used in this framework.

3.6 Performance Evaluation

In this section, we evaluate the performance of the proposed framework on real world crowdsensing systems. Both continuous and categorical sensory data are considered here. The cloud platforms in this experiment are emulated by two Intel(R) Core(TM) 3.4GHz computers running Ubuntu 14.04, with 16GB RAM. We use Nexus 5 Android phones as the mobile sensing devices. The frameworks are implemented with the Paillier encryption tool[1] and the key size is set as 512 bits. Additionally, we use the rounding parameter $R = 10^7$ to round the fractional data. The baseline methods are the original truth discovery method CRH [61, 65] and the privacy-preserving truth discovery framework PPTD [73, 74].

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

3.6.1 Experiment on Crowdsourced Indoor Floorplan Construction System

We first evaluate the performance of the proposed framework on one real-world crowdsensing application, i.e., crowdsourced indoor floorplan construction [11, 33]. In such crowdsensing systems, although the indoor floorplan can be automatically constructed from the sensory data (e.g., compass, accelerometer, gyroscope) collected by smartphone users, the sensor readings may disclose a user's private personal information. For the sake of illustration, here we only focus on estimating the distance (continuous data) between two particular location points in a hallway. 10 smartphone users are employed as the participating users and 20 hallway segments in a building are selected as the objects. We develop an Android App that can estimate the walking distances of a smartphone user through multiplying the user's step size by step count inferred using the in-phone accelerometer. The ground truths are obtained by measuring these segments manually.

Accuracy. We first compare the accuracy of the final estimated object truths between the proposed method and the baseline approach. In this experiment, the root of mean squared error (RMSE) is used to measure the deviation between the estimated results and the ground truths. Here the number of objects is fixed as 20 and we vary the number of users from 3 to 10. The results are shown in Figure 3.3. As seen, our proposed frameworks have almost the same estimation accuracy as CRH regardless of the number of users.

Convergence. In order to evaluate the convergence of the proposed algorithms, we measure the distance between the estimated object truths in two consecutive iterations using a *convergence value* defined as $||\mathbf{x}^t - \mathbf{x}^{t-1}||_2$, where $\mathbf{x}^t$ $(t \geq 1)$ is the vector of estimated truths in iteration t (the values in $\mathbf{x}^0$ are randomly initialized). The convergence values of L-PPTD are shown in Figure 3.4. Here we calculate the convergence values with 5 different initial truth vectors (i.e., $\mathbf{x}^0$). As we can see, the L-PPTD algorithm converges quickly in just a few iterations. L^2-PPTD has similar convergence pattern.

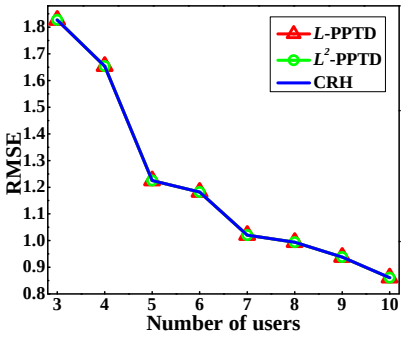


Figure 3.3: Accuracy for the indoor floorplan construction system

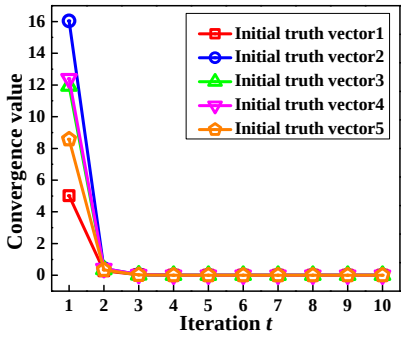


Figure 3.4: Convergence for the indoor floorplan construction system

Computational Cost. Here we evaluate the running time of the proposed frameworks on both the user and the cloud sides. The results are compared with that of the baseline method PPTD.

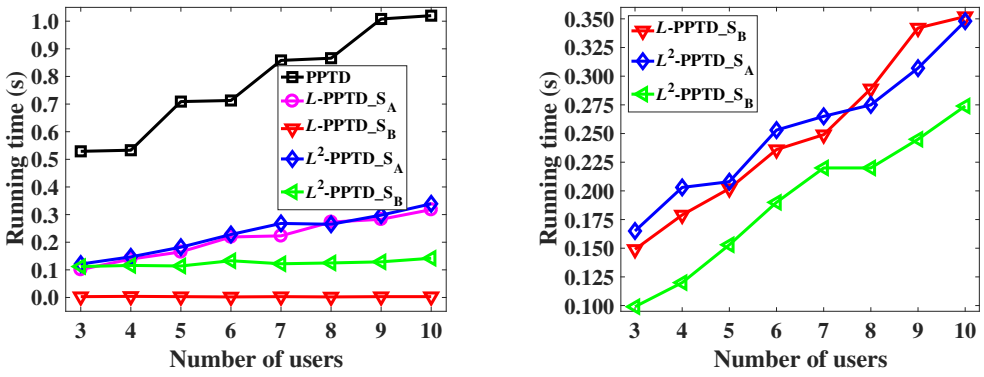
Table 3.1 shows the running time on one user's smartphone while the number of objects is varying from 4 to 20. Since the users are not involved in the iteration phase of L^2-PPTD, there is no result for that phase in Table 3.1. From this table, we can see the running time of the proposed frameworks is much less than that of PPTD. The reason is that all the computations on the user side are based on plaintexts in our frameworks while the users in PPTD need to do some encryptions. Additionally, the results in this table also show that L^2-PPTD is more lightweight than L-PPTD.

Table 3.1: Running time on smartphone for the indoor floorplan construction system

Number of objects		4	8	12	16	20
PPTD/iteration ($\times 10^{-2}s$)		1.3	2.0	2.8	3.4	5.1
L-PPTD ($\times 10^{-6}s$)	Initialization phase	3.0	4.6	6.0	6.7	9.3
	Each iteration	0.9	1.1	1.5	2.1	2.6
L^2-PPTD ($\times 10^{-6}s$)		2.6	3.0	3.8	4.1	4.9

Figure 3.5 shows the running time on the cloud platforms with the user number varying from 3 to 10. The results in one iteration and the initialization phase are provided in Figure. 3.5a and Figure 3.5b, respectively. Since cloud S_A in L-PPTD does not have to do any computation in the initialization phase, there is no result for S_A in Figure 3.5b.

From Figure 3.5a, we can see the running time of our frameworks on both cloud S_A and S_B are less than that of PPTD. This is mainly because of the threshold-based decryption scheme used in PPTD. Although the initialization phase is involved in our frameworks, it does not introduce much computational cost, which can be seen from Figure 3.5b.



(a) The iteration phase (b) The initialization phase

Figure 3.5: Running time on cloud platforms for the indoor floorplan construction system

Communication Overhead. Here we evaluate the data size each user needs to send or receive in the proposed frameworks. In Table 3.2, we provide a comparison between the proposed frameworks and PPTD. As can be seen, the data needed to be transmitted in each iteration of L-PPTD is much less than that of PPTD. The reason is that in the proposed frameworks, users do not need to send or receive any ciphertext, which is usually much larger than plaintext. Since each user is not involved in the iteration phase of L^2-PPTD and the initialization phases of the two frameworks are only needed to be conducted once, the total communication overhead of our scheme are also much less than that of PPTD. Additionally, the results further verify the conclusion that L^2-PPTD is more lightweight than L-PPTD since there is no communication overhead on the smartphone in each iteration of L^2-PPTD.

Table 3.2: Communication overhead on smartphone for the indoor floor plan construction system (KB)

Number of objects		4	8	12	16	20
PPTD/iteration		1.83	3.15	4.37	5.58	6.90
L-PPTD	Initialization phase	0.05	0.09	0.14	0.21	0.25
	Each iteration	0.10	0.18	0.26	0.33	0.41
L^2-PPTD		0.04	0.09	0.14	0.19	0.24

3.6.2 Experiment on Crowd Wisdom System

In this experiment, we evaluate the proposed frameworks on crowd wisdom system in which the sensory data are categorical. Each smartphone user in this system receives questions from the cloud to which she uploads her answers. The cloud infers the true answer for each question through aggregating the answers from smartphone users. However, some users may concern that their personal information may be inferred from their answers. Here we implement the proposed frameworks on this system. 100 smartphone users are employed as the participating users and 54 questions (each has 4 candidate answers) are used as the objects.

Accuracy and Convergence. In this part, we measure the accuracy of the proposed frameworks with *Error Rate*, which is defined as the percentage of mismatched values between the estimated results and the ground truths. Since some objects (i.e., questions) are not answered by all the users in this experiment, we vary the average number of users that answer each question and then calculate the Error Rate. The results of the two frameworks and CRH are shown in Figure 3.6. We can see that the Error Rates of the proposed frameworks are the same with that of CRH in all scenarios, which verifies the accuracy of our proposed frameworks for categorical data. Additionally, the convergence value defined before is also adopted here to measure the convergence. We find both our frameworks and CRH converge within two iterations.

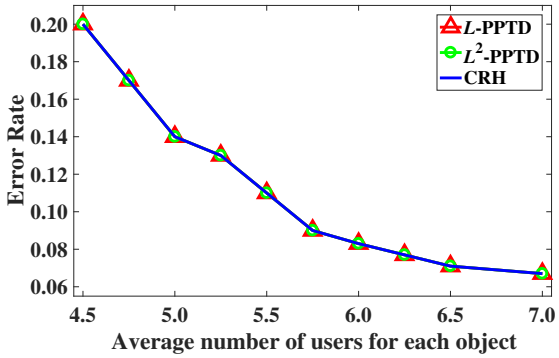


Figure 3.6: Accuracy for the wisdom system

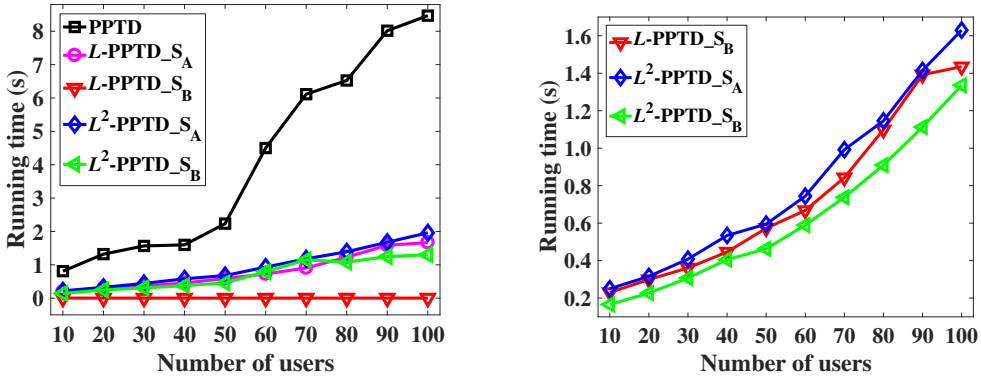
Computational Cost. Here we also evaluate the running time of the proposed frameworks on both smartphone and cloud platforms. In this experiment, we record the running time on smartphone while the number of objects is varying from 2 to 14, which is the maximum number of questions answered by a single user. The results are shown in Table 3.3, from which we can see the computational cost of our frameworks on smartphone is much less than that of PPTD.

Table 3.3: Running time on smartphone for the crowd wisdom system

Number of objects		2	5	8	11	14
PPTD/iteration ($\times 10^{-2}s$)		1.8	3.9	5.8	8.0	12.7
L-PPTD ($\times 10^{-6}s$)	Initialization phase	3.5	10.2	12.6	17.6	23.3
	Each iteration	1.3	2.9	7.3	6.2	8.2
L^2-PPTD ($\times 10^{-6}s$)		1.8	4.6	7.1	9.4	11.6

For the cloud platforms, Figure 3.7a and Figure 3.7b show the running time in each iteration and the initialization phase respectively. As seen, the running time of our frameworks on both cloud S_A and S_B is less than that of PPTD.

Communication Overhead. The size of data needed to be transmitted on each smartphone is shown in Table 3.4, from which we can see the communication overhead on smartphone in each iteration of L-PPTD is much less than that of PPTD. Additionally, the data transmitted by each user in L^2-PPTD is less than that of both L-PPTD and PPTD.



(a) The iteration phase

(b) The initialization phase

Figure 3.7: Running time on cloud platforms for the crowd wisdom system

Table 3.4: Communication overhead on smartphone for the crowd wisdom system (KB)

Number of objects		2	5	8	11	14
PPTD/iteration		3.1	6.9	10.6	14.4	18.1
L-PPTD	Initialization phase	0.09	0.24	0.38	0.52	0.66
	Each iteration	0.11	0.21	0.32	0.43	0.51
L^2-PPTD		0.07	0.18	0.28	0.39	0.49

3.6.3 Experiment on Simulated Crowd Sensing System

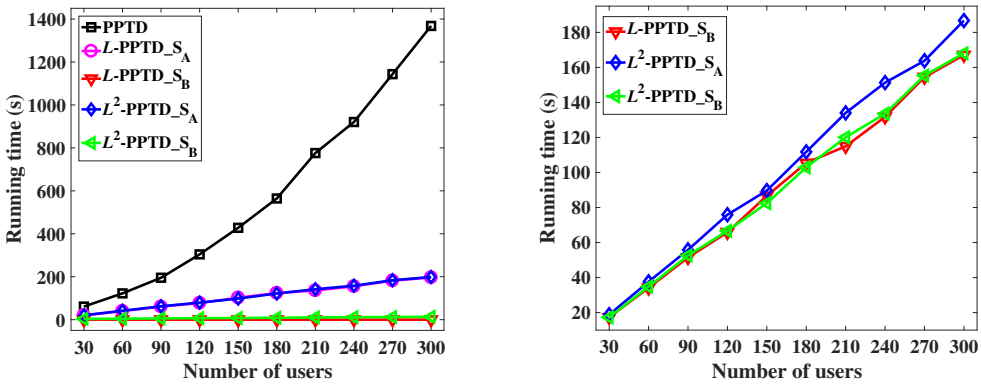
In order to evaluate the scalability and efficiency of the proposed frameworks, we conduct further experiments on a simulated crowd sensing system, in which there are 300 participating users and 1000 objects. We generate the sensory data of users through adding Gaussian noise with different intensities to the ground truths. Table 3.5 shows the running time of the frameworks on smartphone with varying object number ranging from 200 to 1000. From the table, we can see our proposed frameworks can keep high efficiency even when the number of objects is very large. Especially for L^2-PPTD, when the number of objects is 1000, the running time of L^2-PPTD on the smartphone is only $2.15 \times 10^{-4}s$ during the whole truth discovery procedure.

Next, we fix the number of objects as 500 and change the number of users from 30 to 300 in order to evaluate the computational cost of the cloud platforms. The results

Table 3.5: Running time on smartphone for the simulated crowd sensing system

Number of objects		200	400	600	800	1000
PPTD/iteration (s)		0.43	0.83	1.47	2.29	2.88
L-PPTD ($\times 10^{-4}s$)	Initialization phase	0.78	1.62	2.40	3.18	3.95
	Each iteration	0.24	0.51	0.64	0.86	1.14
L^2-PPTD ($\times 10^{-4}s$)		0.43	0.91	1.32	1.69	2.15

shown in Figure 3.8 further verify that the two frameworks in this chapter are more efficient on the cloud platforms than PPTD.



(a) The iteration phase (b) The initialization phase

Figure 3.8: Running time on cloud platforms for the simulated crowd sensing system

Table 3.6 reports the communication overhead on the smartphone while the number of objects observed by each user is changing from 400 to 1000. When the number of objects is 1000, the communication overhead in each iteration of L-PPTD is only 19.6KB while that in PPTD is 314.2KB. Additionally, the communication overhead of L^2-PPTD is only 13.5KB during the whole truth discovery procedure when the number of objects is 1000, which means L^2-PPTD is a more lightweight scheme for participating users.

Table 3.6: Communication overhead on smartphone for the simulated crowd sensing system (KB)

Number of objects		400	600	800	1000
PPTD/iteration		126.1	188.8	251.6	314.2
L-PPTD	Initialization phase	6.4	9.9	13.1	16.4
	Each iteration	7.7	11.6	15.5	19.6
L^2-PPTD		5.1	8.1	10.8	13.5

3.7 Summary

In this chapter, we propose a lightweight privacy-preserving truth discovery (L-PPTD) framework, which is implemented by involving two non-colluding cloud platforms and adopting additively homomorphic cryptosystem. This framework can not only protect the sensory data and reliability information of each participating user but also substantially reduce the overhead on the user side. Additionally, to further reduce each user's overhead in the scenarios where only the sensory data need to be protected, a more lightweight truth discovery (L^2-PPTD) framework is also developed.

Chapter 4

Data Poisoning Attacks in Crowd Sensing Systems

4.1 Introduction

Although crowd sensing provides an effective way to obtain useful information from the physical world, the sensory data collected from participating users are not always trustworthy. Due to the openness of the crowd sensing systems, the malicious parties can easily conduct malicious attacks. One important form of attacks is called *data poisoning*, where an attacker tries to degrade the effectiveness of the crowd sensing systems through creating or recruiting a group of malicious users and letting them submit malicious data. In this chapter, we focus on two types of data poisoning attacks: the *availability attack* and the *target attack*. In the availability attack, the attacker tries to disturb the final results as much as possible through manipulating the malicious users' sensory data. In the target attack, the attacker aims to skew the final results to predetermined target values.

If the attacker can create or recruit many malicious users, the attack goal is relatively easy to achieve, especially if the malicious users outnumber the normal users. However, when the attacker has limited resources, which is more often in real life, the attack strat-

egy becomes very important. Suppose the observations from users are categorical (e.g., the model and make of the car that hit the old lady and ran), then one intuitive attack strategy is to let all the malicious users report the answer with the second highest vote count (for the availability attack), or provide votes to the target answer (for the target attack). This strategy may be the optimal choice if the aggregation results are derived by majority voting. However, the story is much more complicated if the truth discovery approach is used for aggregation. Thanks to the ability of distinguishing users with different reliability degrees, the truth discovery approach can easily detect the malicious users, since they always disagree with the majority even when there is no chance to win, and therefore, assign low weights to the malicious users. Consequently, the impact of the malicious users will be greatly reduced, and the attack goal cannot be achieved.

Although truth discovery methods can tolerate the malicious behaviors of the users to some degree and effectively improve the aggregation results, it is not perfect in all cases. In this chapter, we propose an optimal attack framework that can take down a sensing system even with truth discovery empowered. Compared with the aforementioned naive attack strategy, the strategy derived from the proposed optimal attack framework makes the malicious users behave "smarter". They can successfully disguise themselves as normal users. If there is little hope to achieve the attack goal on some objects, they will tend to agree with the normal users on those objects to gain higher weights, and in turn, can exert stronger impact on other objects.

In our design, the optimal attack strategy is found by solving a bi-level optimization problem where the objective is to maximize the attack utility, in other words, the total number of the objects whose true values are skewed. As the attack goal is either achieved or not on one object, the attack utilities are discrete values, making it hard to solve the optimization problem. To handle this challenge, we use a continuous and differentiable sigmoid function to approximate the discrete attack utilities. Then we derive the optimal attack strategy by iteratively solving the upper-level and lower-level opti-

mization problems, where the former adopts the gradient ascent method and the latter is solved by block coordinate decent method.

4.2 Problem Setting

The crowd sensing system considered in this chapter consists of a *cloud server* and some *participating users*. The cloud server is a platform which holds some sensing tasks. In each sensing task, usually there are multiple objects needed to be observed. The participating users are the mobile device users who carry out the sensing tasks and provide their observations to the cloud server. After collecting the observations from all the users, the cloud server needs to estimate the true information (i.e., truth) of each object by conducting the truth discovery algorithm.

