博士論文

Privacy-Preserving Crowdsourcing

(プライバシ保護クラウドソーシング)

Hiroshi Kajino

(梶野 洸)

Abstract

Crowdsourcing is an idea in which requesters outsource tasks to unspecified workers via the Web. A basic procedure can be described using the following three steps. In the assignment step, a crowdsourcing platform matches tasks and workers, employing either a push-type or pull-type assignment strategy. The push-type assignment strategy uses the platform to assign tasks to appropriate workers based on the features of the workers and tasks (*e.g.*, skills, preferences of tasks, and minimum wages), while the pull-type assignment requires workers to choose tasks they like. In the request step, each requester sends a job instruction and task instances (*e.g.*, audio files in case of an audio transcription task) to the allocated workers. Finally, in the delivery step, each worker, having processed the assigned task, sends the results back to the requester. Crowdsourcing provides requesters with easy access to a huge pool of workers and enables workers to work much more flexibly than in the traditional labor market. These unique advantages have led to a number of real applications and businesses as well as new research opportunities such as human computation.

Despite its revolutionary power, it is often pointed out that using crowdsourcing entails several risks including the risk of poor quality task results. Among others, this thesis focuses on the privacy risks. Although the privacy risks in crowdsourcing have been pointed out in diverse domains, little has been investigated until now. Toward establishing a research basis for privacy-preserving crowdsourcing, this thesis addresses the following two research questions:

- *What types of privacy risks are present in crowdsourcing?*
- *How can we measure and control the privacy risks in crowdsourcing?*

To answer the first research question, we carefully examine the three steps of crowdsourcing and discover that four types of data can lead to privacy issues: features (the assignment step), job instruction and task instances (the request step), and task results (the delivery step). Further, by analyzing the applicability of existing privacy preservation strategies, we find that some

types of data cannot be handled by the existing approaches, highlighting the novelty of privacy-preserving crowdsourcing research.

Given this finding, we develop the following three solutions to answer the second research question.

(1) Privacy Preservation in the Assignment Step

We present a privacy-preserving task assignment (PTA) protocol, which computes an optimal task assignment, keeping the features of the workers and tasks private. Observing that our task assignment problem can be reduced to the maximum flow problem, the PTA protocol constructs an instance of the maximum flow problem and solves it by harnessing the push-relabel algorithm, both in a privacy-preserving way. Because the PTA protocol significantly decreases the number of workers who receive instructions and instances compared to the standard pull-type task assignment, our protocol also reduces the privacy risks associated with them. We evaluate the computation overhead induced by cryptography and discuss relaxation methods to reduce it.

(2) Privacy Preservation in the Request Step

We present the utility-privacy trade-off analyzer (UPTA), which enables us to evaluate the trade-off between the utility and privacy of an instance-privacy-preserving (IPP) protocol. Because an instance is used to perform a task as well as to extract the sensitive information contained within it, an IPP protocol in general has to sacrifice utility for privacy. Therefore, it is essential to quantify the trade-off in order to research instance-privacy preservation. The idea of UPTA is to model the task execution and privacy invasion as sampling of a task result and sensitive value from probability distributions. We estimate the models using crowdsourcing and apply divergence-based measures to the estimated models in order to quantify utility and privacy. As a case study of UPTA, we develop an instance-clipping (IC) protocol and analyze its properties. The IC protocol submits a task with clipped instances of a fixed size. We discuss the performance of the IC protocol as well as the validity of UPTA in the experiments.

(3) Privacy Preservation in the Delivery Step

We present a worker-private latent class (WPLC) protocol, which allows a requester to receive task results without compromising the privacy of the workers associated with the task results. The key observation is that a requester often aggregates results to produce quality-controlled results, which are no longer associated with any worker. The

WPLC protocol simulates the aggregation procedure using cryptography to output quality-controlled results without disclosing the results of each worker. We discuss the validity of WPLC by evaluating the disadvantages induced by cryptography, including its computation time.

Acknowledgments

I would like to express my great gratitude to both of my PhD advisers, Professor Hisashi Kashima and Professor Kenji Yamanishi. Professor Hisashi Kashima supported my doctoral research with perseverance and insightful instruction. His advice on research strategy has always inspired me to deliberate on my research topics. Professor Kenji Yamanishi took over the mentorship when Professor Kashima moved to Kyoto University. He has provided me with great support and the environment to accomplish my research.

I would like to thank all of my thesis committee members, including Professor Noboru Kunihiro, Professor Hiroshi Nakagawa, Professor Jun Sakuma, and both my PhD advisers. Their insightful and valuable comments have surely contributed to the improvement of this thesis.

I would like to thank all of the researchers who have collaborated with me on my doctoral research, including Professor Hiromi Arai, Professor Yukino Baba, and Professor Jun Sakuma. The fruitful comments from experts in human computation, crowdsourcing, privacy-preserving data mining, and cryptography have always enhanced the quality of this research. I would also like to thank my mentors at IBM Research Ireland including Dr. Akihiro Kishimoto, Dr. Adi Botea, Dr. Elizabeth Daly, and Dr. Spyros Kotoulas. Although the paper we wrote is not included in this thesis, our intense discussions and their patient advice regarding the paper were of great help when forming my research philosophy and improving my writing skills.

Finally, I would like to thank my family for their many years of support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Rise of Crowdsourcing

With the spread of the Internet, we have witnessed a variety of changes in our daily lives, including electric commerce, blogs, and social networks, to name a few. Online shopping enables anyone to be an owner of a shop and a customer to obtain a number of choices; blogs provide users with a means of transmitting their opinions and viewers with multiple perspectives. The Internet offers every person an opportunity to freely communicate with other people all over the world, which enables these innovations to happen.

The labor market is being faced thanks to such a drastic change with the rise of crowdsourcing. The term crowdsourcing was originally invented by Jeff Howe and Mark Robinson, and the idea became widely known because of Howe's article published in Wired magazine (Howe, 2006a), stating that several organizations were beginning to investigate the power of crowds. They outsourced part of their tasks requiring professional work to a large network of unspecified people at low cost. For example, a project director of a museum decided to use iStock,[1] the pioneer of microstock photography, instead of traditional stock agencies to reduce the royalty fee. Another example leveraged the diversity of crowds. InnoCentive[2] manages a crowd-based R&D platform in which firms post scientific problems and solvers from diverse disciplines attack them with the aim of incentive fees. Summarizing these examples, Howe defined crowdsourcing (Howe, 2006b) as,

> *"the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call."*

[1] `http://www.istockphoto.com`
[2] `http://www.innocentive.com`

The essential feature of crowdsourcing that distinguishes it from the traditional labor market is that requesters[3] obtain easy access to a huge workforce pool consisting of diverse individuals. A survey about the demographics of Amazon Mechanical Turk, a general-purposed crowdsourcing marketplace, revealed its diversity in age, education, nationality, and so on (Ipeirotis, 2010b,a).[4] The diverse and large workforce pool contributes to the reduction of both financial and time costs and helps facilitate tasks that are difficult to solve using a small group. For example, some workers participate in crowdsourcing in their spare time, and others live in a cheap country, which enables affordably priced crowdsourcing. Moreover, tasks can be processed in a highly parallel manner, and thus, a requester can acquire the result of a task much faster than previously possible.

1.2 Applications of Crowdsourcing

A number of crowdsourcing applications have emerged in a wide variety of domains to address problems previously unsolvable without it. We review five different classes of crowdsourcing to demonstrate the impact of crowdsourcing on diverse fields.

1.2.1 Microtask Marketplace

One of the most famous examples of crowdsourcing is microtask marketplaces such as Amazon Mechanical Turk[5] and CrowdFlower.[6] These marketplaces mainly deal in microtasks, which anyone without special skills can complete within a few minutes, *e.g.*, object classification, content generation, questionnaires, and transcription. A microtask marketplace enables us to request a number of tasks to a crowd quite easily at low cost. In some marketplaces, even a computer program can access enormous human resources through APIs. Workers are motivated by monetary rewards, and the prices paid in such marketplaces are relatively low; fewer than 15% of tasks reward USD 1 or more in Mechanical Turk (Ipeirotis, 2010a). As a result, we are now able to construct a large-scale dataset for computer vision (Deng et al., 2009) that could not be constructed without crowdsourcing. In addition, as a worker,

[3]We call employers who outsource tasks *requesters* and employees who process tasks *workers* in this thesis.

[4]The latest demographics are available at `http://www.mturk-tracker.com/`.

[5]`https://www.mturk.com/mturk/welcome`

[6]`http://www.crowdflower.com`

we can earn money within a small amount of spare time from anywhere on Earth, which drastically changes our working styles.

1.2.2 Macrotask Marketplace

Another variant of crowdsourcing marketplaces is a macrotask marketplace or a freelance marketplace such as Lancers[7] and Upwork,[8] which was previously known as oDesk. They mainly deal in macrotasks, which require some expertise for completion, *e.g.*, programming, translation, design, and writing. Workers in this marketplace are motivated by much higher monetary rewards than those in the microtask marketplace because of its specialty. Macrotask marketplaces provide a direct channel for general tasks between requesters and workers. Workers can easily work as individual freelancers, and requesters are given an easy way to find appropriate workers who can perform their specialized tasks. As a result, requesters can outsource even a professional task faster and more cheaply than in the traditional labor market. Workers are able to start and grow their own business in a macrotask marketplace by gaining credit with requesters and proving their skills with official tests provided by the marketplaces. For example, Upwork provides skill tests for workers to certify their profession. Such achievements increase the chances that a skilled worker will receive well-paid orders.