Suppose there is an *attacker* who aims to attack the crowd sensing system empowered with the truth discovery algorithm. As shown in Figure 4.1, the attacker cannot manipulate the observations of the *normal users* who carry out the sensing tasks without any malicious behavior, but he can create or recruit a group of *malicious users* and conduct attacks by carefully designing their observations. When conducting the availability attack, the attacker wants to maximize the error of the truth discovery algorithm running on the crowd sensing system, and eventually render the discovered truths useless. When conducting the target attack, the attacker aims to skew the final estimated object truths calculated by the cloud server to certain target values. In this chapter, we assume that the attacker has complete knowledge of the truth discovery algorithm and the sensory data from normal users. This assumption enables a robust assessment of the vulnerability of the crowd sensing system. Additionally, it is entirely possible that the attacker can get the normal users' data through eavesdropping the communications between normal users and the cloud server.

We formally define the problem addressed in this chapter as: Suppose the cloud server outsources a sensing task to a group of participating users. In this sensing task,

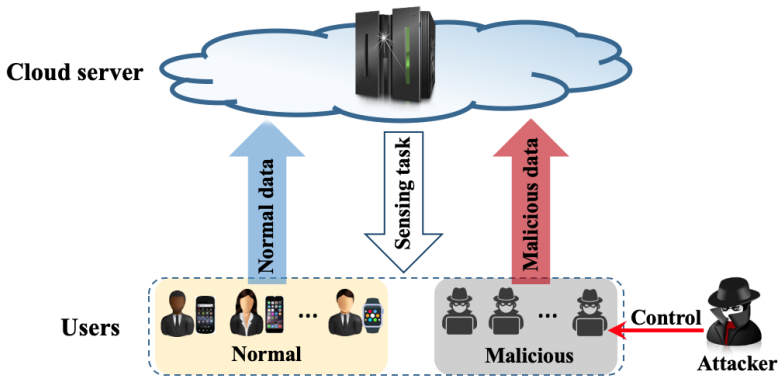


Figure 4.1: The crowd sensing system under attack

there are M objects which are observed by K normal users. We use $W = \{w_k\}_{k=1}^{K}$ to denote the weights (i.e., reliability degrees) of the normal users. The sensory data of normal users are denoted as $X = \{x_m^k\}_{m,k=1}^{M,K}$, in which x_m^k denotes the observation of the k-th normal user for the m-th object. The ground truth of each object is unknown by any party in the crowd sensing system, and the cloud server needs to calculate the estimated values $X^* = \{x_m^*\}_{m=1}^{M}$ of the ground truths for all the objects. Assume that there are K' malicious users that are created or recruited by the attacker in the crowd sensing system, and their weights are denoted as $\widetilde{W} = \{\widetilde{w}_{k'}\}_{k'=1}^{K'}$. We use $\widetilde{X} = \{\widetilde{x}_m^{k'}\}_{k',m=1}^{K',M}$ to denote the observations of all the malicious users and $\widetilde{x}_m^{k'}$ is the observation of the k'-th malicious user for the m-th object. Our goal in this chapter is to *find an optimal attack strategy (i.e., an optimal $\widetilde{X}$) such that the attack goal of the attacker can be achieved as much as possible*.

4.3 Preliminary

Although different truth discovery approaches have been designed for different scenarios, they share the same basic idea: A user ought to be assigned a high weight if his observations are close to the aggregated results, and the observations of a user ought to be counted more in the aggregation procedure if he has a high weight. In this chapter, we still consider the widely adopted truth discovery method CRH [61, 65], in which an

optimization framework is proposed to minimize the weighted deviation from the users' observations to the estimated truths:

$$\begin{aligned} \min_{X^*,W} \quad & f(X^*, W) = \sum_{k=1}^{K} w_k \sum_{m \in O_k} d(x_m^k, x_m^*) \\ \text{s.t.} \quad & \sum_{k=1}^{K} \exp(-w_k) = 1, \end{aligned} \tag{4.1}$$

where O_k is the set of objects observed by the k-th normal user (in this section, we assume that there are no malicious users in the crowd sensing system); $d(\cdot)$ is the loss function to measure the distance between users' observations and the estimated truths.

In this chapter, we consider the scenario where the sensory data are categorical. That is, for each object, the user would choose an answer from one of the C candidate answers. We use $x_m^k = (0, ..., \underset{q}{1}, ..., 0)^T$ to denote that the k-th user selects the q-th candidate answer for the m-th object. Then the distance between the observation vector x_m^k and the estimated truth vector x_m^* is defined as:

$$d(x_m^k, x_m^*) = (x_m^k - x_m^*)^T (x_m^k - x_m^*) = \sum_{c=1}^{C} (x_{mc}^k - x_{mc}^*)^2, \tag{4.2}$$

where x_{mc}^k and x_{mc}^* represent the c-th value in vector x_m^k and vector x_m^*, respectively.

CRH aims to learn the the estimated values X^* of the truths and user weights W together by optimizing the objective function in Eq. (4.1). In order to achieve the goal, block coordinate descent approach [5] is adopted and the following two steps are iteratively conducted until the convergence criterion is satisfied.

Step I: Truths Update. In this step, the users' weights W are fixed, and the estimated object truths X^* are updated according to:

$$x_m^* = \frac{\sum_{k \in U_m} w_k x_m^k}{\sum_{k \in U_m} w_k}, \tag{4.3}$$

where U_m is the set of normal users who observe the m-th object.

Clearly, the estimated object truth x_m^* is a vector of continuous values. It can be viewed as a probability vector in which each element represents the probability of the corresponding candidate answer being true. For example, suppose x_m^* equals to $(0.1, 0.7, 0.1, 0.1)$, then it implies that with 70% probability the second candidate answer is the true answer for the m-th object, and the probabilities of the others are all 10%. In this case, we assign the final estimated truth as the candidate answer with the largest value in vector x_m^*.

Step II: User Weights Update. In this step, the estimated object truths X^* are fixed, and the participating users' weights W are updated according to:

$$w_k = \log\left(\frac{\sum_{l=1}^{K}\sum_{m\in O_l} d(x_m^l, x_m^*)}{\sum_{m\in O_k} d(x_m^k, x_m^*)}\right). \tag{4.4}$$

where O_l is the set of objects observed by the l-th normal user.

4.4 Optimal Attack Framework

In this section, we present our optimal attack framework against the crowd sensing systems with truth discovery empowered. We first analyze the effect of malicious users on the truth discovery framework in Section 4.4.1. Then two types of data poisoning attacks (i.e., the availability attack and the target attack) are discussed in Section 4.4.2 and 4.4.3, respectively.

4.4.1 Truth Discovery with Malicious Users

We use $\widehat{X}^* = \{\widehat{x}_m^*\}_{m=1}^M$ to denote the estimated object truths after the data poisoning attack. After taking the malicious users into account, the truth discovery framework in

Eq.(4.1) becomes:

$$\begin{aligned}\min_{\widehat{X}^*,W,\widetilde{W}} f(\widehat{X}^*,W,\widetilde{W}) &= \sum_{k=1}^{K} w_k \sum_{m\in O_k} d(x_m^k,\widehat{x}_m^*) + \sum_{k'=1}^{K'} \widetilde{w}_{k'} \sum_{m\in \widetilde{O}_{k'}} d(\widetilde{x}_m^{k'},\widehat{x}_m^*) \\ \text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) &+ \sum_{k'=1}^{K'} \exp(-\widetilde{w}_{k'}) = 1,\end{aligned} \tag{4.5}$$

where $\widetilde{O}_{k'}$ is the set of the objects observed by the k'-th malicious user. Here we decompose the participating users into normal and malicious ones for the purpose of analyzing the effect of malicious users on the estimated object truths. However, note that from the cloud server's perspective, it is not aware of the attack and cannot differentiate the two types of users when conducting truth discovery. Based on the block coordinate descent method which is adopted in the original truth discovery framework, the optimal solution $\{\widehat{X}^*, W, \widetilde{W}\}$ can be calculated by iteratively conducting the following two steps until the convergence criterion is satisfied.

Step I: Truths Update. In this step, we first fix the weights of normal and malicious users (i.e., W and $\widetilde{W}$), then update the estimated object truths $\widehat{X}^*$ according to

$$\widehat{x}_m^* = \frac{\sum_{k\in U_m} w_k x_m^k + \sum_{k'\in \widetilde{U}_m} \widetilde{w}_{k'} \widetilde{x}_m^{k'}}{\sum_{k\in U_m} w_k + \sum_{k'\in \widetilde{U}_m} \widetilde{w}_{k'}}. \tag{4.6}$$

where $\widetilde{U}_m$ is the set of malicious users who observe the m-th object. As described in Section 4.3, $\widehat{x}_m^*$ is a vector in which each element represents the probability of the corresponding candidate answer being true after the attack. The c-th element in this vector is updated as

$$\widehat{x}_{mc}^* = \frac{\sum_{k\in U_m} w_k x_{mc}^k + \sum_{k'\in \widetilde{U}_m} \widetilde{w}_{k'} \widetilde{x}_{mc}^{k'}}{\sum_{k\in U_m} w_k + \sum_{k'\in \widetilde{U}_m} \widetilde{w}_{k'}}. \tag{4.7}$$

where $\widetilde{x}_{mc}^{k'}$ is the c-th value in vector $\widetilde{x}_m^{k'}$.

Step II: User Weights Update. In this step, the estimated object truths $\widehat{X}^* = \{\widehat{x}_m^*\}_{m=1}^M$ are fixed. Then we update the weights of normal and malicious users (i.e., W and $\widetilde{W}$) as

$$w_k = \log(\frac{\sum_{l=1}^{K}\sum_{m\in O_l} d(x_m^l, \widehat{x}_m^*) + \sum_{l'=1}^{K'}\sum_{m\in \widetilde{O}_{l'}} d(\widetilde{x}_m^{l'}, \widehat{x}_m^*)}{\sum_{m\in O_k} d(x_m^k, \widehat{x}_m^*)}), \tag{4.8}$$

$$\widetilde{w}_{k'} = \log(\frac{\sum_{l=1}^{K}\sum_{m\in O_l} d(x_m^l, \widehat{x}_m^*) + \sum_{l'=1}^{K'}\sum_{m\in \widetilde{O}_{l'}} d(\widetilde{x}_m^{l'}, \widehat{x}_m^*)}{\sum_{m\in \widetilde{O}_{k'}} d(\widetilde{x}_m^{k'}, \widehat{x}_m^*)}). \tag{4.9}$$

where $\widetilde{O}_{l'}$ is the set of objects observed by the l'-th malicious user.

From the above equations, we can see the estimated object truths $\widehat{X}^* = \{\widehat{x}_m^*\}_{m=1}^M$ are only dependent on the observations of malicious users (i.e., $\widetilde{X} = \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'}$) once the data of normal users are given. In this way, the attacker can attack truth discovery algorithm by elaborately designing the observations of malicious users.

4.4.2 Availability Attack

In the availability attack, the attacker aims to maximize the error of the crowd sensing systems where the observations from multiple users are aggregated by the truth discovery algorithm, and eventually render them useless. In other words, the attacker tries to make the deviation between the outputs of truth discovery before and after the availability attack as much as possible. More specifically, if the final truth discovery result on an object is changed after the attack, it means that the attack on this object succeeds. Otherwise, the attack on this object fails. In this section we discuss how to find the optimal attack strategy from the perspective of the attacker so that the attack can succeed on as many objects as possible.

Given the number of malicious users created or recruited by the attacker and the objects they can observe, the attacker needs to find the optimal assignments for each

malicious user's observations to conduct the availability attack. Let's denote the final estimated answers for the m-th object before and after the attack as x_m^{*f} and $\widehat{x}_m^{*f}$ respectively. We can formulate the goal of the availability attack into an optimization problem as follows:

$$\begin{aligned} \max_{\widetilde{X}} \sum_{m=1}^{M} \mathbb{1}(\widehat{x}_m^{*f} \neq x_m^{*f}) \\ \text{s.t.} \quad \{\widehat{X}^{*f}, W, \widetilde{W}\} = \underset{\widehat{X}^{*f}, W, \widetilde{W}}{\operatorname{argmin}} \; f(\widehat{X}^{*f}, W, \widetilde{W}) \\ \text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) + \sum_{k'=1}^{K'} \exp(-\widetilde{w}_{k'}) = 1, \end{aligned} \tag{4.10}$$

where $\widehat{X}^{*f} = \{\widehat{x}_m^{*f}\}_{m=1}^{M}$ are the final estimated answers after the attack and $\mathbb{1}(\cdot)$ is the indicator function. In this optimization problem, the truth discovery framework Eq.(4.5) becomes a constraint. This is a bi-level optimization problem [3]. The optimization over malicious observations $\widetilde{X}$ is the upper-level problem, and the optimization over $\{\widehat{X}^{*f}, W, \widetilde{W}\}$ given $\widetilde{X}$ is the lower-level problem. x_m^{*f} is the final aggregation result (calculated based on normal users' data) before the attack, and it is a constant once the normal users' sensory data are given. $\widehat{x}_m^{*f}$ depends on the attack strategy (i.e., $\widetilde{X}$) and can be different as the attack strategy varies.

For each object, the malicious users need to pick one candidate answer. An intuitive attack strategy is to choose the answer with the second highest probability to be true. The reason is simple, this answer has the most chance to win over the answer with the highest probability. However, this attack strategy may not be the optimal choice under truth discovery mechanism. Let's consider the following example: If the margin between the answers with the highest and the second highest probability is too large for an object (e.g., 100 votes V.S. 2 votes), it is impossible for the limited number of malicious users to change the aggregation result on this object. Moreover, since the malicious users always disagree with the majority, the truth discovery algorithm can detect them easily and assign them with low weights. Consequently, the impact of the

malicious users on other objects also decreases, and thus may fail on all objects. To address this challenge, we take the truth discovery framework as a constraint in our designed optimization problem (4.10). Then the weights of malicious users will be taken into account during the procedure of finding the optimal attack strategy. As a result, we may find a better attack strategy compared with the intuitive one. The optimal attack may sacrifice on some of the objects where there is little chance to succeed, and agree with the majority users on those objects. The benefit of doing so is that the truth discovery algorithm may consider them as normal users or even good users and increase their weights, and eventually increase their impact on other objects.

Since the answer with the second highest probability has the most chance to win over the answer with the highest probability, we only consider the change on these two answers. We reformulate problem (4.10) as:

$$\max_{\widetilde{X}} \sum_{m=1}^{M} \frac{1}{2}\{1 - \text{sgn}[(x^*_{mc_2} - x^*_{mc_1}) \cdot (\widehat{x}^*_{mc_2} - \widehat{x}^*_{mc_1})]\} \tag{4.11}$$

$$\text{s.t.} \quad \{\widehat{X}^*, W, \widetilde{W}\} = \underset{\widehat{X}^*, W, \widetilde{W}}{\text{argmin}} \ f(\widehat{X}^*, W, \widetilde{W})$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) + \sum_{k'=1}^{K'} \exp(-\widetilde{w}_{k'}) = 1$$

$$\{\widetilde{x}^{k'}_{mc}\}^{K',M,C}_{k',m,c=1} \in \{0, 1\} \tag{4.12}$$

where c_1 and c_2 indicate the answers with the highest and the second highest element in the probability vector (i.e., x^*_m) output by the truth discovery algorithm before the attack. Constraint (4.12) is used to limit each element $\widetilde{x}^{k'}_{mc}$ to 0 or 1. This optimization problem reflects the following idea: After the attack, if the answer with the second highest probability does not win over the answer with the highest probability, the attack fails on this object. Consequently, there is no gain in the objective value.

In the objective function (4.11), $(x^*_{mc_2} - x^*_{mc_1}) < 0$ when the normal users' data are given. Then we can know:

$$\text{sgn}[(x^*_{mc_2} - x^*_{mc_1}) \cdot (\widehat{x}^*_{mc_2} - \widehat{x}^*_{mc_1})] = \begin{cases} 1 & if\ \widehat{x}^*_{mc_2} < \widehat{x}^*_{mc_1} \\ 0 & if\ \widehat{x}^*_{mc_2} = \widehat{x}^*_{mc_1} \\ -1 & if\ \widehat{x}^*_{mc_2} > \widehat{x}^*_{mc_1}. \end{cases} \tag{4.13}$$

However, Eq. (4.13) is not continuous, and this makes it difficult to solve the above optimization problem. A potential way to address this challenge is to approximate the objective function (4.11) by a continuous and differentiable function. Considering that function $u_1(x) = \frac{1}{2}(1 - \text{sgn}\, x)$ can be well approximated by function $u_2(x) = 1 - \frac{1}{1+\exp(-\theta x)}$ when θ (i.e., the steepness of the curve) is set to an appropriate value, we approximate the objective function (4.11) by the following objective function:

$$\max_{\widetilde{X}} \sum_{m=1}^{M} \{1 - \frac{1}{1 + \exp[-\theta(x^*_{mc_2} - x^*_{mc_1})(\widehat{x}^*_{mc_2} - \widehat{x}^*_{mc_1})]}\}. \tag{4.14}$$

From the perspective of the attacker, $\sum_{m=1}^{M}\{1 - \frac{1}{1+\exp[-\theta(x^*_{mc_2} - x^*_{mc_1})(\widehat{x}^*_{mc_2} - \widehat{x}^*_{mc_1})]}\}$ in the objective function (4.14) can be treated as his utility, and he needs to find an appropriate attack strategy such that the utility can be maximized.

When solving the above optimization problem, we still have another challenge, i.e., the value of each element (i.e., $\widetilde{x}^{k'}_{mc}$) in $\widetilde{X}$ is categorical, which makes it difficult to solve the upper-level problem. Here we treat each observation of malicious users (i.e., $\widetilde{x}^{k'}_{m}$) as a probability vector and relax the value of $\widetilde{x}^{k'}_{mc}$ to the range $(0, 1)$. In this way, we can solve the optimization problem according to the gradient-based methods. Please note that the summation of all the elements in vector $\widetilde{x}^{k'}_{m}$ should be 1, and the candidate answer with the largest value in this vector will be submitted to the cloud server. Then the following

optimization problem needs to be solved in order to maximize the attacker's utility.

$$\max_{\widetilde{X}} \; g(\widetilde{X}) = \sum_{m=1}^{M}\{1 - \frac{1}{1+\exp[-\theta(x^*_{mc_2} - x^*_{mc_1})(\widehat{x}^*_{mc_2} - \widehat{x}^*_{mc_1})]}\} + \delta_1 \sum_{k'=1}^{K'} \sum_{m\in\widetilde{O}_{k'}} \sum_{c=1}^{C} \log \widetilde{x}^{k'}_{mc} + \delta_2 \sum_{k'=1}^{K'} \sum_{m\in\widetilde{O}_{k'}} \sum_{c=1}^{C} \log(1-\widetilde{x}^{k'}_{mc}) \tag{4.15}$$

$$\text{s.t.} \quad \{\widehat{X}^*, W, \widetilde{W}\} = \underset{\widehat{X}^*, W, \widetilde{W}}{\operatorname{argmin}} \; f(\widehat{X}^*, W, \widetilde{W}) \tag{4.16}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) + \sum_{k'=1}^{K'} \exp(-\widetilde{w}_{k'}) = 1$$

$$\sum_{c=1}^{C} \widetilde{x}^{k'}_{mc} = 1, \; k' = 1, ..., K' \text{ and } m = 1, ..., M. \tag{4.17}$$

The objective function $g(\widetilde{X})$ contains three terms: The first term is the utility of the attacker. The second and the third terms work as the barriers to limit the sensory data of malicious users to the range $(0, 1)$. Parameters δ_1 and δ_2 are used to adjust the trade-off between these three terms. Here we use the barriers instead of constraint $\{\widetilde{x}^{k'}_{mc}\}^{K',M,C}_{k',m,c=1} \in (0, 1)$ in the optimization problem to reduce the computation complexity. The optimal solution of the above problem is very close to that of the original optimization problem when parameters δ_1 and δ_2 are small, and θ is large. Constraint (4.17) can guarantee the summation of the elements in the probability vector (i.e., $\widetilde{x}^{k'}_{m}$) equals to 1.