1.2.3 Personal Crowdsourcing

Aside from the previous two types of crowdsourcing, most of the other crowdsourcing applications manage their own platforms for their particular tasks. ESP Game (von Ahn and Dabbish, 2004) collects annotations on images, reCAPTCHA (von Ahn et al., 2008) asks workers to transcribe scanned documents, Foldit (Cooper et al., 2010) makes use of workers' intuition to derive the structure of proteins, and Wikipedia[9] constructs a free-access online encyclopedia. Personal crowdsourcing often devises a task design to motivate workers to perform tasks without monetary reward. One approach to motivating workers is gamification, *i.e.*, task processing is converted into the form of a computer game, and workers take part in crowdsourcing for fun. As a result, requesters can process a number of tasks in a short time without any cost if it succeeds.

[7] `http://www.lancers.co.jp`
[8] `https://www.upwork.com`
[9] `https://www.wikipedia.org`

1.2.4 Mobile Crowdsourcing

Mobile crowdsourcing is different from the others in that it makes use of workers' mobile devices such as smartphones for task processing. A typical example is crowdsensing (Ganti et al., 2011), which aims to collect sensing data from mobile device users. Ganti et al. (2011) classify applications of crowdsensing into three categories: environmental, infrastructural, and social applications. An environmental deployment of crowdsensing uses a crowd to monitor aspects of the environment such as air pollution or water level. For example, Kim et al. (2011) presented a system called Creek Watch, which asks workers to collect information about waterways for monitoring, including the GPS location, a photo, and observations of water level, flow rate, and trash in the water. An infrastructural application monitors public infrastructure. Shah et al. (2011) developed a system in which workers can report criminal incidents along with location information, and users can search for a safe route by specifying the origin and destination. Schnitzler et al. (2014) utilized crowdsourcing to check the traffic status of places where sensors return inconsistent results. A social application, which is somewhat different from typical crowdsourcing, sets up a platform to share individuals' data, *e.g.*, exercise data or diet data, with other participants to motivate themselves. In summary, a crowdsensing platform aggregates data usually belonging to individuals to achieve large-scale sensing at low cost, which is difficult to achieve without the idea of crowdsourcing. These crowdsourcing services mainly rely on the voluntary contributions of workers who are concerned about the sensing results.

Another example of mobile crowdsourcing is errand crowdsourcing such as TaskRabbit.[10] TaskRabbit manages a marketplace for errands such as cleaning, shopping, delivery, moving help, and handyman tasks. Because the tasks are closely tied with their locations, a task recommendation function is often implemented.

1.2.5 Citizen Science

Citizen science is the application of crowdsourcing to scientific research. It involves nonprofessional individuals in scientific procedures. Most of them are volunteer-based projects, and workers are often motivated by their desire to contribute to science. With citizen science, scientists are able to analyze a large quantity of data that have been untouched because of a workforce shortage.

[10] `https://taskrabbit.com`

An early successful example of citizen science is Foldit (Cooper et al., 2010), which aims to fold proteins into a better structure with the help of the crowd. Foldit converts the process of folding proteins into a puzzle game in order to motivate workers. Top-ranked players in Foldit have been shown to outperform the performance of existing computer algorithms, which demonstrates the massive power of crowds. GalaxyZoo (Lintott et al., 2008) is another successful citizen science project. It asks workers to classify huge numbers of images of galaxies, which astronomers by themselves cannot afford to analyze. It has achieved such success that eight million classifications were made in ten days (Clery, 2011). A survey about the motivations of citizen scientists in GalaxyZoo (Jordan Raddick et al., 2013) reveals that a large number of them are motivated by their desire to contribute to science, which is a unique motivation among a number of crowdsourcing applications.

1.3 Risks in Crowdsourcing

As we have seen, crowdsourcing has been applied to diverse domains in order to make the impossible possible. As a consequence, many people from different fields can enjoy the advantages of crowdsourcing to realize a number of services that could not exist without it. However, we cannot dismiss disadvantages that are inherent to crowdsourcing and which the traditional labor market does not have. In this section, we illustrate three main risks in crowdsourcing and briefly review the existing research approaches to addressing them.

1.3.1 Quality Risk

One of the well-known risks is that the quality of task results varies depending on individual workers. Even the earliest crowdsourcing article reported a comment by one anonymous requester that *"I think half of the people signed up are trying to pull a scam"* (Howe, 2006a). In addition, one of the earliest adopters of crowdsourcing in natural language processing noticed that the reliability of the results depended on the workers (Snow et al., 2008).

The quality issue has three main causes. First, some workers called spammers intentionally return meaningless results to requesters to earn easy money. Some spammers use a computer bot to automatically generate uninformative results. Second, some workers, although they do not intend to trick the system, are not skilled enough to produce a high-quality result that meets the requirements of a requester. In addition, when a part of the task description is not clearly stated, workers often return results that are totally

different from the requester's true intention, which is another cause of poor quality results.

In the machine learning and data mining communities, extensive research has been conducted to address the quality issue by extracting informative results from redundantly collected results, which is called quality control (Lease, 2011). A common approach is an unsupervised learning method; we model the workers' processes of generating results from unobservable true results and use the model to aggregate multiple labels assigned to each instance into one label. This approach originally dates back to research that aggregates diagnoses by multiple doctors with different abilities (Dawid and Skene, 1979). This approach is valid when a majority of the workers are reliable. The detailed survey on this research topic appears in Section 6.2.1.

1.3.2 Unethical Abuse Risk

Unethical tasks are often posted to crowdsourcing marketplaces. For example, there exists a task asking workers to create multiple accounts on a web service and transfer the account information to the requester. Harris (2011) illustrates the following three examples of such unethical tasks.

The first example is review manipulation. Review sites such as Amazon.com and TripAdvisor have substantial influence on customers' decision making, and firms are sometimes tempted to post fake reviews by themselves. Crowdsourcing is used as a means to request such fake review writing (Lai et al., 2010). The New York Times reported that a task to write a positive review for a dentist was posted on Amazon Mechanical Turk, and a task to write a negative review was also found on Fiverr.com (Segal, 2011). These examples suggest that fake review writing tasks do exist in crowdsourcing marketplaces, and their purposes are not only to increase the reputation of the requesters, but also to criticize their competitors.

The second and third examples are surveillance and information gathering. These examples use crowdsourcing to collect information on a target person in the case of surveillance and secret information such as passwords and credit card numbers in the case of information gathering. Lasecki et al. (2014) showed that a non-negligible number of workers in Mechanical Turk were willing to engage in a task to extract a card number from a photograph, even if the task was apparently malicious. This research suggests that crowdsourcing has outlaw workers who will perform whatever task being posed to them.

A promising research direction to avoid the unethical abuse risk is to eliminate unethical tasks by machine learning techniques (Baba et al., 2014). Their approach was to learn a classifier discriminating improper tasks from

proper tasks using task descriptions together with the requester information, and they showed that it achieves satisfactory performance. In addition, they employed workers to check posted tasks in order to reduce the cost of experts patrolling the platform. By defining properness and improperness according to the terms and conditions of a crowdsourcing service, the unethical abuse risks described here can be addressed.

1.3.3 Privacy Risk

The third risk, which is the main topic of this thesis, is privacy. Noting that the traditional labor market has been handling a great deal of sensitive information in its daily routine, a crowdsourcing-based labor market also has to deal with various sensitive information. Such information can be protected by contracts and the law in the traditional labor market; however, this approach is unrealistic in a crowdsourcing setting because workers and requesters are transient and they usually are not required to sign contracts to retain the convenience of crowdsourcing.

An example of the privacy risk is crowdsourced Closed-Circuit Television (CCTV) surveillance. Its real deployment, called Internet eyes, utilizes crowdsourced labor to detect shoplifters; workers are allowed to view CCTV camera feeds and are encouraged to detect crimes for a monetary reward (Trottier, 2014). While this service successfully addresses the human resource issue of continuous surveillance of abundant CCTVs, it potentially brings about a privacy issue. A crowdsourced CCTV surveillance service broadcasts a shopping scene of innocent customers to unspecified crowd workers, which itself is distressing to the customers. In addition, it may induce discrimination; for example, customers who are often racially-discriminated against may more often be caught than those who are not by workers who are not trained properly. After its beta release in 2010, a dispute over privacy concerns was triggered. A shop owner called Jinx Hundal abandoned Internet Eyes after receiving a number of complaints (Big Brother Watch, 2011b). Big Brother Watch, a campaign group for privacy and civil liberties, warns that Internet Eyes in the UK accepts workers in countries who are not bounded by UK data protection and privacy laws (Big Brother Watch, 2011a), which suggests that workers from such countries can freely use the sensitive data, even in a malicious way.

Another example is the mobile crowdsourcing introduced in Section 1.2.4. Workers in mobile crowdsourcing are often asked to provide their location information as part of the task result and in order to assign tasks efficiently; workers far from the location specified by a task will not be appropriate for the task. However, location information is considered to be one of the most

sensitive information about an individual, and therefore, transferring it to other people can easily induce privacy issues. In fact, a number of research groups have pointed out the privacy risk of mobile crowdsourcing (*e.g.*, Wang et al. (2013); Yang et al. (2015)).