Next, we discuss how to solve this optimization problem. Inspired by the dual ascent method [7], we first get the Lagrangian form of the upper-level problem:

$$L_1(\widetilde{X}, \Psi) = \sum_{m=1}^{M}\{1 - \frac{1}{1+\exp[-\theta(x^*_{mc_2} - x^*_{mc_1})(\widehat{x}^*_{mc_2} - \widehat{x}^*_{mc_1})]}\} + \delta_1 \sum_{k'=1}^{K'} \sum_{m\in\widetilde{O}_{k'}} \sum_{c=1}^{C} \log \widetilde{x}^{k'}_{mc} + \delta_2 \sum_{k'=1}^{K'} \sum_{m\in\widetilde{O}_{k'}} \sum_{c=1}^{C} \log(1-\widetilde{x}^{k'}_{mc}) + \sum_{k'=1}^{K'} \sum_{m\in\widetilde{O}_{k'}} \psi^{k'}_m (\sum_{c=1}^{C} \widetilde{x}^{k'}_{mc} - 1), \tag{4.18}$$

where $\Psi = \{\psi_m^{k'}\}_{m,k'=1}^{M,K'}$ are the Lagrangian multipliers. The solution we adopted here is a two-phase iterative procedure:

Phase I: In this phase, we first fix the Lagrange multipliers $\Psi = \{\psi_m^{k'}\}_{m,k'=1}^{M,K'}$, which are calculated in the previous iteration. Then we solve the following optimization problem:

$$\begin{aligned} \max_{\widetilde{X}} \quad & L_1(\widetilde{X}, \Psi) \\ \text{s.t.} \quad & \{\widehat{X}^*, W, \widetilde{W}\} = \underset{\widehat{X}^*, W, \widetilde{W}}{\operatorname{argmin}} \ f(\widehat{X}^*, W, \widetilde{W}) \\ & \text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) + \sum_{k'=1}^{K'} \exp(-\widetilde{w}_{k'}) = 1. \end{aligned} \tag{4.19}$$

The method used to solve optimization problem (4.19) is also a two-step iterative procedure. Here we call this procedure the *inner iterative procedure* in order to differentiate it from the two-phase iterative procedure mentioned above. The two steps of the inner iterative procedure are summarized as follows:

Step ①: We fix the malicious users' observations $\widetilde{X}$, which are calculated in the previous iteration of the inner iterative procedure. Then we solve the lower-level problem to get the optimal solution $\{\widehat{X}^*, W, \widetilde{W}\}$, which is the truth discovery problem discussed in Section 4.4.1.

Step ②: We adopt the gradient ascent method to solve the upper-level problem. More specifically, in iteration r of the inner iterative procedure, $\widetilde{x}_{mc_1}^{k'}$ and $\widetilde{x}_{mc_2}^{k'}$ in vector $\widetilde{x}_m^{k'}$ are updated as

$$\widetilde{x}_{mc}^{k'(r+1)} \leftarrow \widetilde{x}_{mc}^{k'(r)} + \alpha_r^1 \cdot \nabla_{\widetilde{x}_{mc}^{k'}} L_1(\widetilde{X}, \Psi), \quad c \in \{c_1, c_2\} \tag{4.20}$$

where α_r^1 is the step size in iteration r of the inner iterative procedure. For gradient $\nabla_{\widetilde{x}_{mc}^{k'}} L_1(\widetilde{X}, \Psi)$, it is calculated as

$$\begin{aligned}\nabla_{\widetilde{x}_{mc}^{k'}} L_1(\widetilde{X}, \Psi) = &\sum_{m'=1}^{M} \{\frac{\exp(\theta d_1 d_2)\theta d_1}{[1+\exp(\theta d_1 d_2)]^2} \cdot \frac{\partial \widehat{x}^*_{m'c_1}}{\partial \widetilde{x}_{mc}^{k'}}\} \\ &- \sum_{m'=1}^{M} \{\frac{\exp(\theta d_1 d_2)\theta d_1}{[1+\exp(\theta d_1 d_2)]^2} \cdot \frac{\partial \widehat{x}^*_{m'c_2}}{\partial \widetilde{x}_{mc}^{k'}}\} \\ &+ \frac{\delta_1}{\widetilde{x}_{mc}^{k'}} - \frac{\delta_2}{1-\widetilde{x}_{mc}^{k'}} + \psi_m^{k'},\end{aligned} \tag{4.21}$$

where $d_1 = x^*_{m'c_2} - x^*_{m'c_1}$ and $d_2 = \widehat{x}^*_{m'c_2} - \widehat{x}^*_{m'c_1}$. Here $\frac{\partial \widehat{x}^*_{m'c_1}}{\partial \widetilde{x}_{mc}^{k'}}$ and $\frac{\partial \widehat{x}^*_{m'c_2}}{\partial \widetilde{x}_{mc}^{k'}}$ are calculated based on Eq. (4.7):

$$\frac{\partial \widehat{x}^*_{m'c_1}}{\partial \widetilde{x}_{mc}^{k'}} = \begin{cases} \frac{\widetilde{w}_{k'}}{\sum_{k\in U_m} w_k + \sum_{k'\in \widetilde{U}_m} \widetilde{w}_{k'}} & m = m' \text{ and } c = c_1 \\ 0 & \text{others.} \end{cases} \tag{4.22}$$

$$\frac{\partial \widehat{x}^*_{m'c_2}}{\partial \widetilde{x}_{mc}^{k'}} = \begin{cases} \frac{\widetilde{w}_{k'}}{\sum_{k\in U_m} w_k + \sum_{k'\in \widetilde{U}_m} \widetilde{w}_{k'}} & m = m' \text{ and } c = c_2 \\ 0 & \text{others.} \end{cases} \tag{4.23}$$

The reason why we only update $\widetilde{x}_{mc_1}^{k'}$ and $\widetilde{x}_{mc_2}^{k'}$ is that only the two answers (i.e., c_1 and c_2) with the highest and second highest probability in vector x_m^* are considered when we assign the observations for the malicious users. The malicious users who observe the m-th object should select one of the answers (i.e., c_1 or c_2) as his observation for this object in order to achieve the attack goal.

Step ① and step ② in the inner iterative procedure will be conducted until the convergence criterion is satisfied. Here the convergence criterion is that all the gradients $\{\nabla_{\widetilde{x}_{mc}^{k'}} L_1(\widetilde{X}, \Psi)\}_{m,k'}^{M,K'}$ are less than a threshold.

Phase II: We adopt the gradient descent method to update the Lagrangian multipliers $\Psi = \{\psi_m^{k'}\}_{m,k'=1}^{M,K'}$ based on $\widetilde{X}$ calculated in phase I. More specifically, in iteration t, $\psi_m^{k'}$

is updated as

$$\psi_m^{k'(t+1)} \leftarrow \psi_m^{k'(t)} - \alpha_t^2 \cdot (\sum_{c=1}^{C} \widetilde{x}_{mc}^{k'} - 1) \tag{4.24}$$

where α_t^2 is the step size in iteration t.

The above two phases will be iteratively conducted until the Lagrangian multipliers $\{\psi_m^{k'}\}_{m,k'=1}^{M,K'}$ converge. We can get malicious users' observation vectors $\widetilde{X} = \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'}$. Then the k'-th malicious user selects the candidate answer with the largest value in vector $\widetilde{x}_m^{k'}$ as the final observation of the m-th object, and submit it to the cloud server. The procedure is summarized as Algorithm 4.

Algorithm 4: Optimizing $\widetilde{X}$ for the availability attack

Input: The number of objects: M; the number of normal users: K; the normal users' observations: X; the number of malicious users: K'; the objects observed by the malicious users: $\{\widetilde{O}_{k'}\}_{k'=1}^{K'}$.

Output: The optimal attack strategy $\widetilde{X}_{opt}$

```
Initialize the malicious users' observations X̃ and the Lagrange multipliers Ψ;
X̃_opt ← ∅;
while Ψ = {ψ_m^{k'}}_{m,k'=1}^{M,K'} does not converge do
    while the gradients do not satisfy the convergence criterion do
        Calculate the optimal solution {X̂*, W, W̃} based on Eq.(4.7), Eq.(4.8) and
          Eq.(4.9);
        Update X̃ based on Eq.(4.20);
    end
    Update Ψ based on Eq.(4.24);
end
for each x̃_m^{k'} ∈ X̃ do
    x̃_m^{k'(opt)} ← the candidate answer with the largest value in vector x̃_m^{k'};
    X̃_opt ← X̃_opt ∪ {x̃_m^{k'(opt)}};
end
return The optimal attack strategy X̃_opt;
```

4.4.3 Target Attack

In the target attack, the attacker tries to skew the estimated truths of some objects (called the *target objects*) to certain target answers through poisoning the sensory data. The target answers are usually predetermined by the attacker. When conducting the target attack, if the final truth discovery result on a target object is changed to the target answer after the attack, it means that the attack on this object succeeds. Otherwise, the attack on this object fails.

Given the limited capability of the attacker, in this section we discuss how to find the optimal attack strategy so that the attack can succeed on as many target objects as possible. Suppose that the attacker wants to attack $\overline{M}$ ($\overline{M} \leq M$) objects among all the objects observed by the normal users. The target answer and the final estimated answer for the $\overline{m}$-th target object after the attack are denoted as $\overline{x}_{\overline{m}}^{*f}$ and $\widehat{x}_{\overline{m}}^{*f}$ respectively. We can formulate the goal of the target attack into an optimization problem as follows:

$$\begin{aligned} \max_{\widetilde{X}} \sum_{\overline{m}=1}^{\overline{M}} \mathbb{1}(\widehat{x}_{\overline{m}}^{*f} = \overline{x}_{\overline{m}}^{*f}) \\ \text{s.t.} \quad \{\widehat{X}^{*f}, W, \widetilde{W}\} = \underset{\widehat{X}^{*f}, W, \widetilde{W}}{\operatorname{argmin}} \ f(\widehat{X}^{*f}, W, \widetilde{W}) \\ \text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) + \sum_{k'=1}^{K'} \exp(-\widetilde{w}_{k'}) = 1, \end{aligned} \tag{4.25}$$

where $\widehat{X}^{*f} = \{\widehat{x}_m^{*f}\}_{m=1}^{M}$ are the final estimated object answers after the attack and $\{\widehat{x}_{\overline{m}}^{*f}\}_{\overline{m}=1}^{\overline{M}} \in \widehat{X}^{*f}$.

When conducting the target attack, the malicious users need to pick one candidate answer for each target object. An intuitive attack strategy is to choose the target answer. However, this strategy may not be the optimal choice under truth discovery framework. The reason is similar to that in the availability attack. The weights of malicious users can be greatly decreased since they would disagree with the majority of the normal users. A better strategy may sacrifice on some of the target objects where it is unlikely to skew the

estimated truths, so that the weights of malicious users can be increased, and eventually their impact on other objects can be improved. Here we assume that each malicious user only observes the target objects. Since we only consider the change between the target answers and the answers with the highest probability values in the estimated truth vectors before the attack, we reformulate problem (4.25) as

$$\begin{aligned}
\max_{\widetilde{X}} \sum_{\overline{m}=1}^{\overline{M}} \frac{1}{2}\{1-\mathrm{sgn}[(x^*_{\overline{m}c_1}-x^*_{\overline{m}c_T})\cdot(\widehat{x}^*_{\overline{m}c_1}-\widehat{x}^*_{\overline{m}c_T})]\} \qquad (4.26)\\
\text{s.t.} \quad \{\widehat{X}^*, W, \widetilde{W}\} = \underset{\widehat{X}^*, W, \widetilde{W}}{\mathrm{argmin}} \; f(\widehat{X}^*, W, \widetilde{W})\\
\text{s.t.} \quad \sum_{k=1}^{K}\exp(-w_k)+\sum_{k'=1}^{K'}\exp(-\widetilde{w}_{k'})=1\\
\{\widetilde{x}^{k'}_{\overline{m}c}\}^{K',\overline{M},C}_{k',\overline{m},c=1} \in \{0,1\},
\end{aligned}$$

where $\widehat{X}^*$ is the set of probability vectors output by truth discovery algorithm. In the objective function (4.26), c_T and c_1 represent the target answer and the answer with the highest value in the probability vector calculated before the attack. This optimization problem reflects the idea that if the object truth does not switch from the answer with the highest probability value to the target answer after the attack, the attack fails on this target object.

Similar to the availability attack, in order to solve this optimization problem, we approximate the objective function (4.26) by:

$$\max_{\widetilde{X}} \sum_{\overline{m}=1}^{\overline{M}}\{1-\frac{1}{1+\exp[-\theta(x^*_{\overline{m}c_1}-x^*_{\overline{m}c_T})(\widehat{x}^*_{\overline{m}c_1}-\widehat{x}^*_{\overline{m}c_T})]}\}, \qquad (4.27)$$

where θ denotes the steepness of the curve. Then we can formulate the following optimization problem to achieve the attacker's goal.

$$\max_{\widetilde{X}} \; h(\widetilde{X}) = \sum_{\overline{m}=1}^{\overline{M}}\{1-\frac{1}{1+\exp[-\theta(x^*_{\overline{m}c_1}-x^*_{\overline{m}c_T})(\widehat{x}^*_{\overline{m}c_1}-\widehat{x}^*_{\overline{m}c_T})]}\}$$

$$+ \delta_1 \sum_{k'=1}^{K'} \sum_{\overline{m}=1}^{\overline{M}} \sum_{c=1}^{C} \log \widetilde{x}_{\overline{m}c}^{k'} + \delta_2 \sum_{k'=1}^{K'} \sum_{\overline{m}=1}^{\overline{M}} \sum_{c=1}^{C} \log(1 - \widetilde{x}_{\overline{m}c}^{k'}) \tag{4.28}$$

$$\text{s.t.} \quad \{\widehat{X}^*, W, \widetilde{W}\} = \underset{\widehat{X}^*, W, \widetilde{W}}{\operatorname{argmin}} \ f(\widehat{X}^*, W, \widetilde{W})$$

$$\text{s.t.} \quad \sum_{k=1}^{K} \exp(-w_k) + \sum_{k'=1}^{K'} \exp(-\widetilde{w}_{k'}) = 1$$

$$\sum_{c=1}^{C} \widetilde{x}_{\overline{m}c}^{k'} = 1, \text{ where } k' = 1, ..., K' \text{ and } \overline{m} = 1, ..., \overline{M}.$$

Similar to the optimization problem formulated in the availability attack, this problem is a bi-level optimization problem and the objective function $h(\widetilde{X})$ contains three terms: The first term represents the utility of the attacker. The second and the third terms are the barriers used to limit each element in the malicious users' observation vectors to the range $(0, 1)$. The solution for this optimization problem is also a two-phase iterative procedure which is similar to that for the availability attack.

4.5 Experiments on the Crowd Wisdom System

We build a crowd wisdom system to evaluate the performance of the proposed attack framework. In this system, the cloud server publishes some multi-choice trivia questions using the Android App we developed, and the users can view and submit their answers using the App. After receiving the answers from the users, the cloud server applies the truth discovery approach (i.e., the CRH framework) to infer the true answer for each question. The attack occurs after the data of the normal users are submitted to the cloud server but before the truth discovery procedure starts. In our experiment, 30 smartphone users are employed as the normal users and 19 questions are used as the objects. Each question has 4 candidate answers and the users can only choose one answer for each question. The participants are not required to answer all questions. Instead, they can choose any questions as they will.

4.5.1 Availability Attack

In the availability attack, the attacker tries to maximally disturb the truth discovery results. In this experiment, a fixed number of malicious users can be created, and each of them can observe a randomly selected subset of objects. We compare the proposed availability attack framework with the following attack strategy.

Baseline. The attacker first runs the truth discovery algorithm (CRH) on the observations provided by the normal users. Then the attacker sets each malicious user's observation on a given object as the candidate answer which has the second highest probability value based on the truth discovery result on this object. For example, for one question, CRH outputs the aggregation result as $(0.6, 0.1, 0.2, 0.1)$. Then the malicious users who are assigned to this question will provide observations as $(0, 0, 1, 0)$. This baseline method is intuitive since this candidate answer is more likely to win over the estimated object truth before the attack than other candidate answers. In fact, it is the optimal attack strategy if the aggregation method is voting (that is, for each object, the candidate answer which has the highest vote counts is the aggregation result).

For the proposed attack framework, the optimal observations for each malicious user are calculated according to Algorithm 4. We set $\theta = 100$ and initialize the observations of malicious users on an object as the truth discovery results from the normal users' observations on that object. In order to evaluate the performance of the availability attack strategies, we adopt two metrics: the *utility* defined in Eq. (4.14), and the *change rate*. For the latter, it is defined as the percentage of the objects which has different final aggregation results before and after the attack. It is equivalent to the utility defined in Eq. (4.10). All the experiments are conducted 20 times and we report the average results.

The Effect of the Percentage of Malicious Users

Here we assume that each malicious user can observe 10 randomly selected objects. Then we vary the percentage of malicious users from 0.03 to 0.3 and calculate the attacker's utility and the change rate. The results are shown in Figure 4.2, from which we

can see that the proposed optimal attack framework outperforms the baseline method in all cases. This figure also shows that the advantage of the proposed attack framework is marginal when the percentage of malicious users is 0.03. This is because the number of malicious users in this case is too small, and it is hard to change the aggregation results much. However, the advantage of the proposed attack framework becomes bigger when the percentage of malicious users gradually increases. To change the aggregation results on 20% of the objects, the proposed attack framework only needs less than 12% of malicious users whereas the baseline method needs about 21% of malicious users. For the proposed attack framework, the increment of change rate slows down after the malicious users occupy 15% of the total users, but the utility keeps increasing steadily. The reason is that the change of an estimated object truth is either 0 (not changed) or 1 (changed), while the utility is a continuous value. For example, the estimated object truth vector that changes from $(0.7, 0.2, 0.1, 0)$ to $(0.5, 0.4, 0.1, 0)$ does not increase the change rate, but increases the utility.

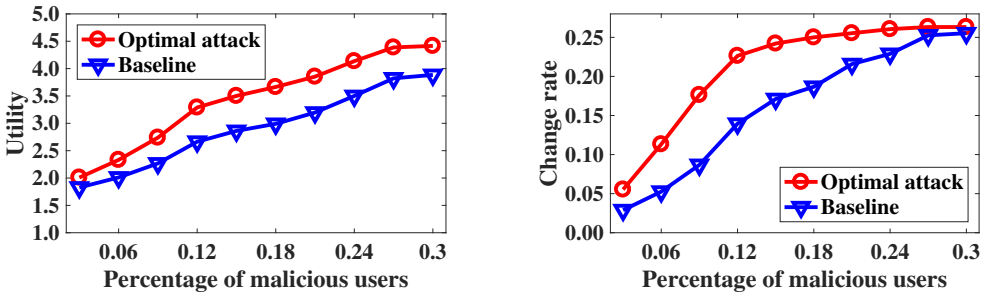


Figure 4.2: Utility and Change rate w.r.t. the percentage of malicious users for availability attack

The Effect of the Number of the Observed Objects

With the fixed number of malicious users, if one malicious user can observe more objects, he can make impact to more objects, and thus achieve higher impact to the overall sensing system. In this experiment, we examine the effect of the number of the objects a malicious user can observe. Here we fix the percentage of malicious users to be 10%. Then we vary the number of objects that each malicious user can observe from 2 to

18. The results are reported in Figure 4.3. The results clearly demonstrate the advantage of the proposed attack framework over the baseline method. With the increment of the observed objects, the malicious users exert more and more impact on the sensing system, and the advantage of the proposed attack framework over the baseline method also increases. Figure 4.3 shows that to achieve 20% change rate, the malicious users of the proposed attack framework only need to observe on 10 objects, while the malicious users of the baseline method needs to observe on 18 objects.

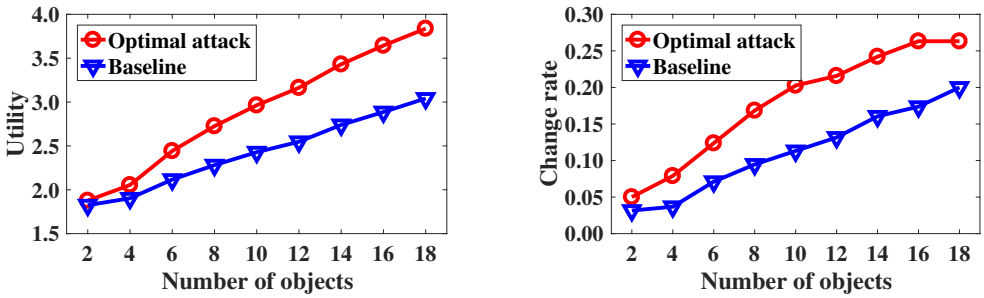


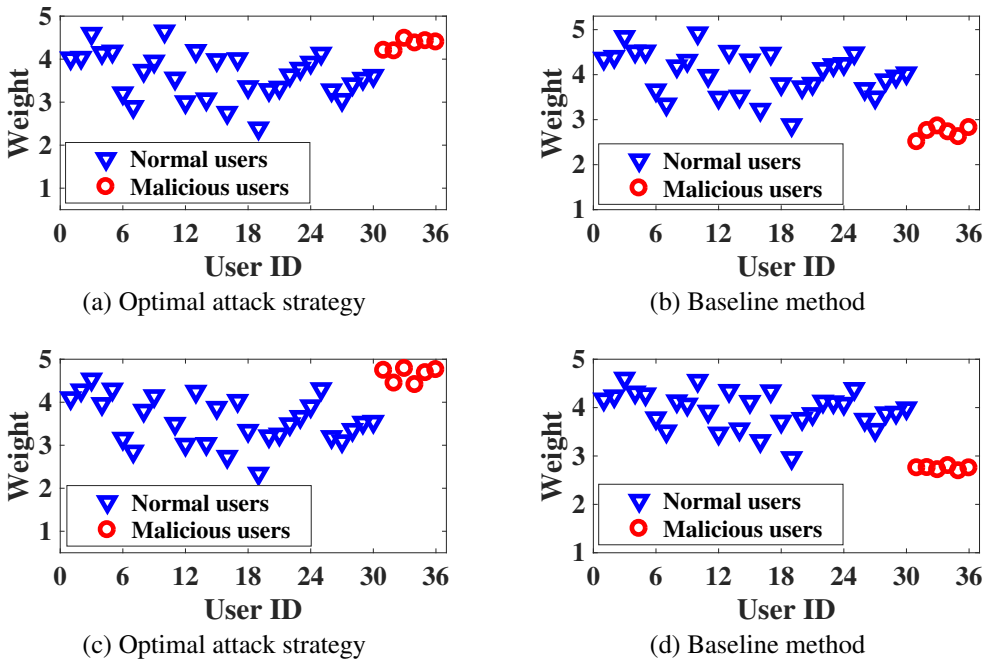
Figure 4.3: Utility and Change rate w.r.t. the number of the objects which are observed by each malicious user

Comparison on Weights of the Malicious Users

The reason that the proposed attack framework outperforms the baseline method lies in the fact that the effect of user reliability estimation in the truth discovery algorithm is considered. The proposed attack framework will let the malicious user "fake" like a normal user or even a good user on some objects to enhance its weight. Whereas for the baseline method, the malicious users always disagree with the majority of the normal users, and thus the suspicious behavior may be detected by the truth discovery algorithm and cause the decrease in the weights.

In this experiment, we examine the weight distributions for both the normal users and the malicious users. We choose the following two settings: the percentage of malicious users is set as 0.15 (i.e., 6 malicious users), and we let them observe 5 objects and 15 objects respectively. In Figure 4.4, we plot the weights for all users after the proposed attack framework attacks the sensing system, and the weights for all users after the

baseline method attacks the sensing system for the aforementioned two scenarios. From Figures 4.4a and 4.4c, we can see that the malicious users from the proposed attack framework all have high weights comparing with the normal users. This means that the malicious users successfully blend into the normal users. Therefore, it is hard for the truth discovery algorithm to detect the attack. In contrast, the malicious users from the baseline method all have very low weights comparing with the normal users, as shown in Figures 4.4b and 4.4d. The two figures confirm our expectations that the truth discovery algorithm finds these malicious users since they behave differently from the normal users. The low user weights not only limit the impact of the malicious users, but also make them vulnerable to straightforward defense mechanism.



(a) Optimal attack strategy

(b) Baseline method

(c) Optimal attack strategy

(d) Baseline method

Figure 4.4: The weight of each user for availability attack. (a) and (b) show the user weights when each malicious user observe 5 objects. (c) and (d) show the user weights when each malicious user observe 15 objects.

4.5.2 Target Attack

In the target attack, the attacker tries to skew the truth discovery results to the target values on certain objects. To make the problem more interesting, we assume that the target values are not the same as the values that have the highest probabilities derived by the truth discovery algorithm before attack; and to make the target attack different from the availability attack, we further assume that not all the target values are the same as the values with the second highest probabilities. We compare the proposed target attack framework with the following attack strategy.

Baseline. For the target objects, the attacker sets the malicious users' observations as the target choices.

For the proposed attack framework, we set $\theta = 100$ and initialize the observations of malicious users on an object as the truth discovery results from the normal users on that object. In order to evaluate the performance of the target attack strategies, we adopt two metrics: the *utility* defined in Eq. (4.27), and the *change rate*. For the latter, it is defined as the number of objects that are successfully changed to the target value divided by the total number of target objects. It is equivalent to the utility defined in Eq. (4.25).

The Effect of the Percentage of Malicious Users and the Number of Target Objects

In this experiment, we examine how the percentage of malicious users and the number of the target objects affect the attack results. We vary the percentage of malicious users from 0.03 to 0.27 with 10 and 15 target objects. The results are plotted in Figure 4.5. We can still observe that the proposed attack framework outperforms the baseline method in all cases. The proposed attack framework can usually use one or two fewer malicious users to achieve the same change rate comparing with the baseline method. The effect of the percentage of malicious users in the target attack is similar to that of in the availability attack: the more malicious users, the higher the utility and the change rate. Increasing the number of the target objects, however, makes the attack goal harder to achieve. This is because that under our problem settings, the target attack is significantly more difficult than the availability attack, since the target values may be

supported by much fewer normal users. Therefore, when there are more target objects, the attacker needs to add more malicious users to achieve the same change rate.

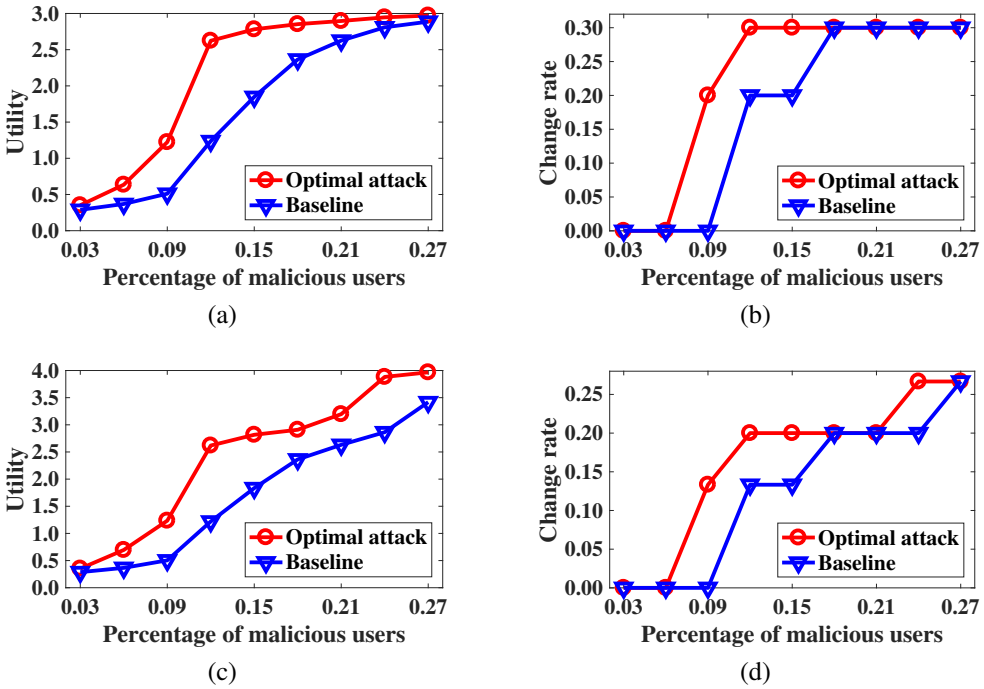
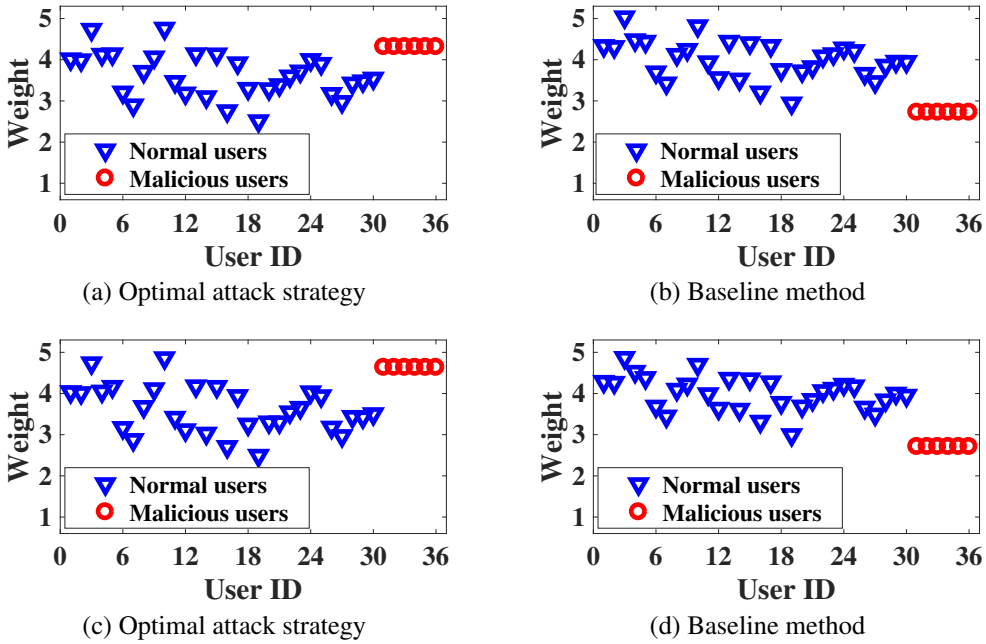


Figure 4.5: Utility and Change rate w.r.t. the percentage of malicious users for target attack. (a) and (b) show the results when 10 objects are attacked. (c) and (d) show the results when 15 objects are attacked.

Comparison on Weights of the Malicious Users

Next, we explore the user weight distributions in the target attack. The following two settings are compared: the percentage of malicious users is set as 0.15, (i.e., 6 malicious users), and we let the number of target objects to be 10 and 15 respectively. In Figure 4.6, we plot the weights for all users after the proposed attack framework attacks the sensing system, and the weights for all users after the baseline method attacks the sensing system. From Figures 4.6a and 4.6c, we can see that the weights of the malicious users from the proposed attack framework are similar to the weights of the normal users, so they again successfully blend into the normal users. The reason is that for some target objects where the target values are too hard to achieve, the malicious users may disguise their purpose by agreeing with the normal users. In contrast, the malicious users from the baseline method all have very low weights (Figures 4.6b and 4.6d), as they always

choose the choices that are not supported by the normal users. The results suggest that the malicious users from the baseline method are more vulnerable to straightforward defense mechanisms.



(a) Optimal attack strategy

(b) Baseline method

(c) Optimal attack strategy

(d) Baseline method

Figure 4.6: The weight of each user for target attack. (a) and (b) show the results when 10 objects are attacked. (c) and (d) show the results when 15 objects are attacked.

4.6 Summary

In this chapter, we study two types of data poisoning attacks, i.e., the availability attack and the target attack, against a crowd sensing system empowered with the truth discovery mechanism. We first analyze the pitfalls when attacking such a crowd sensing system and then design an optimal attack framework to derive the (approximately) optimal attack strategy, based on which the attacker can not only maximize his attack utility but also successfully disguise the attack behaviors. The proposed optimal attack framework is tested on a real-world crowd sensing system. The experimental results demonstrate that compared with the naive baseline schemes, the proposed attack framework can achieve higher attack utility and at the same time, let the malicious users gain higher reliability degrees such that they cannot be detected easily.

Chapter 5

Security Vulnerability Analysis for the Dawid-Skene Model

5.1 Introduction

In Chapter 4, we investigate data poisoning attacks against the crowd sensing systems empowered with truth discovery mechanism. In this chapter, we study the crowd sensing systems employing another widely adopted reliability-aware data aggregation method, i.e., the Dawid-Skene model [22], and analyze its security vulnerability to data poisoning attacks. Here we consider a scenario in which the attacker aims to maximize the error of the final results through creating or recruiting a group of malicious users and letting them provide manipulated data. This attack goal can be easily achieved if the attacker has the capability of creating or recruiting an overwhelming number of malicious users. However, in practice, the attacker usually has limited resources and he can only control a few malicious users. In such cases, the attack strategy plays an important role.

A naive attack strategy is to let the malicious users always disagree with the normal users. If some straightforward aggregation methods, such as majority voting, are used to aggregate the data, this naive attack model may be the optimal choice, since every malicious user exerts the most influence in the aggregation. However, the story would

become much more complicated if the Dawid-Skene model is employed. In the Dawid-Skene model, each user is associated with an underlying confusion matrix, which can reflect the reliability degree of this user. After the labels are collected from the users, the final results and the users' confusion matrices are jointly estimated based on the maximum likelihood principle. As a result, users with low reliability degrees will have low impact in the aggregation. In this case, if an attacker adopts the aforementioned naive attack, in which the malicious users always disagree with the normal users, the malicious users are very likely to be assigned a significantly low reliability degree by the Dawid-Skene model, and thus will not be able to make any difference in the final aggregated results.

To attack a crowd sensing system with the Dawid-Skene model empowered, we propose an intelligent data poisoning attack mechanism that takes into account the malicious users' reliability degrees. In this mechanism, the malicious users behave more "intelligently", i.e., try to improve their reliability degrees by agreeing with the normal users on some objects whose values are unlikely to be overturned. Compared with the aforementioned naive strategy, the proposed intelligent attack model can not only disguise the malicious users, but also enable them to launch more effective attacks on the objects that are more vulnerable to attack.

Towards this end, we formulate a bi-level optimization problem. The objective in the optimization problem is to maximize the attacker's utility, which is the combination of the number of the successfully attacked objects and the malicious users' reliability degrees. Since the number of the successfully attacked objects is discrete, it is hard to directly solve the optimization problem. To address this challenge, a continuous and differentiable sigmoid function is adopted to approximate the discrete component in the objective function. We solve the bi-level optimization problem by iteratively solving the upper-level and lower-level subproblems, which are solved by the projected gradient ascent and expectation-maximization (EM) methods, respectively.

5.2 Problem Setting

In this chapter, we consider a crowd sensing scenario in which a cloud server and some participating users are involved. The cloud server is a platform which can outsource the sensing tasks to the participating users. The sensing task is to collect labels for a pool of objects, each of which belongs to one of two possible categories (e.g., same/different; positive/negative; etc.). To ensure the quality of the final result, each object will be observed by multiple participating users, who are the individuals that carry out the crowd sensing tasks, and each user will provide labels for a number of objects. After collecting the labels from the participating users, the cloud server aggregates these labels to derive the true label of each object.

The security threats considered in this chapter mainly come from an attacker who aims to attack the crowd sensing system for malicious purposes. *The goal of the attacker is to maximize the error of the derived true labels, and meanwhile disguise his malicious behaviors so that the attack cannot be detected easily*. We assume that the attacker can recruit or create multiple participating users (called *malicious users*) and arbitrarily manipulate their labels, but he cannot influence the behaviors of the *normal users* who carry out the sensing tasks without any malicious purpose. If there is no limitation on the ability of the attacker, he can achieve the attack goal easily through creating a large number of malicious users. However, in practice, the attacker usually has limited resources and can only recruit or create a few malicious users. In such cases, it is essential for the attacker to design a sophisticated *attack strategy* (i.e., the labels provided by the malicious users) such that the attack goal can be maximally achieved. In order to assess the vulnerability of the crowd sensing system in the worst case, we also assume that the attacker has full knowledge of the aggregation method and the labels from normal users. This assumption is reasonable as it is possible for the attacker to learn the labels of normal users through eavesdropping the communications between the cloud server and the normal users.

Problem formulation. Suppose the cloud server releases a sensing task which contains a set of objects $O = \{o_1, o_2, ..., o_M\}$, and these objects are queried to K normal users which are represented as $U = \{u_1, u_2, ..., u_K\}$. The labels provided by these normal users are denoted as $X = \{x_m^k\}_{m,k=1}^{M,K}$, in which x_m^k is the label provided by user u_k for object o_m. For each object o_m, there is a true label x_m^* which is unknown *a priori* and needs to be estimated by the cloud server based on the labels collected from all the users. We use $X^* = \{x_m^*\}_{m=1}^{M}$ to denote the set of true labels for all objects. Assume that the attacker can create K' malicious users represented as $\widetilde{U} = \{\widetilde{u}_1, \widetilde{u}_2, ..., \widetilde{u}_{K'}\}$. The set of labels provided by all the malicious users is denoted as $\widetilde{X} = \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'}$, and $\widetilde{x}_m^{k'}$ is the label provided by malicious user $\widetilde{u}_{k'}$ for object o_m. *Our goal in this chapter is to find an optimal attack strategy (i.e., an optimal $\widetilde{X}$) from the perspective of the attacker such that the attack goal can be maximally achieved.*

5.3 Preliminary

In the Dawid-Skene model, each participating user is associated with an unknown confusion matrix which reflects the user's ability (or reliability degree) when carrying out the sensing task. Each diagonal element in this matrix represents the probability that the user provides the true label for a particular object, while each off-diagonal element represents the probability that a particular wrong label is provided. After the labels are collected from all the users, the maximum likelihood estimation method is adopted to jointly estimate each object's true label and each user's confusion matrix.