Despite these privacy concerns, little has been systematically investigated regarding privacy in crowdsourcing. In the research on mobile crowdsourcing, a series of works (Kazemi and Shahabi, 2012b,a, 2011; To et al., 2014) has discussed the privacy issues involved in the geographical task assignment. Other than the research on mobile crowdsourcing, few independent studies (Little and Sun, 2011; Varshney, 2012) have been conducted on the privacy issues in crowdsourcing before the studies contained in this thesis (Kajino et al., 2014a,b, 2015).

1.4 Research Questions

Of the risks enumerated in Section 1.3, we focus on the privacy risk in this thesis. While the quality and unethical abuse risks have been studied by many research groups and promising clues to their solutions are present, only a small part of the privacy risk has been clarified, despite its significance. In order to shed light on the privacy risk and establish the research basis for privacy-preserving crowdsourcing (PPCS), we propose the following two research questions:

- *What types of privacy risks are present in crowdsourcing?*
  It is essential to clarify the privacy risks to better understand the privacy concerns that lies behind the process of crowdsourcing. The focus of the existing PPCS studies is to develop a solution to a specific privacy issue in crowdsourcing, rather than to understand the whole picture of privacy risk. In this light, we summarize the privacy risks that can occur in crowdsourcing. Further, by examining the applicability of existing privacy preservation methods to each privacy risk, we highlight the novelty of PPCS against the existing privacy preservation research.

- *How can we measure and control the privacy risks in crowdsourcing?*
  For each privacy concern, we aim to develop an individual solution to preserve privacy as well as a measure to quantify the privacy risk. As we see later, each privacy risk has different properties, which hinders us from applying a single solution to all privacy issues. Therefore, we are obliged to tailor a privacy preservation method as well as a privacy guarantee for each privacy issue considering its unique property, which boils down to technical challenges.

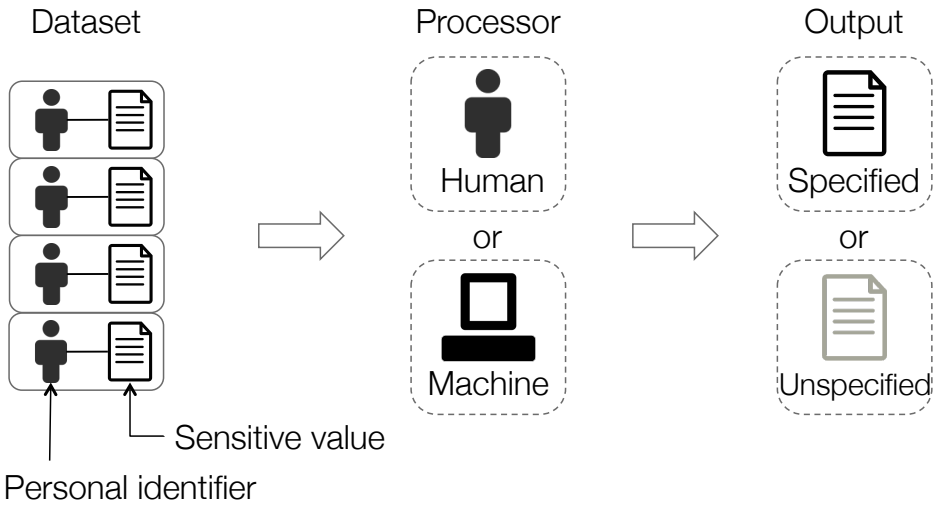


Figure 1.1: Illustration of typical data processing procedures. We assume that the dataset is made up of multiple records, each of which is an association of a personal identifier and a sensitive value. The dataset is processed by either a human or a machine to generate an output, which may not be specified. The existing privacy preservation research assumes that the processor is a machine, whereas the PPCS research deals with the case in which with a human processor is used, which is a new research area.

1.5 Privacy Risk Analysis

This section provides the answer to the first research question by identifying the privacy risks in crowdsourcing and clarifying the relationship between PPCS and the existing research on privacy preservation.

We first introduce two major approaches to preserving privacy and point out that two properties of a data processing procedure, the processor and output, play an important role in the selection of an appropriate privacy preservation approach. We then abstract crowdsourcing procedures into a simple model to specify the data processing procedures within it and discuss the possible privacy risks associated with them. Finally, we examine the two properties of the data processing procedures in crowdsourcing to discuss the relationship between PPCS and the existing privacy preservation research.