In this chapter, we consider the binary case, i.e., we assume that each object has only two possible labels: 0 and 1. Based on the Dawid-Skene model, each user u_k provides label for object o_m according to parameters $\alpha_k = \Pr(x_m^k = 1 | x_m^* = 1)$ and $\beta_k = \Pr(x_m^k = 0 | x_m^* = 0)$, where α_k and β_k are the diagonal elements in user u_k's confusion matrix. They are also treated as u_k's ability parameters or reliability degrees. The larger α_k and β_k are, the higher the probability that user u_k provides a true label.

Additionally, this model assumes that the probability that an object drawn at random has true label 1 is p, which is usually unknown *a priori*. Denote $\Theta = \{p, \{\alpha_k, \beta_k\}_{k=1}^K\}$ as the set of all the model parameters. The Dawid-Skene model adopts the maximum likelihood estimation method to estimate Θ. However, due to the latent variables X^* are unknown *a priori*, it is hard to directly conduct the estimation.

To address the above challenge, the Dawid-Skene model adopts the EM algorithm [24] which contains an expectation step (E-step) and a maximization step (M-step). In the E-step, the objects' true labels are derived based on the estimated model parameters Θ, and in the M-step, the parameters Θ are calculated based on the derived true labels. The details of the two steps are described as follows.

E-step: In this step, the model parameters Θ are fixed. For each object o_m, we calculate $\omega_m = \Pr\{x_m^* = 1|X\}$ based on the Bayes theorem:

$$\begin{aligned}\omega_m = \Pr\{x_m^* = 1|X;\Theta\} &= \frac{\Pr\{X|x_m^*=1\}\cdot p}{\Pr\{X|x_m^*=1\}\cdot p + \Pr\{X|x_m^*=0\}\cdot(1-p)} \\ &= \frac{\prod_{k\in U_m}\alpha_k^{x_m^k}(1-\alpha_k)^{1-x_m^k}\cdot p}{\prod_{k\in U_m}\alpha_k^{x_m^k}(1-\alpha_k)^{1-x_m^k}\cdot p + \prod_{k\in U_m}\beta_k^{1-x_m^k}(1-\beta_k)^{x_m^k}\cdot(1-p)},\end{aligned} \tag{5.1}$$

where U_m represents the set of normal users who provide labels for object o_m.

Here ω_m is the posterior probability that the true label of the object o_m is 1. With the calculated $\Omega = \{\omega_m\}_{m=1}^M$, the expected value of the log likelihood function can be expressed as

$$\begin{aligned}Q(\Theta) =& E[\log L(\Theta; X, X^*)] = E[\log \prod_{m=1}^{M} L(\Theta; X_m, x_m^*)] \\ =& \sum_{m=1}^{M}\{\omega_m \log[\prod_{k\in U_m}\alpha_k^{x_m^k}(1-\alpha_k)^{1-x_m^k}\cdot p] \\ &+ (1-\omega_m)\log[\prod_{k\in U_m}\beta_k^{1-x_m^k}(1-\beta_k)^{x_m^k}\cdot(1-p)]\},\end{aligned} \tag{5.2}$$

where X_m represents the set of labels for object o_m.

M-step: In this step, the posterior probabilities $\{\omega_m\}_{m=1}^{M}$ are fixed. The model parameters Θ are estimated by maximizing the expected value of the log likelihood function $Q(\Theta)$, and they are updated as follows:

$$p = \frac{\sum_{m=1}^{M} \omega_m}{M}, \tag{5.3}$$

$$\alpha_k = \frac{\sum_{m \in O_k} \omega_m \cdot x_m^k}{\sum_{m \in O_k} \omega_m}, \tag{5.4}$$

$$\beta_k = \frac{\sum_{m \in O_k} (1 - \omega_m) \cdot (1 - x_m^k)}{\sum_{m \in O_k} (1 - \omega_m)}, \tag{5.5}$$

where O_k represents the set of objects queried to u_k.

The above two steps are iteratively conducted until the convergence criterion is satisfied. Finally, if ω_m is larger than 0.5, the true label of the object o_m is assigned as 1, otherwise, it is assigned as 0.

5.4 The Intelligent Attack Mechanism

In order to achieve the attack goal as much as possible, it is essential for the attacker to find an optimal attack strategy with the limited resources (i.e., the number of created or recruited malicious users and the number of observed objects). We first investigate the Dawid-Skene model under the adversarial environment in Section 5.4.1, and then discuss how to design an optimal attack strategy from the perspective of the attacker in Section 5.4.2.

5.4.1 Dawid-Skene Model with Malicious Users

In the adversarial environment, the malicious users may blend into the crowd sensing system and provide manipulated labels to the cloud server in order to distort the final

aggregated results. In this section, we decompose the participating users into normal and malicious ones, and investigate the relationship between the final aggregation results and the labels provided by malicious users. Please note that the cloud server in the crowd sensing system cannot differentiate the two types of participating users when aggregating the collected labels.

As described in the problem setting, we assume that the attacker creates K' malicious users to conduct the data poisoning attacks against the crowd sensing system. The attacker cannot influence the behaviors of the normal users, but he can arbitrarily manipulate the labels of malicious users. We denote the ability parameters of malicious user $\widetilde{u}_{k'}$ in the Dawid-Skene model as $\widetilde{\alpha}_{k'} = \Pr(\widetilde{x}_m^{k'} = 1 | x_m^* = 1)$ and $\widetilde{\beta}_{k'} = \Pr(\widetilde{x}_m^{k'} = 0 | x_m^* = 0)$. We use $\widetilde{\Theta} = \{p, \{\alpha_k, \beta_k\}_{k=1}^K, \{\widetilde{\alpha}_{k'}, \widetilde{\beta}_{k'}\}_{k'=1}^{K'}\}$ to denote the set of the model parameters and $\{\alpha_k, \beta_k\}_{k=1}^K$ are the ability parameters of the normal users. Suppose $\widehat{X}$ is the set of the labels provided by all the participating users, including the normal and malicious ones. The E-step and M-step in the Dawid-Skene model after data poisoning attacks can be described as follows:

E-step: For each object o_m, we calculate $\widetilde{\omega}_m = \Pr\{x_m^* = 1 | \widehat{X}\}$ based on the Bayes theorem:

$$\begin{aligned} \widetilde{\omega}_m &= \Pr\{x_m^* = 1 | \widehat{X}; \widetilde{\Theta}\} \\ &= \frac{\Pr\{\widehat{X} | x_m^* = 1\} \cdot p}{\Pr\{\widehat{X} | x_m^* = 1\} \cdot p + \Pr\{\widehat{X} | x_m^* = 0\} \cdot (1-p)} \\ &= \frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1} + \widetilde{A}_{m0}}, \end{aligned} \tag{5.6}$$

where

$$\widetilde{A}_{m1} = \prod_{k \in U_m} \alpha_k^{x_m^k} (1-\alpha_k)^{1-x_m^k} \cdot \prod_{k' \in \widetilde{U}_m} \widetilde{\alpha}_{k'}^{\widetilde{x}_m^{k'}} (1-\widetilde{\alpha}_{k'})^{1-\widetilde{x}_m^{k'}} \cdot p \tag{5.7}$$

$$\widetilde{A}_{m0} = \prod_{k \in U_m} \beta_k^{1-x_m^k} (1-\beta_k)^{x_m^k} \cdot \prod_{k' \in \widetilde{U}_m} \widetilde{\beta}_{k'}^{1-\widetilde{x}_m^{k'}} (1-\widetilde{\beta}_{k'})^{\widetilde{x}_m^{k'}} \cdot (1-p). \tag{5.8}$$

Here we use $\widetilde{U}_m$ to denote the set of malicious users who provide labels for object o_m. $\widetilde{\omega}_m$ represents the posterior probability that the true label of object o_m is 1 after the data poisoning attacks.

M-step: In this step, we fix the the posterior probabilities $\{\widetilde{\omega}_m\}_{m=1}^{M}$ and update the model parameters $\widetilde{\Theta} = \{p, \{\alpha_k, \beta_k\}_{k=1}^{K}, \{\widetilde{\alpha}_{k'}, \widetilde{\beta}_{k'}\}_{k'=1}^{K'}\}$ as follows:

$$p = \frac{\sum_{m=1}^{M} \widetilde{\omega}_m}{M}, \tag{5.9}$$

$$\alpha_k = \frac{\sum_{m \in O_k} \widetilde{\omega}_m \cdot x_m^k}{\sum_{m \in O_k} \widetilde{\omega}_m}, \quad \beta_k = \frac{\sum_{m \in O_k} (1 - \widetilde{\omega}_m) \cdot (1 - x_m^k)}{\sum_{m \in O_k} (1 - \widetilde{\omega}_m)}, \tag{5.10}$$

$$\widetilde{\alpha}_{k'} = \frac{\sum_{m \in \widetilde{O}_{k'}} \widetilde{\omega}_m \cdot \widetilde{x}_m^{k'}}{\sum_{m \in \widetilde{O}_{k'}} \widetilde{\omega}_m}, \quad \widetilde{\beta}_{k'} = \frac{\sum_{m \in \widetilde{O}_{k'}} (1 - \widetilde{\omega}_m) \cdot (1 - \widetilde{x}_m^{k'})}{\sum_{m \in \widetilde{O}_{k'}} (1 - \widetilde{\omega}_m)}, \tag{5.11}$$

where $\widetilde{O}_{k'}$ represents the set of objects observed by $\widetilde{u}_{k'}$.

The above equations show that once the labels of normal users (i.e., X) are given, the final estimated true labels of the objects and the ability parameters (i.e., $\{\alpha_k, \beta_k\}_{k=1}^{K}, \{\widetilde{\alpha}_{k'}, \widetilde{\beta}_{k'}\}_{k'=1}^{K'}$) of the participating users are only dependent on the malicious users' data. Different values of the malicious users' labels can lead to different estimated results. Based on this fact, the attacker can conduct data poisoning attacks through carefully designing the malicious users' labels such that the goal of the attacker can be optimally achieved.

5.4.2 Optimal Attack Strategy

In this chapter, the attacker conducts the data poisoning attacks for the purpose of maximizing the error of the final aggregated results, and at the same time tries to disguise his attack behaviors as much as possible. We can understand the goal of the attacker in two aspects. On one hand, the attacker aims to maximize the deviation between the outputs of the Dawid-Skene model before and after the data poisoning attacks. In other words,

the attacker wants to maximize the number of the successfully attacked objects, where we say an object is attacked successfully if the estimated true label is changed from one label to the other after taking the malicious users' labels into account. On the other hand, the attacker wants to disguise the malicious users as normal users in the crowd sensing system such that the attack behaviors cannot be detected easily. One way to achieve the disguise is to get high values on the malicious users' ability parameters (or reliability degrees), i.e., $\{\widetilde{\alpha}_{k'}\}_{k'=1}^{K'}$ and $\{\widetilde{\beta}_{k'}\}_{k'=1}^{K'}$. Since the users with large ability parameters will be treated as high-quality users in the Dawid-Skene model, the crowd sensing system then cannot distinguish the malicious users from the normal users. In this section, we stand on the attacker's position and discuss how to find an optimal attack strategy so that the goal of the attacker can be achieved as much as possible.

Suppose the attacker is able to create or recruit K' malicious users, and for each malicious user, the observed objects are given. When conducting the data poisoning attacks to break the crowd sensing system, the attacker needs to find the optimal assignments for the malicious users' labels. An intuitive strategy is let the malicious users provide the label which is not likely to be true for each observed object. This strategy may work well when the aggregation method is majority voting. But for the Dawid-Skene model, it is not the optimal choice, especially when only a few malicious users are created or recruited. Due to the fact that malicious users always disagree with the majority, the Dawid-Skene model will assign low ability values to these malicious users, and consequently, their impact will also be decreased. In such way, the malicious users can be detected easily and the attack may fail on all the objects. Thus the ability parameters of malicious users (i.e., $\{\widetilde{\alpha}_{k'}\}_{k'=1}^{K'}$ and $\{\widetilde{\beta}_{k'}\}_{k'=1}^{K'}$) should be taken into account when finding the optimal attack strategy.

In order to address the above challenge, we formulate the goal of the attacker as the following

$$\max_{\widetilde{X}} \quad \sum_{m=1}^{M} \mathbb{1}(x_m^{*a} \neq x_m^{*b}) + \lambda \sum_{k'=1}^{K'} (\widetilde{\alpha}_{k'} + \widetilde{\beta}_{k'}) \tag{5.12}$$

$$\text{s.t.} \quad \{X^{*a}, \widetilde{\Theta}\} = \underset{X^{*a}, \widetilde{\Theta}}{\operatorname{argmax}} \ \log L(\widetilde{\Theta}; \widehat{X}, X^{*a})$$
$$\{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'} \in \{0, 1\}$$

where $X^{*a} = \{x_m^{*a}\}_{m=1}^{M}$ denotes the set of the estimated true labels after the data poisoning attacks and x_m^{*b} denotes the estimated true label for object o_m before the attacks (i.e., calculated based on the labels of normal users). Once the normal users' labels are given, x_m^{*b} is a constant. The objective function contains two components. The first component, i.e., $\sum_{m=1}^{M} \mathbb{1}(x_m^{*a} \neq x_m^{*b})$, where $\mathbb{1}(\cdot)$ is the indicator function, represents the number of the successfully attacked objects. In the second component, $\sum_{k'=1}^{K'}(\widetilde{\alpha}_{k'} + \widetilde{\beta}_{k'})$ is the summation of the malicious users' ability parameters, and λ is a parameter used to trade off the two components. The summation of the two components can also be treated as the utility of the attacker. The intuition of the objective function is to maximize the number of the successfully attacked objects and the malicious users' ability values simultaneously, where the first component is the goal of the attack and the latter ensures that the malicious users cannot be detected easily. In this optimization problem, the Dawid-Skene model becomes a constraint. This is a bi-level optimization problem [3]. The optimization over the labels of the malicious users (i.e., $\widetilde{X}$) is the upper-level problem, and the optimization over $\{X^{*a}, \widetilde{\Theta}\}$ is the lower-level problem.

In the Dawid-Skene model, the final estimated true labels of the objects are dependent on the posterior probabilities $\Omega = \{\omega_m\}_{m=1}^{M}$ or $\widetilde{\Omega} = \{\widetilde{\omega}_m\}_{m=1}^{M}$: if ω_m (or $\widetilde{\omega}_m$) is larger than 0.5, x_m^{*b} (or x_m^{*a}) is assigned as 1, otherwise, it is assigned as 0. Thus we can reformulate optimization problem (5.12) as follows:

$$\begin{aligned} \max_{\widetilde{X}} \quad & \sum_{m=1}^{M} \frac{1}{2}\{1 - \operatorname{sgn}[(\omega_m - 0.5) \cdot (\widetilde{\omega}_m - 0.5)]\} + \lambda \sum_{k'=1}^{K'}(\widetilde{\alpha}_{k'} + \widetilde{\beta}_{k'}) \\ \text{s.t.} \quad & \{\widetilde{\Omega}, \widetilde{\Theta}\} = \underset{\widetilde{\Omega}, \widetilde{\Theta}}{\operatorname{argmax}} \ \log L(\widetilde{\Theta}; \widehat{X}, \widetilde{\Omega}) \\ & \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'} \in \{0, 1\}, \end{aligned} \tag{5.13}$$

where

$$\mathrm{sgn}[(\omega_m - 0.5)\cdot(\widetilde{\omega}_m - 0.5)] = \begin{cases} 1 & if\ (\omega_m - 0.5)\cdot(\widetilde{\omega}_m - 0.5) > 0 \\ 0 & if\ (\omega_m - 0.5)\cdot(\widetilde{\omega}_m - 0.5) = 0 \\ -1 & if\ (\omega_m - 0.5)\cdot(\widetilde{\omega}_m - 0.5) < 0. \end{cases} \tag{5.14}$$

Once the labels of normal users are given, the posterior probability ω_m for object o_m is a constant. $\widetilde{\omega}_m$, $\widetilde{\alpha}_{k'}$ and $\widetilde{\beta}_{k'}$ are dependent on the labels of the malicious users (i.e., the attack strategy $\widetilde{X}$) and they can be different when the malicious users vary their labels. In this way, $\widetilde{\omega}_m$, $\widetilde{\alpha}_{k'}$ and $\widetilde{\beta}_{k'}$ can be expressed as the functions of $\widetilde{X}$ according to Eq. (5.6) and Eq. (5.11). Then problem (5.13) becomes:

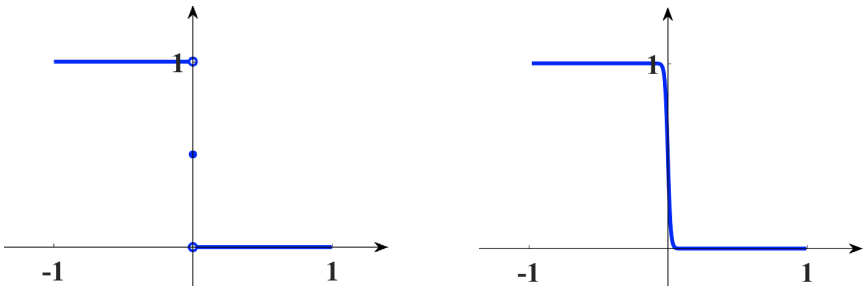
$$\begin{aligned} \max_{\widetilde{X}} \quad & \sum_{m=1}^{M} \frac{1}{2}\{1 - \mathrm{sgn}[(\omega_m - 0.5)\cdot(\frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1}+\widetilde{A}_{m0}} - 0.5)]\} \\ & + \lambda \sum_{k'=1}^{K'} (\frac{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m \cdot \widetilde{x}_m^{k'}}{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m} + \frac{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)\cdot(1-\widetilde{x}_m^{k'})}{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)}) \\ \text{s.t.} \quad & \{\widetilde{\Omega}, \widetilde{\Theta}\} = \underset{\widetilde{\Omega},\widetilde{\Theta}}{\mathrm{argmax}}\ \log L(\widetilde{\Theta}; \widehat{X}, \widetilde{\Omega}) \\ & \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'} \in \{0,1\}. \end{aligned} \tag{5.15}$$

Since the objective function in problem (5.15) is not continuous, it is hard to directly solve this optimization problem. In order to address this challenge, we approximate the objective function in problem (5.15) by the following one:

$$\begin{aligned} \max_{\widetilde{X}} \quad & \sum_{m=1}^{M} \{1 - \frac{1}{1+\exp[-\theta(\omega_m - 0.5)\cdot(\frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1}+\widetilde{A}_{m0}} - 0.5)]}\} \\ & + \lambda \sum_{k'=1}^{K'} (\frac{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m \cdot \widetilde{x}_m^{k'}}{\sum_{m\in\widetilde{O}_{k'}} \widetilde{\omega}_m} + \frac{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)\cdot(1-\widetilde{x}_m^{k'})}{\sum_{m\in\widetilde{O}_{k'}} (1-\widetilde{\omega}_m)}). \end{aligned} \tag{5.16}$$

The basic idea behind the approximation is that function $h_1(x) = \frac{1}{2}(1 - \mathrm{sgn}\, x)$ can be approximated by function $h_2(x) = 1 - \frac{1}{1+\exp(-\theta x)}$ when $x \in (-1, 1)$. The parameter

θ in $h_2(x)$ represents the steepness of the curve. The curves of the two functions when $\theta = 100$ are shown in Figure 5.1. We can see $h_2(x)$ is a good approximation of $h_1(x)$. Additionally, the continuous property of $h_2(x)$ allows us to solve the optimization problem based on the objective function (5.16).