1.5.1 Two Approaches to Privacy Preservation

The research on privacy preservation aims to preserve privacy associated with data while maintaining its utility. The basic strategy for privacy preservation is to specify the data processing procedure and choose an appropriate

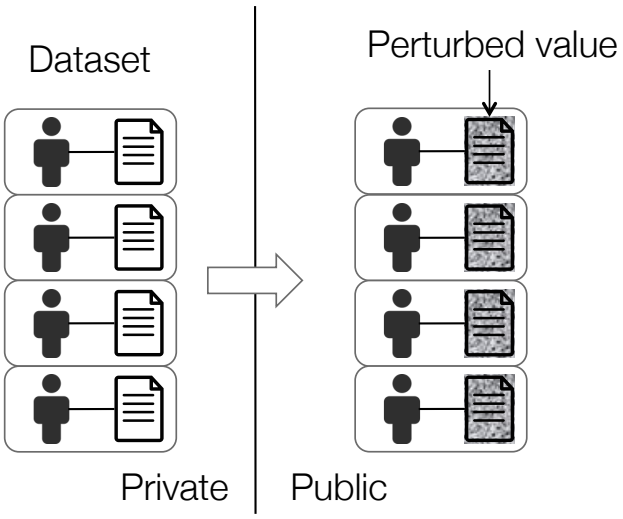


Figure 1.2: The perturbation approach publishes a perturbed dataset, in which the sensitive values are perturbed. It is appropriate if the output is *unspecified*.

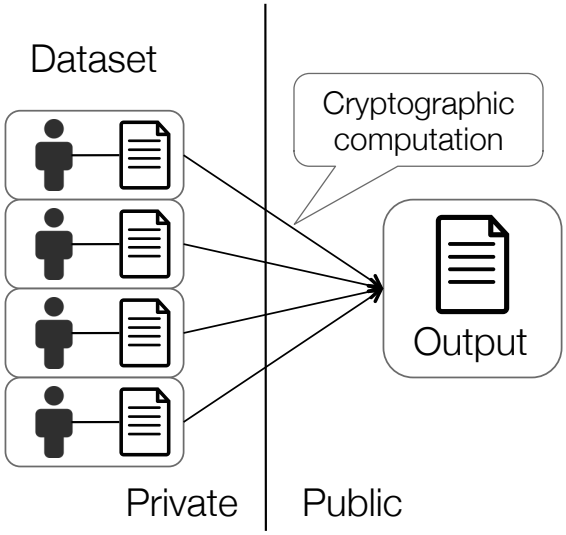


Figure 1.3: The cryptographic approach simulates the machinery computation of the output relying on cryptography and publishes the output only. It is preferable if the output is *specified*.

approach depending on its properties. Figure 1.1 depicts typical data processing procedures; given a possibly sensitive dataset, the processor (human or machine) processes the dataset to generate the output, which is either specified or not. The existing privacy preservation research assumes the processor to be a *machine*, while the output is allowed to be either *specified* or *not*.

There are two main approaches[11] to preserving privacy. The first approach is a perturbation approach (Figure 1.2), which perturbs the sensitive values in the dataset so that it satisfies some privacy criterion and publishes the perturbed dataset. This approach is suitable when the output is *unspecified*; given the perturbed dataset, it is possible to apply any computation to the dataset to obtain the output the user wants. The second approach is a cryptographic approach (Figure 1.3), which simulates the prescribed computation using cryptography to derive the output while still keeping the dataset private. This approach is feasible only when the output is *specified*. The advantage of the cryptographic approach over the perturbation approach is that it can compute the output exactly; the perturbation approach cannot compute the exact output because of the perturbation. The disadvantage of the cryptographic approach is its computation time; encryption, decryption, and key generation require additional computation time.

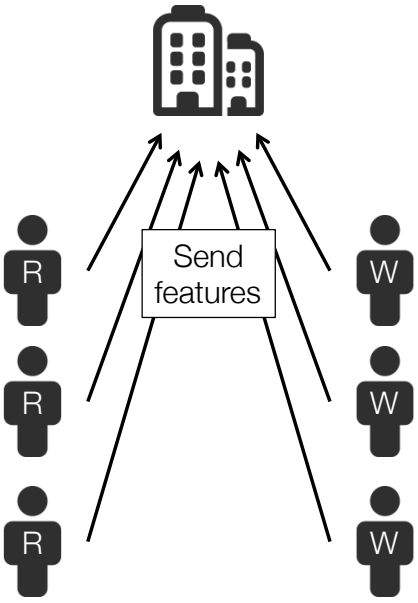
In summary, the existing research on privacy preservation focuses on the case where the processor is a machine and the output is either specified or not. Because both cryptographic and perturbation approaches have advantages and disadvantages, it is important to choose suitable approaches depending on the situation.