(a) $h_1(x)$ (b) $h_2(x)$ with $\theta = 100$

Figure 5.1: Curves of $h_1(x)$ and $h_2(x)$

Another challenge when solving the above optimization problem is that each element in $\widetilde{X}$ has a categorical value (0 or 1). This introduces difficulties when solving the upper-level problem. In this chapter, we relax the values of the elements in $\widetilde{X}$ to the range $[0, 1]$ such that the optimization problem can be solved according to the gradient-based methods. In other words, we treat $\widetilde{x}_m^{k'}$ as the probability that malicious user $\widetilde{u}_{k'}$ provides label 1 for object o_m. Finally, the value of $\widetilde{x}_m^{k'}$ will be transformed to categorical data: if the probability is larger than 0.5, $\widetilde{x}_m^{k'}$ is assigned as 1, otherwise, it is assigned as 0. Then the attacker needs to solve the following optimization problem in order to get the optimal attack strategy:

$$
\begin{aligned}
\max_{\widetilde{X}} \quad & f(\widetilde{X}) = \sum_{m=1}^{M} \{1 - \frac{1}{1 + \exp[-\theta(\omega_m - 0.5) \cdot (\frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1} + \widetilde{A}_{m0}} - 0.5)]}\} \\
& + \lambda \sum_{k'=1}^{K'} (\frac{\sum_{m \in \widetilde{O}_{k'}} \widetilde{\omega}_m \cdot \widetilde{x}_m^{k'}}{\sum_{m \in \widetilde{O}_{k'}} \widetilde{\omega}_m} + \frac{\sum_{m \in \widetilde{O}_{k'}} (1 - \widetilde{\omega}_m) \cdot (1 - \widetilde{x}_m^{k'})}{\sum_{m \in \widetilde{O}_{k'}} (1 - \widetilde{\omega}_m)}) \\
\text{s.t.} \quad & \{\widetilde{\Omega}, \widetilde{\Theta}\} = \underset{\widetilde{\Omega}, \widetilde{\Theta}}{\operatorname{argmax}} \ \log L(\widetilde{\Theta}; \widehat{X}, \widetilde{\Omega}) \\
& \{\widetilde{x}_m^{k'}\}_{m,k'=1}^{M,K'} \in [0, 1].
\end{aligned}
\tag{5.17}
$$

Next, we discuss how to solve the above optimization problem. The solution we adopted here is a two-step iterative procedure.

Step 1: In this step, we first fix the labels of malicious users, i.e., $\widetilde{X}$, which are estimated in the previous iteration. If it is the first iteration, the elements in $\widetilde{X}$ can be initialized randomly or be set as some particular values. Then we solve the lower-level problem through conducting the E-step and M-step described in Section 5.4.1 to get the optimal parameters $\{\widetilde{\Omega}, \widetilde{\Theta}\}$. Please note that all the elements in $\widetilde{X}$ need to be transformed to the categorical values (i.e., 1 or 0) before solving the lower-level problem.

Step 2: In this step, we fix the parameters $\{\widetilde{\Omega}, \widetilde{\Theta}\}$ calculated in Step 1, and then adopt the projected gradient ascent method to solve the upper-level problem. More specifically, in iteration t, we update $\widetilde{x}_m^{k'}$ as follows:

$$\widetilde{x}_m^{k'(t+1)} \leftarrow \mathrm{Proj}_{[0,1]}(\widetilde{x}_m^{k'(t)} + s_t \cdot \nabla_{\widetilde{x}_m^{k'}} f(\widetilde{X})) \tag{5.18}$$

where s_t is the step size in iteration t and $\mathrm{Proj}_{[0,1]}(\cdot)$ is the projection operator onto the range $[0, 1]$. The gradient $\nabla_{\widetilde{x}_m^{k'}} f(\widetilde{X})$ is calculated as follows:

$$\begin{aligned} \nabla_{\widetilde{x}_m^{k'}} f(\widetilde{X}) = &- \frac{\exp(\theta d_1 d_2)}{[1+\exp(\theta d_1 d_2)]^2} \cdot \theta d_1 \cdot \frac{\partial d_2}{\partial \widetilde{x}_m^{k'}} \\ &+ \lambda \left(\frac{\widetilde{\omega}_m}{\sum_{\bar{m} \in \widetilde{O}_{k'}} \widetilde{\omega}_{\bar{m}}} + \frac{\widetilde{\omega}_m - 1}{\sum_{\bar{m} \in \widetilde{O}_{k'}} (1 - \widetilde{\omega}_{\bar{m}})}\right) \end{aligned} \tag{5.19}$$

where $d_1 = \omega_m - 0.5$, $d_2 = \frac{\widetilde{A}_{m1}}{\widetilde{A}_{m1}+\widetilde{A}_{m0}} - 0.5$. Through combining with Eq. (5.7) and Eq. (5.8), we can calculate $\frac{\partial d_2}{\partial \widetilde{x}_m^{k'}}$ as

$$\frac{\partial d_2}{\partial \widetilde{x}_m^{k'}} = \frac{\frac{\partial \widetilde{A}_{m1}}{\partial \widetilde{x}_m^{k'}} \cdot \widetilde{A}_{m0} - \frac{\partial \widetilde{A}_{m0}}{\partial \widetilde{x}_m^{k'}} \cdot \widetilde{A}_{m1}}{(\widetilde{A}_{m1} + \widetilde{A}_{m0})^2} \tag{5.20}$$

where

$$\frac{\partial \widetilde{A}_{m1}}{\partial \widetilde{x}_m^{k'}} = p \prod_{k \in U_m} \alpha_k^{x_m^k} (1-\alpha_k)^{1-x_m^k} \prod_{\bar{k}' \in \widetilde{U}_m \setminus \{k'\}} \widetilde{\alpha}_{\bar{k}'}^{\widetilde{x}_m^{\bar{k}'}} (1-\widetilde{\alpha}_{\bar{k}'})^{1-\widetilde{x}_m^{\bar{k}'}} \cdot$$
$$[\widetilde{\alpha}_{k'}^{\widetilde{x}_m^{k'}} (1-\widetilde{\alpha}_{k'})^{1-\widetilde{x}_m^{k'}} \log(\widetilde{\alpha}_{k'}) - \widetilde{\alpha}_{k'}^{\widetilde{x}_m^{k'}} (1-\widetilde{\alpha}_{k'})^{1-\widetilde{x}_m^{k'}} \log(1-\widetilde{\alpha}_{k'})], \tag{5.21}$$

$$\frac{\partial \widetilde{A}_{m0}}{\partial \widetilde{x}_m^{k'}} = (1-p) \prod_{k \in U_m} \beta_k^{1-x_m^k} (1-\beta_k)^{x_m^k} \prod_{\bar{k}' \in \widetilde{U}_m \setminus \{k'\}} \widetilde{\beta}_{\bar{k}'}^{1-\widetilde{x}_m^{\bar{k}'}} (1-\widetilde{\beta}_{\bar{k}'})^{\widetilde{x}_m^{\bar{k}'}} \cdot$$
$$[-\widetilde{\beta}_{k'}^{1-\widetilde{x}_m^{k'}} (1-\widetilde{\beta}_{k'})^{\widetilde{x}_m^{k'}} \log(\widetilde{\beta}_{k'}) + \widetilde{\beta}_{k'}^{1-\widetilde{x}_m^{k'}} (1-\widetilde{\beta}_{k'})^{\widetilde{x}_m^{k'}} \log(1-\widetilde{\beta}_{k'})]. \tag{5.22}$$

The above two steps will be iteratively conducted until the convergence criterion is satisfied. In this chapter, we define the convergence criterion as $\sqrt{\sum_{k'=1}^{K'} \sum_{m=1}^{M} (\widetilde{x}_m^{k'(t+1)} - \widetilde{x}_m^{k'(t)})^2} < \delta$, which represents the change of $\widetilde{X}$ in two consecutive iterations being less than a threshold δ. After the attacker get the final $\widetilde{X}$, the elements in $\widetilde{X}$ will be transformed to 0 or 1 and then provided to the cloud server as the labels of the malicious users. The submitted labels $\widetilde{X}$ will be treated as the optimal attack strategy of the attacker. The optimization procedure is summarized as Algorithm 5.

5.5 Attack with Limited Knowledge

In order to assess the vulnerability of the crowd sensing system in the worst case, we consider the full knowledge scenario in the above mechanism and assume that the attacker has complete knowledge of the labels from the normal users (i.e., normal labels) for all objects. In fact, the proposed mechanism can also be employed to implement an effective attack even when the attacker only has limited knowledge of the objects' normal labels.

Suppose the attacker only knows the normal labels for M' ($M' < M$) objects represented as $O' = \{o'_1, o'_2, ..., o'_{M'}\}$. We denote the set of the normal labels for the M'

Algorithm 5: Optimal attack against the Dawid-Skene model

Input: The number of objects: M; the number of normal users: K; the normal users' labels: X; the number of malicious users: K'; the objects observed by the malicious users: $\{\widetilde{O}_{k'}\}_{k'=1}^{K'}$

Output: The optimal attack strategy: $\widetilde{X}$

```
Initialize the optimal attack strategy X̃;
repeat
    Estimate the optimal parameters {Ω̃, Θ̃} through conducting the EM algorithm
      described in Section 5.4.1;
    for each x̃_m^{k'} ∈ X̃ do
        Update x̃_m^{k'} according to Eq. (5.18);
    end
until The convergence criterion is satisfied;
Transform the elements in X̃ to 0 or 1;
return The optimal attack strategy X̃;
```

objects as $X' = \{x'^{k}_{m}\}_{m,k=1}^{M',K}$, which is a subset of X. Since the attacker has no knowledge of the objects except those in O', a good choice for him in such a scenario is to let the malicious users only provide manipulated labels for the objects in O' and try to maximize the error of the final results for the M' objects. In order to achieve the goal, the attacker could treat X' as the surrogate data of X and employ the above proposed mechanism to derive the attack strategy. In other words, the attack strategy in such a scenario can be derived by solving the following optimization problem:

$$\begin{aligned} \max_{\widetilde{X'}} \quad & \sum_{m=1}^{M'} \mathbb{1}(x'^{*a}_{m} \neq x'^{*b}_{m}) + \lambda \sum_{k'=1}^{K'} (\widetilde{\alpha}_{k'} + \widetilde{\beta}_{k'}) \\ \text{s.t.} \quad & \{X'^{*a}, \widetilde{\Theta}\} = \underset{X'^{*a}, \widetilde{\Theta}}{\operatorname{argmax}} \ \log L(\widetilde{\Theta}; \widehat{X}', X'^{*a}) \\ & \{\widetilde{x'}^{k'}_{m}\}_{m,k'=1}^{M',K'} \in \{0,1\}, \end{aligned} \tag{5.23}$$

where $\widetilde{X'} = \{\widetilde{x'}^{k'}_{m}\}_{m,k'=1}^{M',K'}$ is the attack strategy, i.e., the labels provided by the malicious users for the objects in O'. $X'^{*a} = \{x'^{*a}_{m}\}_{m=1}^{M'}$ and $X'^{*b} = \{x'^{*b}_{m}\}_{m=1}^{M'}$ represent the estimated true labels for the M' objects based on $\widehat{X}' = X' \cup \widetilde{X'}$ and X' respectively. Although the attack strategy $\widetilde{X'}$ derived based on Eq. (5.23) may not be as good

as $\widetilde{X}$ based on the full knowledge X, it is the optimal choice for the attacker in the limited knowledge scenario. The performance of the proposed mechanism with limited knowledge is evaluated in Section 5.6.5.

5.6 Performance Evaluation

We conduct experiments based on real-world crowd-contributed datasets to verify the performance of the proposed intelligent attack mechanism.

5.6.1 Experiment Setup

In this section, we introduce the adopted real-world crowd-contributed datasets, the baseline methods which are compared with the proposed mechanism, and the performance measure.

Datasets

To verify the advantages of the proposed intelligent attack mechanism, we adopt the following real-world crowd-contributed datasets.

Duchenne Smile Dataset [105]. In this dataset, the task is to judge whether the simile in a face image (an object) is Duchenne (enjoyment smile) or Non-Duchenne. The authors in [105] create tasks on the Amazon Mechanical Turk platform, and collect the labels from the participating users. The number of the objects in this dataset is 2,134. Totally, there are 64 normal users and they provide 17,729 labels.

Product Dataset [101, 118]. Each object in this dataset contains two products (with descriptions), the task is to judge whether the two products are the same or not. The participating users need to identify whether the two descriptions describe the same product or not, and then provide their labels. In this dataset, there are 8,315 objects which are observed by 176 normal users. Totally, these participating users provide 24,945 labels.

Sentiment Dataset [118]. Each object in the dataset is a tweet related to a company. The participating users need to identify whether the tweet has positive sentiment or not

to the company. The authors in [118] create 1,000 objects and collect labels from 85 normal users through the AMT platform. Totally, there are 20,000 labels in this dataset.

Baseline Methods

We compare the proposed attack mechanism with two baseline methods: *Baseline_rand* and *Baseline_inversion*.

In the *Baseline_rand* method, the attacker does not consider any strategy, and he just randomly sets the labels of each malicious user on a given object. This method introduce less overhead to the attacker, as he does not need to take effort to obtain and analyze the crowd-contributed data collected from the normal users.

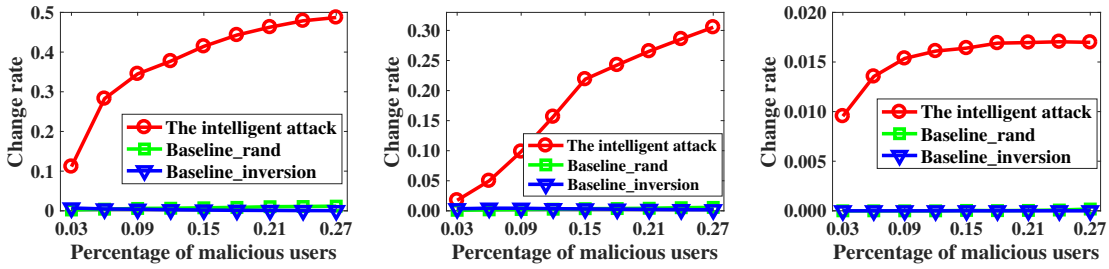
In the *Baseline_inversion* method, the attacker first conducts the Dawid-Skene model on the labels provided by the normal users and get the estimated true label for each object. Then he sets each malicious user's label on a given object as the candidate answer which is different from the estimated true label. This method is an intuitive attack strategy, in which the attacker tries to maximize the number of bad labels injected into the crowd-contributed data.

Performance Measure

In order to evaluate the performance of the proposed attack mechanism, we compare the aggregation results before and after the data poisoning attacks, and adopt the *change rate* as the measure metric. The *change rate* is defined as $\frac{||X^{*a}-X^{*b}||}{M}$, where $X^{*a} = \{x_m^{*a}\}_{m=1}^{M}$ and $X^{*b} = \{x_m^{*b}\}_{m=1}^{M}$ are the estimations for the objects' true labels after and before the data poisoning attacks. Since the goal of the attacker is to maximize the error of the aggregation results and meanwhile maximally raise the reliability degrees of the malicious users, thus, the larger the *change rate*, the better the method.

5.6.2 The Effect of the Percentage of the Malicious Users

When conducting the data poisoning attack, we assume that the attacker cannot manipulate the labels of normal users, but he can create or recruit multiple malicious users.



(a) (b) (c)

Figure 5.2: Change rate w.r.t. the percentage of the malicious users. (a): Duchenne Smile Dataset. (b): Product Dataset. (c): Sentiment Dataset.

Thus, the number of the malicious users created or recruited by the attacker plays an important role in the attack. If the attacker is able to create or recruit overwhelming number of malicious users, the goal of the attacker can be easily achieved with the intuitive attack strategy, i.e., the *Baseline_inversion* method. However, in practice, the attacker can only create or recruit a limited number of malicious users due to the limitation of his ability. In this experiment, we consider the scenarios where the percentage of malicious users is low, and evaluate the performance of the proposed mechanism when the percentage is varying.

Suppose N is the number of labels provided by the normal users for all objects. Here we assume that each malicious user can observe N/K objects, which is the average number of the objects observed by each normal user. For each malicious user, the N/K observed objects are randomly selected. In this chapter, we set the parameters θ and λ as 100 and 1, respectively. Then we vary the percentage of the malicious users from 0.03 to 0.27. All the experiments are conducted 50 times and we report the average results. The *change rate* for the three real-world crowd-contributed datasets is shown in Figure 5.2, in which we represent the proposed mechanism as *The intelligent attack*. From this figure, we can see the proposed attack mechanism performs better than the baseline methods in all cases. When the percentage of the malicious users is very low (e.g., 3%), since the malicious users are too few to change the final aggregation results much, the advantage of the proposed mechanism is small. However, when the percentage of the

malicious users increases, the advantage of the proposed attack scheme becomes bigger. For example, when the percentage of malicious users is 27%, the proposed mechanism successfully attacks nearly 50% of the objects in the Duchenne Smile datasets while the baseline methods only obtain marginal utility.

5.6.3 The Effect of the Number of the Observed Objects

When the percentage of the malicious users is given, the number of the objects queried to each malicious user is another important factor in the attack. In this experiment, we study the performance of the proposed mechanism when the number of the objects observed by each malicious user varies.

Here we consider a scenario where the percentage of the malicious users is very low and we set the value as 3%, i.e., the attacker creates or recruits 2, 6 and 3 malicious users to the three datasets, respectively. For the Duchenne Smile dataset and the Product dataset, we vary the number of objects observed by each malicious user from 50 to 500, and for the Sentiment dataset, the number of the queried objects varies from 100 to 700. The *change rate* for the three datasets is shown in Figure 5.3. The results in this figure clearly verify that the proposed attack mechanism outperforms the baseline methods in all cases. When the number of the objects observed by each malicious user increases, the advantage of the proposed attack mechanism also increases. The reason is that with the increment of the number of the observed objects, the malicious users can exert more impact on the final aggregation results based on the proposed mechanism. Additionally, this figure also shows that the proposed mechanism can achieve good utility even with very few malicious users. Take the Duchenne Smile dataset as an example, when each malicious user provides 250 labels (less than the average number of that from normal users), the proposed mechanism can successfully attack more than 10% of the objects with only 2 malicious users.