1.5.2 Crowdsourcing Model

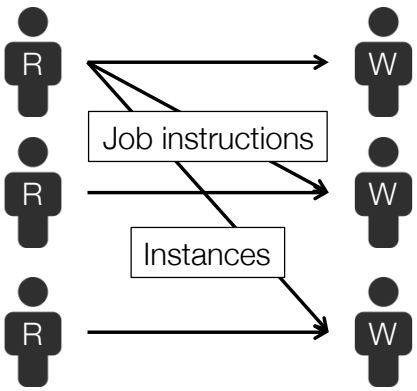
The crowdsourcing procedure is defined by the set of entities participating in it and the communication between them. We first introduce the entities who contribute to crowdsourcing and then describe a typical crowdsourcing routine.

[11] Anonymization (*i.e.*, removing the personal identifiers and making the sensitive values public) is one of the easiest solutions. However, anonymization by itself is not enough to preserve privacy. The risk of linking attacks has been discussed since Sweeney (2002) pointed out the risk.

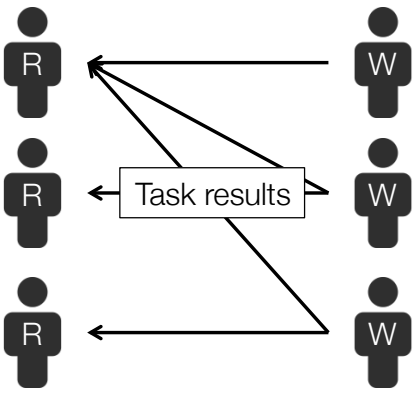
Figure 1.4: Entities in crowdsourcing.



(a) The platform collects features of the requesters and workers and computes a task assignment accordingly.



(b) Each requester sends the job instructions and task instances to the assigned workers.



(c) Each worker, having processed the tasks, sends back the task results to the requesters.

Figure 1.5: Overview of a crowdsourcing routine.

Entities

As shown in Figure 1.4, crowdsourcing consists of three types of entities:[12] requester, worker, and platform.

A requester uses crowdsourcing to have his/her task processed, which is made up of a job instruction and multiple instances. For example, if a task is an English-German translation task, a job instruction will be a text such as *"Please translate the English sentence shown below into German,"* and an instance corresponds to an English sentence to be translated.

A worker participates in crowdsourcing to perform tasks and sends the results back to requesters in exchange for a reward. For example, if a worker is assigned a translation task from English to German, the worker receives a job instruction and the English sentences (instances) from the requester, translates them into German sentences as instructed, and sends them back to the requester to be rewarded.

A platform manages the crowdsourcing service, and it mainly fulfills the following two roles. First, the platform works as a hub of communication. Any pair of entities are allowed to communicate with each other only via the platform. Both requesters and workers benefit from this service because most of the procedures necessary in a labor market can be completed by communicating only with the platform, including the task request, delivery of task results, and payment. Second, it manages a task assignment service between workers and tasks. There are two possible approaches to task assignment services: push-type and pull-type strategies. In a pull-type strategy (which is also called an open-call strategy), the platform provides a list of available tasks to workers, and workers pull the tasks they like. In a push-type strategy, the platform computes a task assignment based on the features of the workers and tasks and pushes the tasks to the appropriate workers.

In this thesis, we adopt the push-type task assignment for the following two reasons. The first reason is efficiency. Because some professional tasks and location-based tasks require special features of workers, the global task assignment is an effective way to increase the throughput of crowdsourcing. The second reason is security. The push-type assignment can significantly reduce the number of workers who browse instructions and instances of tasks compared to the pull-type assignment; in the pull-type assignment, all the workers are able to browse all the tasks, whereas in the push-type assignment, only the allocated workers can browse them. Because the instructions and instances also induce privacy risks, as discussed later, the push-type assignment is preferable to the pull-type assignment.

[12]We exchangeably call an entity a party, following the convention used in computer security.

Table 1.1: Four data processing procedures in crowdsourcing with privacy risks. Although the procedure for processing task results is generally unspecified, it is common that a requester applies a quality control method to them.

Data	Step	Processor	Output	Our Solution
Feature	Assignment	Machine	Assignment	Chap. 3
Instruction	Request	Human	Result	Chap. 3
Instance	Request	Human	Result	Chaps. 3 & 4
Result	Delivery	Unspecified	Unspecified	—
		Machine	Quality-controlled result	Chap. 5

Crowdsourcing Routine

The crowdsourcing routine is made up of three steps: assignment, request, and delivery. We adopt this abstraction for crowdsourcing throughout this dissertation. The details are specified when necessary.

Step 1: Assignment (Figure 1.5a)
The platform collects the features of the workers and tasks, and computes the task assignment maximizing the throughput based on the features. The features of a worker include the skills s/he has, his/her current location, minimum wage, and working hours, for example. The features of a task include the corresponding properties the task requires to be completed.

Step 2: Request (Figure 1.5b)
Each requester sends the job instruction and instances to the allocated workers via the platform.