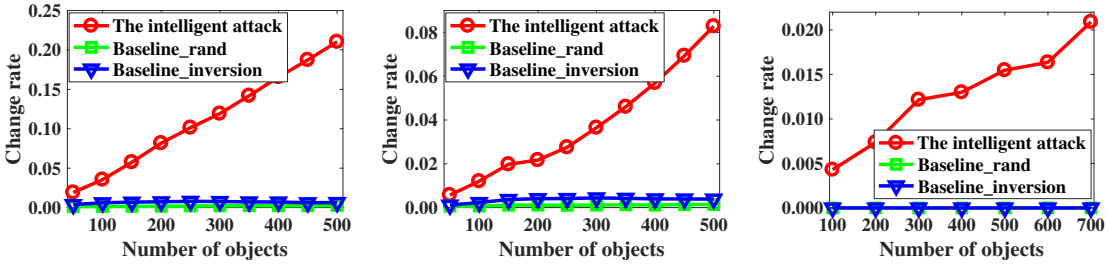


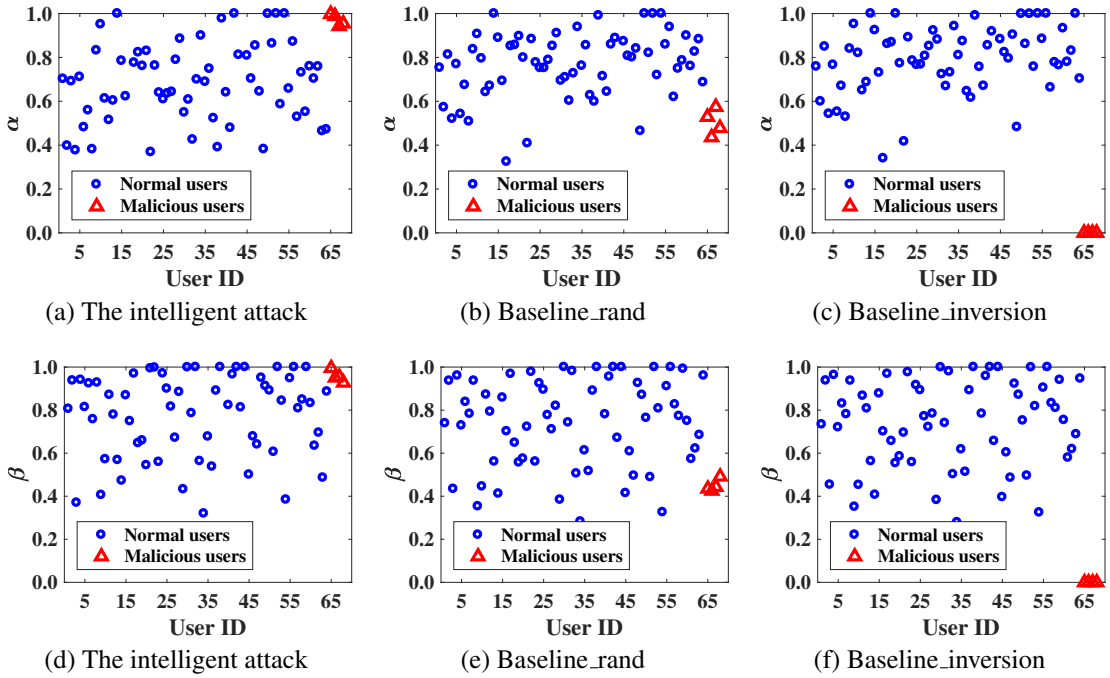
Figure 5.3: Change rate w.r.t. the number of the objects observed by each malicious user. (a): Duchenne Smile Dataset. (b): Product Dataset. (c): Sentiment Dataset.

5.6.4 Comparison on the Ability Parameters of the Malicious Users

Besides maximizing the error of the aggregation results, the attacker also tries to maximize the malicious users' reliability degrees (or ability) such that they can be treated as high-quality users and thus be disguised well. In fact, the proposed mechanism outperforms the baseline methods mainly because we take the effect of the malicious users' reliability degrees into account. The malicious users can disguise themselves as good users on some objects to enhance their reliability degrees. For the baseline methods, since the malicious users always disagree with the normal ones or randomly provide their labels, the attack behaviors may be detected by the Dawid-Skene model and the malicious users will be assigned with low reliability degrees.

In this experiment, we investigate the distribution of the participating users' ability parameters, i.e., $\alpha = \{\alpha_k, \widetilde{\alpha}_{k'}\}_{k,k'=1}^{K,K'}$ and $\beta = \{\beta_k, \widetilde{\beta}_{k'}\}_{k,k'=1}^{K,K'}$, which can be treated as the reliability degrees of these users based on the Dawid-Skene model. For each dataset, the percentage of the malicious users is fixed as 5%. We report the results of the parameters α and β for the three datasets after the data poisoning attacks in Figure 5.4, Figure 5.5 and Figure 5.6, respectively. The results show that the malicious users from the proposed mechanism have high reliability degrees (both α and β) comparing with the normal users. This means that the malicious users blend into the normal users successfully and they will be treated as high-quality users according to the Dawid-Skene model. This also verifies that the proposed mechanism can well disguise the malicious behaviors of

the attacker while maximizing the error of the aggregated results. In contrast, in the *Baseline_inversion* method, since the malicious users always disagree with the normal users, they will be assigned significantly low reliability degrees, which not only limit the performance of the malicious users, but also make them easy to be detected. As for the *Baseline_rand* method, since the malicious users randomly select their labels, the values of the ability parameters will be around 0.5. Although the malicious users from the *Baseline_rand* method can disguise themselves to some extent, their reliability degrees are not large enough to impact the aggregated results.



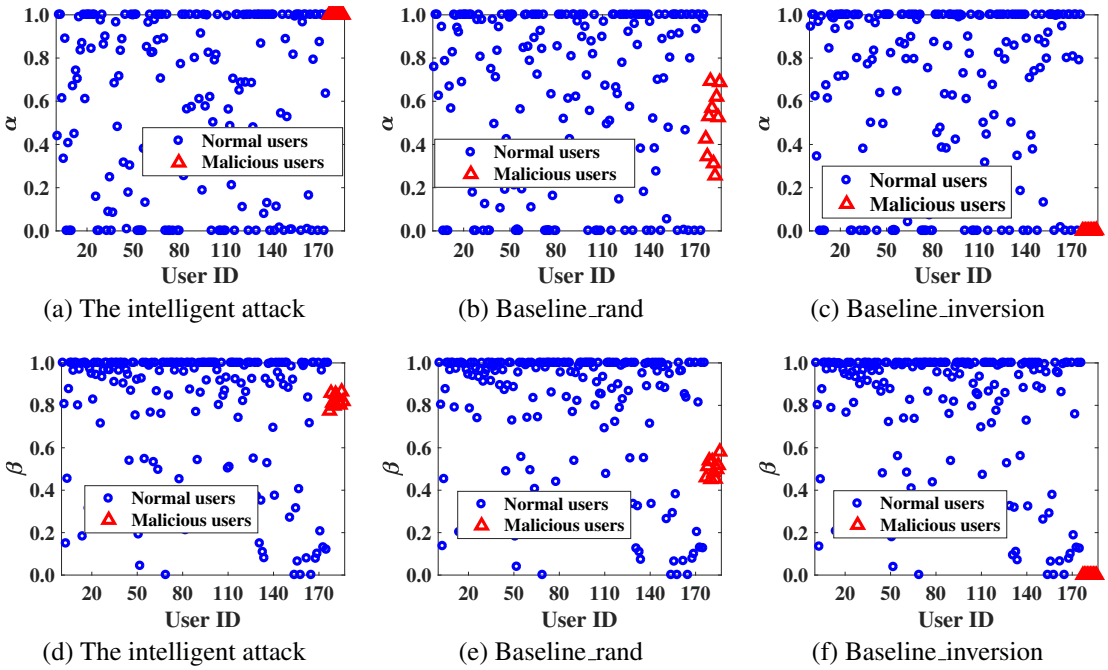
(a) The intelligent attack (b) Baseline_rand (c) Baseline_inversion

(d) The intelligent attack (e) Baseline_rand (f) Baseline_inversion

Figure 5.4: The ability parameters of the normal and malicious users for the Duchenne Smile dataset

5.6.5 The Effect of the Attacker's Knowledge

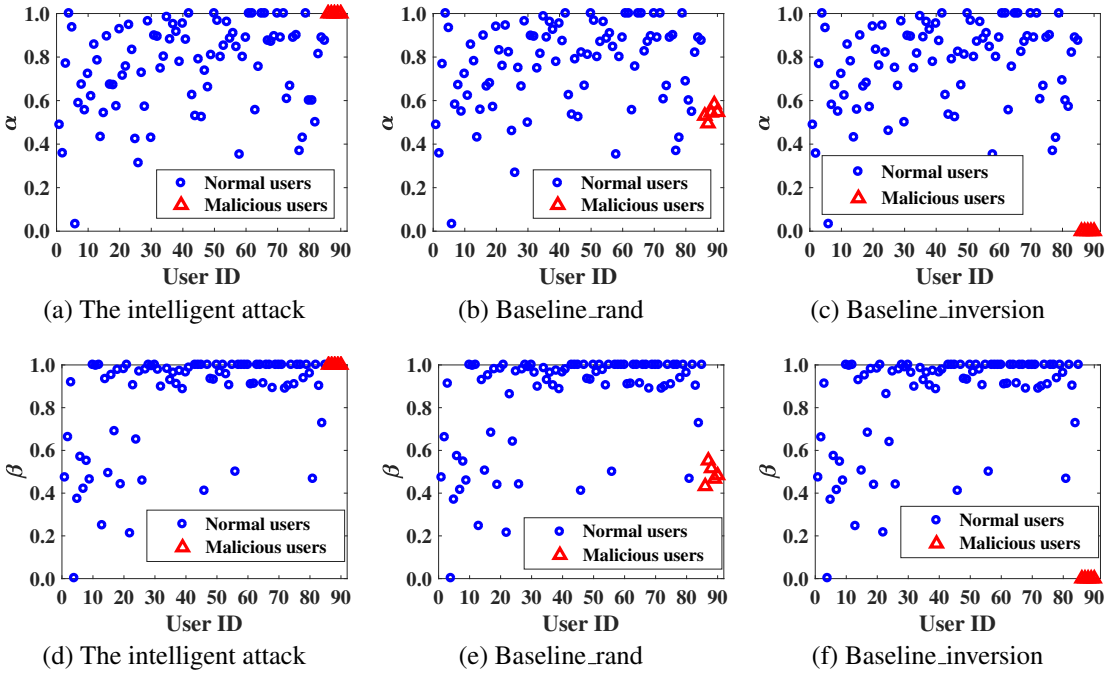
As described in Section 5.5, the proposed mechanism can also be employed when the attacker only has limited knowledge of the objects' normal labels. In this experiment, we evaluate the performance of the proposed mechanism with respect to the value of



(a) The intelligent attack (b) Baseline_rand (c) Baseline_inversion

(d) The intelligent attack (e) Baseline_rand (f) Baseline_inversion

Figure 5.5: The ability parameters of the normal and malicious users for the Product dataset



(a) The intelligent attack (b) Baseline_rand (c) Baseline_inversion

(d) The intelligent attack (e) Baseline_rand (f) Baseline_inversion

Figure 5.6: The ability parameters of the normal and malicious users for the Sentiment dataset

M'/M, i.e., the percentage of the objects whose labels from the normal users can be known by the attacker. Here we still consider a scenario where the percentage of the malicious users is very low (3%). We also assume that each malicious user can observe N/K objects, which are randomly selected from O'. Then we vary the value of M'/M from 0.3 to 1 and calculate the *change rate* for the three real-world datasets. We conduct the experiment for 50 times and report the average results in Figure 5.7, from which we can see the proposed mechanism outperforms the baseline methods in all cases, and the advantage of the proposed mechanism becomes bigger when the attacker's knowledge increases. These results verify that the proposed mechanism can still achieve good utility when the attacker only has limited knowledge of the objects' normal labels.

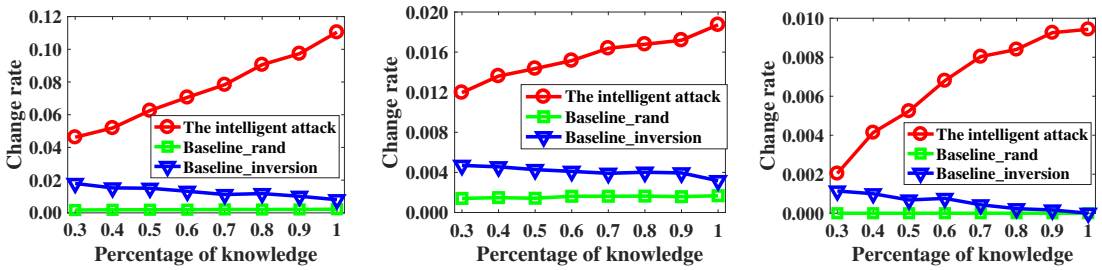
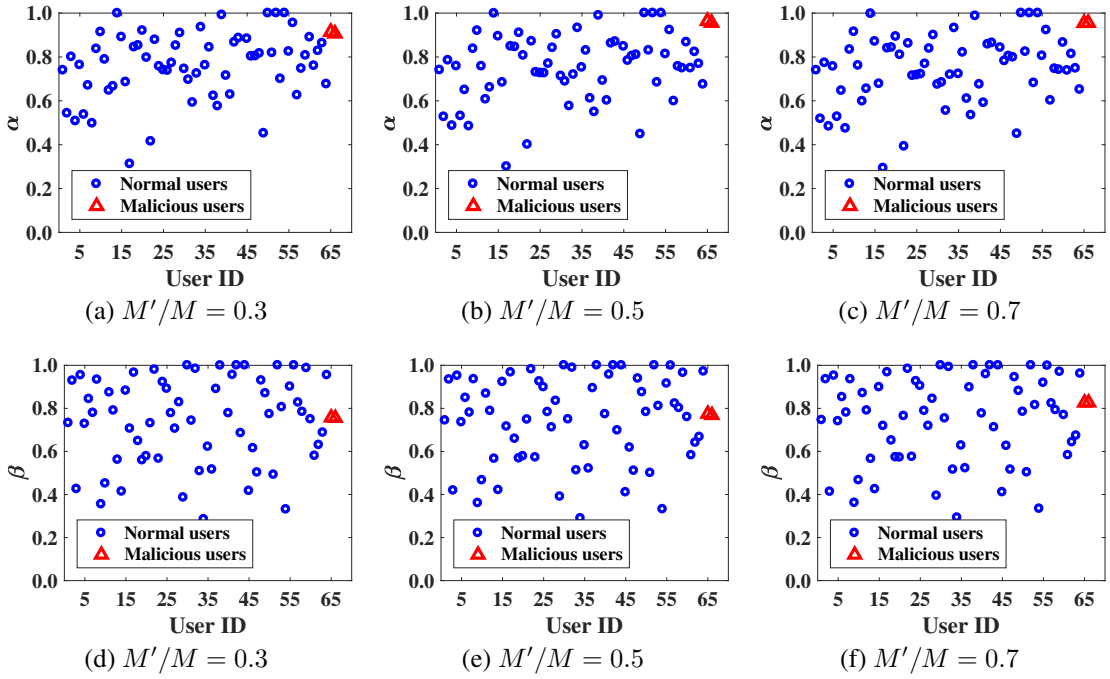


Figure 5.7: Change rate w.r.t. the percentage of the knowledge known by the attacker. (a): Duchenne Smile Dataset. (b): Product Dataset. (c): Sentiment Dataset.

To further evaluate the performance of the proposed mechanism in the limited knowledge scenarios, we investigate the distribution of the users' ability parameters when the attacker only has partial knowledge of the normal labels. Here we consider three cases in which the percentage of the known objects (i.e., M'/M) is set as 0.3, 0.5 and 0.7, respectively. In Figure 5.8 we report the results of the parameters α and β derived from the proposed mechanism on the the Duchenne Smile Dataset. The results show that the malicious users keep the high reliability degrees, which means that the proposed mechanism can well disguise the attack behaviors in the limited knowledge scenarios. As for the baseline methods, the results of them on the Duchenne Smile dataset are similar to those in Figure 5.4.



(a) $M'/M = 0.3$ (b) $M'/M = 0.5$ (c) $M'/M = 0.7$

(d) $M'/M = 0.3$ (e) $M'/M = 0.5$ (f) $M'/M = 0.7$

Figure 5.8: The users' ability parameters calculated by the intelligent attack mechanism in the limited knowledge scenarios for the Duchenne Smile Dataset

5.7 Summary

In this chapter, we study the data poisoning attacks against the crowd sensing systems with the Dawid-Skene model empowered. Specifically, we design an intelligent attack mechanism, based on which the attacker can not only achieve maximum attack utility but also intelligently disguise the introduced malicious users as normal ones or even good ones. The experimental results based on real-world datasets demonstrate that the proposed attack mechanism can achieve higher attack utility with very few malicious users and at the same time, is harder to be detected by the defense mechanisms.

Chapter 6

Related Work

In this chapter, we provide a brief overview of the literature in related fields. Specifically, we first introduce the literature related to privacy-preserving truth discovery. Then we discuss existing work regarding the data poisoning attacks.

6.1 Privacy-Preserving Truth Discovery

As an effective technique to extract reliable information from crowd sensing systems, truth discovery has drawn more and more attention [48, 60–65, 65, 70, 72, 94, 98, 102, 103, 111, 112] in recent years. 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 [31].

The importance of privacy protection has long been recognized in many fields [18, 42, 78]. The representative strategies to tackle various privacy concerns include 1) anonymization [18, 83, 95], which removes identification information from all the interactions between the participant and other entities, 2) data perturbation [52, 53], 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 [35, 50], 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, [43, 54, 91, 92] 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 [30, 82, 85, 97, 114]. 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 [27]. Recently, the fully homomorphic encryption scheme [34] 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 schemes are 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. In addition, paper [51] proposes a homomorphic encryption based approach to protect user privacy in crowdsourcing applications. However, it addresses a different scenario and mainly focuses on categorical data. In contrast, our schemes can deal with not only categorical data but also other data types. Finally, Catalano et al. propose a two-server based protocol [8] for the delegation of computation

on encrypted data. The frameworks presented in Chapter 3, though also involving two cloud servers, are designed for different problem settings and application scenarios.

6.2 Data Poisoning Attacks in Crowd Sensing Systems

The data poisoning attacks, also known as false data injection attacks, have recently been widely studied in crowd sensing and crowdsourcing applications [9, 25, 26, 28, 45, 58, 84, 88, 99, 100, 104, 113, 116]. The data poisoning attacks and related defense schemes are also studied in the applications other than crowd sensing and crowdsourcing, such as electric power grids [69] and network coding [56]. Besides, there also has been prior research exploring the data poisoning attacks on machine learning algorithms [1, 4, 6, 44, 57, 71, 106]. However, these previous works do not investigate how to effectively attack the crowd sensing systems empowered with truth discovery mechanism or the the Dawid-Skene model [22], which could tolerate the malicious users to some degree and are hard to be attacked.

Although different variants have been developed and the theoretical analysis has been conducted for the truth discovery mechanism [60, 61, 63, 65, 94, 102, 103, 110] and the Dawid-Skene model [13, 20, 59, 67, 79, 89, 93, 117, 119], these works do not take into consideration the sophisticated data poisoning attacks. In this thesis, our proposed data poisoning attack mechanisms can effectively maximize the utility of the attacker and disguise the the malicious behaviours. Thus, the above methods cannot defend against our designed attacks effectively. The most relevant papers to our work are [45, 46], in which the proposed schemes can identify the malicious users who conduct the sophisticated data poisoning attacks. However, based on these schemes, the users who agree with the majority will be classified as normal ones. Since the malicious users in our proposed mechanisms can disguise themselves by agreeing with the majority on some items, these methods will fail to detect them.

Chapter 7

Conclusions

The pervasive sensing devices in the era of Internet of Things have 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. Although crowd sensing can serve a wide spectrum of applications having significant impact on our daily lives, the privacy and security issues in these applications have largely degraded its effectiveness in practice. On one hand, the server that collects the sensory data may want to infer a user's sensitive personal information from the collected data. On the other hand, the participating users may launch malicious attacks and submit malicious sensory data. Therefore, there is a great need for privacy-preserving and security mechanisms to protect user privacy as well as defend against malicious attacks.

Towards the objective of enabling privacy-preserving and secure crowd sensing in the Internet of Things, in this thesis, we first propose a series of privacy-preserving truth discovery frameworks for crowd sensing systems. Then, we study the data poisoning attacks against the crowd sensing systems empowered with the truth discovery mechanism and the Dawid-Skene model.

- **Privacy-Preserving Truth Discovery for Crowd Sensing Systems.** In order to address the participating users' privacy concerns while identifying truthful values from the crowd sensing data, we first propose a novel privacy-preserving truth

discovery (PPTD) framework, which can protect not only users' sensory data but also their reliability scores derived by the truth discovery approaches. The key idea of the proposed framework is to perform weighted aggregation on users' encrypted data using a homomorphic cryptosystem. To deal with large-scale data, we also propose to parallelize PPTD with MapReduce framework. Additionally, we design an incremental PPTD scheme for the scenarios where the sensory data are collected in a streaming manner. Although the proposed PPTD framework can achieve strong privacy guarantee, however, at a cost of significant computation and communication overhead. To address this challenge, we then propose two lightweight privacy-preserving truth discovery frameworks (i.e., L-PPTD and L^2-PPTD), which are implemented by involving two non-colluding cloud platforms and adopting additively homomorphic cryptosystem. These two frameworks not only achieve the protection of each user's private information but also introduce little overhead to the users.

- **Data Poisoning Attacks in Crowd Sensing Systems.** Besides the privacy aspect, we also investigate the security aspect of the crowd sensing systems. Specifically, we study two types of data poisoning attacks, i.e., the availability attack and the target attack, against a crowd sensing system empowered with the truth discovery mechanism. We first analyze the pitfalls when attacking such a crowd sensing system and then design an optimal attack framework to derive the (approximately) optimal attack strategy. Through manipulating the malicious users' sensory data based on the derived attack strategy, the attacker can not only maximize his attack utility but also successfully disguise the attack behaviors. In addition, we study the security vulnerability of the Dawid-Skene model to data poisoning attacks in crowd sensing systems. Following a similar attacking philosophy for the truth discovery mechanism, we design an intelligent data poisoning attack framework that can effectively take down a crowd sensing system empowered with the Dawid-Skene model.

Bibliography

[1] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*, pages 1452–1458, 2016.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[3] Jonathan F Bard. *Practical bilevel optimization: algorithms and applications*. Kluwer Academic Publishers, 1998.

[4] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security (ASIACCS'06)*, pages 16–25, 2006.

[5] Dimitri P Bertsekas. *Nonlinear programming*. 1999.

[6] Battista Biggio, B Nelson, and P Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*, pages 1807–1814, 2012.

[7] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. In *Foundations and Trends® in Machine Learning*, volume 3, pages 1–122, 2011.

[8] Dario Catalano and Dario Fiore. Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS'15)*, pages 1518–1529, 2015.

[9] Shih-Hao Chang and Zhi-Rong Chen. Protecting mobile crowd sensing against sybil attacks using cloud based trust management system. *Mobile Information Systems*, 2016, 2016.

[10] Si Chen, Muyuan Li, Kui Ren, Xinwen Fu, and Chunming 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)*, pages 59–71, 2015.

[11] Si Chen, Muyuan Li, Kui Ren, Xinwen Fu, and Chunming 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)*, pages 59–71, 2015.

[12] Si Chen, Muyuan Li, Kui Ren, and Chunming 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)*, pages 1–10, 2015.

[13] Xi Chen, Qihang Lin, and Dengyong Zhou. Optimistic knowledge gradient policy for optimal budget allocation in crowdsourcing. In *Proceedings of the the 30th International Conference on Machine Learning (ICML'13)*, pages 64–72, 2013.

[14] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, Yilong Li, Ji Jia, and Xiaofan 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)*, pages 251–265, 2014.

[15] Yohan Chon, Yunjong Kim, and Hojung 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)*, pages 19–30, 2013.

[16] Yohan Chon, Nicholas D Lane, Yunjong Kim, Feng Zhao, and Hojung 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)*, pages 3–12, 2013.

[17] Yohan Chon, Nicholas D Lane, Fan Li, Hojung Cha, and Feng Zhao. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp'12)*, pages 481–490, 2012.

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

[19] Ronald Cramer, Ivan Damgård, and Jesper B Nielsen. *Multiparty computation from threshold homomorphic encryption*. 2001.

[20] Nilesh Dalvi, Anirban Dasgupta, Ravi Kumar, and Vibhor Rastogi. Aggregating crowdsourced binary ratings. In *Proceedings of the 22nd International Conference on World Wide Web (WWW'13)*, pages 285–294, 2013.

[21] Ivan Damgård and Mads Jurik. A generalisation, a simpli. cation and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography*, pages 119–136, 2001.

[22] Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979.

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

[24] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society*, pages 1–38, 1977.

[25] Akshay Dua, Nirupama Bulusu, Wu-Chang Feng, and Wen Hu. Combating software and sybil attacks to data integrity in crowd-sourced embedded systems. *ACM Transactions on Embedded Computing Systems*, 13(5s):1–19, 2014.

[26] Lingjie Duan, Alexander W Min, Jianwei Huang, and Kang G Shin. Attack prevention for collaborative spectrum sensing in cognitive radio networks. *IEEE Journal on Selected Areas in Communications*, 30(9):1658–1665, 2012.

[27] Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007:15, 2007.

[28] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. Understanding malicious behavior in crowdsourcing platforms: The case of online surveys. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI'15)*, pages 1631–1640, 2015.

[29] Raghu K Ganti, Nam Pham, Hossein Ahmadi, Saurabh Nangia, and Tarek 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)*, pages 151–164, 2010.

[30] Raghu K Ganti, Nam Pham, Yu-En Tsai, and Tarek F Abdelzaher. Poolview: stream privacy for grassroots participatory sensing. In *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys'08)*, pages 281–294, 2008.

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

[32] Lin Gao, Fen Hou, and Jianwei Huang. Providing long-term participation incentive in participatory sensing. In *Proceedings of the 34th IEEE Conference on Computer Communications (INFOCOM'15)*, pages 2803–2811, 2015.

[33] Ruipeng Gao, Mingmin Zhao, Tao Ye, Fan Ye, Yizhou Wang, Kaigui Bian, Tao Wang, and Xiaoming 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)*, pages 249–260, 2014.

[34] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, 2009.

[35] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

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

[37] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[38] Shaohan Hu, Hengchang Liu, Lu Su, Hongyan Wang, Tarek F Abdelzaher, Pan Hui, Wei Zheng, Zhiheng Xie, John 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)*, pages 1752–1760, 2014.

[39] Shaohan Hu, Lu Su, Shen Li, Shiguang Wang, Chenji Pan, Siyu Gu, T Amin, Hengchang Liu, Suman Nath, Romit Roy Choudhury, et al. Experiences with enav: A low-power vehicular navigation system. In *Proceedings of the 2015 ACM Conference on Ubiquitous Computing (UbiComp'15)*, pages 433–444, 2015.

[40] Shaohan Hu, Lu Su, Hengchang Liu, Hongyan Wang, and Tarek 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)*, pages 331–332, 2013.

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

[42] Kuan Lun Huang, Salil S Kanhere, and Wen Hu. Towards privacy-sensitive participatory sensing. In *Proceedings of the 7th IEEE International Conference on Pervasive Computing and Communications (PerCom'09)*, pages 1–6, 2009.

[43] Kuan Lun Huang, Salil S Kanhere, and Wen Hu. A privacy-preserving reputation system for participatory sensing. In *Proceedings of the 37th Annual IEEE Conference on Local Computer Networks (LCN'12)*, pages 10–18, 2012.

[44] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence (AISec'11)*, pages 43–58, 2011.

[45] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. Reputation-based worker filtering in crowdsourcing. In *Proceedings of the 28th Conference on Neural Information Processing Systems (NeurIPS'14)*, pages 2492–2500, 2014.

[46] Srikanth Jagabathula, Lakshminarayanan Subramanian, and Ashwin Venkataraman. Identifying unreliable and adversarial workers in crowdsourced labeling tasks. *The Journal of Machine Learning Research*, 18(1):3233–3299, 2017.

[47] Changkun Jiang, Lin Gao, Lingjie Duan, and Jianwei Huang. Scalable mobile crowdsensing via peer-to-peer data sharing. *IEEE Transactions on Mobile Computing*, 17(4):898–912, 2017.

[48] Wenjun Jiang, Chenglin Miao, Lu Su, Qi Li, Shaohan Hu, ShiGuang Wang, Jing Gao, Hengchang Liu, Tarek Abdelzaher, Jiawei Han, et al. Towards quality aware information integration in distributed sensing systems. *IEEE Transactions on Parallel and Distributed Systems*, 29(1):198–211, 2017.

[49] Haiming Jin, Baoxiang He, Lu Su, Klara Nahrstedt, and Xinbing Wang. Data-driven pricing for sensing effort elicitation in mobile crowd sensing systems. *IEEE/ACM Transactions on Networking*, 27(6):2208–2221, 2019.

[50] Taeho Jung, XuFei Mao, Xiang-Yang Li, Shao-Jie Tang, Wei Gong, and Lan 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)*, pages 2634–2642, 2013.

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

[52] Hillol Kargupta, Souptik Datta, Qi Wang, and Krishnamoorthy Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the third IEEE International Conference on Data Mining (ICDM'03)*, pages 99–106, 2003.

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

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

[55] Nicholas D Lane, Yohan Chon, Lin Zhou, Yongzhe Zhang, Fan Li, Dongwon Kim, Guanzhong Ding, Feng Zhao, and Hojung 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)*, pages 1–14, 2013.

[56] Anh Le and Athina Markopoulou. Cooperative defense against pollution attacks in network coding using spacemac. *IEEE Journal on Selected Areas in Communications*, 30(2):442–449, 2012.

[57] Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. Data poisoning attacks on factorization-based collaborative filtering. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NeurIPS'16)*, pages 1885–1893, 2016.

[58] Hongjuan Li, Xiuzhen Cheng, Keqiu Li, Chunqiang Hu, Nan Zhang, and Weilian Xue. Robust collaborative spectrum sensing schemes for cognitive radio networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(8):2190–2200, 2013.

[59] Hongwei Li, Bin Yu, and Dengyong Zhou. Error rate analysis of labeling by crowdsourcing. In *ICML Workshop: Machine Learning Meets Crowdsourcing*, 2013.

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

[61] Qi Li, Yaliang Li, Jing Gao, Bo Zhao, Wei Fan, and Jiawei 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)*, pages 1187–1198, 2014.

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

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

[64] Yaliang Li, Qi Li, Jing Gao, Lu Su, Bo Zhao, Wei Fan, and Jiawei Han. On the discovery of evolving truth. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'15)*, pages 675–684, 2015.

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

[66] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Proceedings of the 20th Annual International Cryptology Conference (CRYPTO'00)*, pages 36–54, 2000.

[67] Qiang Liu, Jian Peng, and Alexander T Ihler. Variational inference for crowdsourcing. In *Proceedings of the 26th Conference on Neural Information Processing Systems (NeurIPS'12)*, pages 692–700, 2012.

[68] Shengzhong Liu, Zhenzhe Zheng, Fan Wu, Shaojie Tang, and Guihai Chen. Context-aware data quality estimation in mobile crowdsensing. In *Proceedings of the 36th IEEE Conference on Computer Communications (INFOCOM'17)*, pages 1–9, 2017.

[69] Yao Liu, Peng Ning, and Michael K Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security*, 14(1):1–33, 2011.

[70] Fenglong Ma, Yaliang Li, Qi Li, Minghui Qiu, Jing Gao, Shi Zhi, Lu Su, Bo Zhao, Heng Ji, and Jiawei 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)*, pages 745–754, 2015.

[71] Shike Mei and Xiaojin Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 2871–2877, 2015.

[72] Chuishi Meng, Wenjun Jiang, Yaliang Li, Jing Gao, Lu Su, Hu Ding, and Yun Cheng. Truth discovery on crowd sensing of correlated entities. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15)*, pages 169–182, 2015.

[73] Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin, Houping Xiao, Jing Gao, and Kui Ren. Cloud-enabled privacy-preserving truth discovery in crowd sensing systems. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys'15)*, pages 183–196, 2015.

[74] Chenglin Miao, Wenjun Jiang, Lu Su, Yaliang Li, Suxin Guo, Zhan Qin, Houping Xiao, Jing Gao, and Kui Ren. Privacy-preserving truth discovery in crowd sensing systems. *ACM Transactions on Sensor Networks*, 15(1):1–32, 2019.

[75] Chenglin Miao, Qi Li, Lu Su, Mengdi Huai, Wenjun Jiang, and Jing Gao. Attack under disguise: An intelligent data poisoning attack mechanism in crowdsourcing. In *Proceedings of the 2018 World Wide Web Conference (WWW'18)*, pages 13–22, 2018.

[76] Chenglin Miao, Qi Li, Houping Xiao, Wenjun Jiang, Mengdi Huai, and Lu Su. Towards data poisoning attacks in crowd sensing systems. In *Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'18)*, pages 111–120, 2018.

[77] Chenglin Miao, Lu Su, Wenjun Jiang, Yaliang Li, and Miaomiao Tian. A lightweight privacy-preserving truth discovery framework for mobile crowd sensing systems. In *Proceedings of the 36th Annual IEEE International Conference on Computer Communications (INFOCOM'17)*, pages 1–9, 2017.

[78] Sangho Oh, Tam Vu, Marco Gruteser, and Suman Banerjee. Phantom: Physical layer cooperation for location privacy protection. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM'12)*, pages 3061–3065, 2012.

[79] Jungseul Ok, Sewoong Oh, Jinwoo Shin, and Yung Yi. Optimality of belief propagation for crowdsourced classification. In *Proceedings of the the 33rd International Conference on Machine Learning (ICML'16)*, pages 535–544, 2016.

[80] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the 17th international conference on Theory and application of cryptographic techniques (EUROCRYPT'99)*, pages 223–238, 1999.

[81] Dan Peng, Fan Wu, and Guihai Chen. Data quality guided incentive mechanism design for crowdsensing. *IEEE transactions on mobile computing*, 17(2):307–319, 2017.

[82] Nam Pham, Raghu K Ganti, Yusuf S Uddin, Suman Nath, and Tarek Abdelzaher. Privacy-preserving reconstruction of multidimensional data maps in vehicular participatory sensing. In *Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN'10)*, pages 114–130. 2010.

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

[84] Zhengrui Qin, Qun Li, and George Hsieh. Defending against cooperative attacks in cooperative spectrum sensing. *IEEE Transactions on Wireless Communications*, 12(6):2680–2687, 2013.

[85] Daniele Quercia, Ilias Leontiadis, Liam McNamara, Cecilia Mascolo, and Jon Crowcroft. Spotme if you can: Randomized responses for location obfuscation on mobile phones. In *Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS'11)*, pages 363–372, 2011.

[86] Moo-Ryong Ra, Bin Liu, Tom F La Porta, and Ramesh 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)*, pages 337–350, 2012.

[87] Kiran K Rachuri, Cecilia Mascolo, Mirco Musolesi, and Peter 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)*, pages 73–84, 2011.

[88] Ankit Singh Rawat, Priyank Anand, Hao Chen, and Pramod K Varshney. Collaborative spectrum sensing in the presence of byzantine attacks in cognitive radio networks. *IEEE Transactions on Signal Processing*, 59(2):774–786, 2010.

[89] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322, 2010.

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

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

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

[93] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*, pages 254–263, 2008.

[94] Lu Su, Qi Li, Shaohan Hu, Shiguang Wang, Jing Gao, Hengchang Liu, Tarek F Abdelzaher, Jiawei Han, Xue Liu, Yan 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)*, pages 1–10, 2014.

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

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

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

[98] Dong Wang, Lance Kaplan, Hieu Le, and Tarek 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)*, pages 233–244, 2012.

[99] Gang Wang, Bolun Wang, Tianyi Wang, Ana Nika, Haitao Zheng, and Ben Y Zhao. Defending against sybil devices in crowdsourced mapping services. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys'16)*, pages 179–191, 2016.

[100] Gang Wang, Tianyi Wang, Haitao Zheng, and Ben Y Zhao. Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security'14)*, pages 239–254, 2014.

[101] Jiannan Wang, Tim Kraska, Michael J Franklin, and Jianhua Feng. Crowder: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11):1483–1494, 2012.

[102] Shiguang Wang, Lu Su, Shen Li, Shaohan Hu, Tanvir Amin, Hongwei Wang, Shuochao Yao, Lance Kaplan, and Tarek Abdelzaher. Scalable social sensing of interdependent phenomena. In *Proceedings of the 14th ACM International Conference on Information Processing in Sensor Networks (IPSN'15)*, pages 202–213, 2015.

[103] Shiguang Wang, Dong Wang, Lu Su, Lance Kaplan, and Tarek 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)*, pages 74–85, 2014.

[104] Wei Wang, Lin Chen, Kang G Shin, and Lingjie Duan. Thwarting intelligent malicious behaviors in cooperative spectrum sensing. *IEEE Transactions on Mobile Computing*, 14(11):2392–2405, 2015.

[105] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Proceedings of the 23rd Conference on Neural Information Processing Systems (NeurIPS'09)*, pages 2035–2043, 2009.

[106] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, pages 1689–1698, 2015.

[107] Susu Xu, Xinlei Chen, Xidong Pi, Carlee Joe-Wong, Pei Zhang, and Hae Young Noh. ilocus: Incentivizing vehicle mobility to optimize sensing distribution in crowd sensing. *IEEE Transactions on Mobile Computing*, 2019.

[108] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *Proceedings of the 18th annual international conference on Mobile computing and networking (MobiCom'12)*, pages 173–184, 2012.

[109] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones. *IEEE/ACM transactions on networking*, 24(3):1732–1744, 2015.

[110] Shuo Yang, Fan Wu, Shaojie Tang, Xiaofeng Gao, Bo Yang, and Guihai Chen. On designing data quality-aware truth estimation and surplus sharing method for mobile crowdsensing. volume 35, pages 832–847, 2017.

[111] Shuochao Yao, Md Tanvir Amin, Lu Su, Shaohan Hu, Shen Li, Shiguang Wang, Yiran Zhao, Tarek Abdelzaher, Lance Kaplan, Charu Aggarwal, et al. Recursive ground truth estimator for social data streams. In *Proceedings of the 15th*

ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN'16), pages 1–12, 2016.

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

[113] Dong Yuan, Guoliang Li, Qi Li, and Yudian Zheng. Sybil defense in crowdsourcing platforms. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM'17)*, pages 1529–1538, 2017.

[114] Fan Zhang, Li He, Wenbo He, and Xue Liu. Data perturbation with state-dependent noise for participatory sensing. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM'12)*, pages 2246–2254, 2012.

[115] Mengyuan Zhang, Lei Yang, Xiaowen Gong, and Junshan Zhang. Privacy-preserving crowdsensing: Privacy valuation, network effect, and profit maximization. In *Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM'16)*, pages 1–6, 2016.

[116] Rui Zhang, Jinxue Zhang, Yanchao Zhang, and Chi Zhang. Secure crowdsourcing-based cooperative pectrum sensing. In *Proceedings of the 32nd IEEE Conference on Computer Communications (INFOCOM'13)*, pages 2526–2534, 2013.

[117] Yuchen Zhang, Xi Chen, Denny Zhou, and Michael I Jordan. Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Proceedings of the 28th Conference on Neural Information Processing Systems (NeurIPS'14)*, pages 1260–1268, 2014.

[118] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: is the problem solved? *Proceedings of the VLDB Endowment*, 10(5):541–552, 2017.

[119] Denny Zhou, Sumit Basu, Yi Mao, and John C Platt. Learning from the wisdom of crowds by minimax entropy. In *Proceedings of the 26th Conference on Neural Information Processing Systems (NeurIPS'12)*, pages 2195–2203, 2012.