Step 3: Delivery (Figure 1.5c)
Each worker, receiving multiple tasks, completes each task to generate the results by processing the instances following each job instruction. Then, s/he sends the results back to the respective requesters via the platform. If the requester confirms the delivered results, the worker is rewarded.

1.5.3 Analysis

We specify the four data processing procedures of crowdsourcing and discuss the privacy risks associated with them as well as the applicability of the

existing privacy preservation strategies. We also classify the existing PPCS studies into the four categories to clarify the current research progress of PPCS. The analysis in this section is summarized in Table 1.1.

Assignment Step

In the assignment step, both workers and requesters are asked to send their features to the platform so that it can compute the task assignment. The features of workers can be used to identify them, infer sensitive information about them, physically harass them using the location information, and unfairly treat them by excessively favoring highly-skilled and hardworking workers. The features of tasks can also be used to identify the requesters and reveal the contents of the tasks. Therefore, the features of both workers and tasks are sensitive.

The features are processed by a *machine* to compute the task assignment (*i.e.*, the output is *specified*). Therefore, it is possible to compute the task assignment without invading privacy using a cryptographic approach.

The privacy issues in task assignment have been discussed mostly in the research area of spatial crowdsourcing (Kazemi and Shahabi, 2011, 2012b; To et al., 2014). Their concern is the leakage of worker location information, and they resort to perturbation approaches to preserve this privacy.

Request Step

In the request step, each requester sends his/her job instruction along with the task instances, both of which can cause privacy troubles. A job instruction leaks the intention of the requester, which can be the future direction of his/her business. In addition, it can leak the identity of the requester if the task is highly specialized. For example, a worker may be able to infer the future direction of a real business firm. In addition, instances can leak a substantial amount of sensitive information to the worker who processes them. For example, consider a task to transcribe audio recordings of business meetings, where a single audio recording corresponds to an instance. The content of such a recording can be confidential information of the requester or a third party. Many other tasks involve instances containing sensitive information, such as a task to digitize handwritten texts or to detect objects in images. Therefore, both job instructions and instances are sensitive.

Both job instructions and instances are processed by *humans* with the aim of completing the tasks (*i.e.*, the output is *specified*). Because the existing privacy preservation methods cannot handle a procedure with a human processor, neither of the privacy preservation methods is appropriate. Al-

though it is possible to apply the perturbation approach heuristically, there are no guarantees on utility and privacy, which hinders us from putting it into practice.

The privacy issues of instances have been discussed in the community of human-computer interaction. Little and Sun (2011) focused on transcribing a medical chart with privacy guarantees. By decomposing a medical chart into forms using a template, it is possible to transcribe it without privacy invasion. The privacy issues of job instructions have never been discussed in the research community.

Delivery Step

In the delivery step, each worker, having finished the tasks, returns the results of the tasks to the corresponding requesters. In the case of a location-based task, a result contains the location information of the worker, which itself is sensitive information. Furthermore, even if the result itself is not sensitive, it is possible to infer the features of workers from the results by probabilistic inference; it is already common to infer the ability of workers even from binary labels in order to eliminate low-ability workers. These inferred features can be used to identify and harm the workers. Therefore, not only the location information but even simple labels can lead to privacy invasion.

The data processing procedure of the results is in general unspecified, that is, both the processor and the output are unspecified. Therefore, it is not possible to apply the existing privacy preservation methods. However, with careful observation on crowdsourcing, we notice that in some cases, task results are processed by a *machine* for the purpose of quality control (*i.e.*, the output is *specified*). In this special case, the cryptographic approach is appropriate.

The privacy issues of results have been discussed in participatory sensing (Cornelius et al., 2008; Huang et al., 2009; Hu and Shahabi, 2010; Puttaswamy et al., 2010; Hu and Shahabi, 2010). Their concern is that the location information of workers can leak from the results, and they resort to a perturbation approach or anonymization for privacy preservation.

1.6 Solutions

As we stated in the previous section, there are four types of data that can trigger their own privacy risks, and they are associated with heterogeneous data processing procedures, some of which cannot be handled by the existing privacy preservation methods. We tackle these privacy risks with the follow-

ing three solutions. Table 1.1 summarizes the correspondence between the data and solutions.

1.6.1 Privacy-Preserving Task Assignment (Chapter 3)

We present a privacy-preserving task assignment that executes a push-type assignment without revealing the features of either the workers or requesters. It alleviates the privacy risks associated with features as well as instructions and instances compared to the standard pull-type task assignment, as discussed above. Noting that the output is specified in the feature processing procedure, we employ the cryptographic approach for privacy preservation rather than the perturbation approach that the existing studies employ. In specific, we formalize the task assignment problem as a maximum flow problem, and develop a privacy-preserving push-relabel algorithm using the Paillier cryptosystem (Paillier, 1999) to obtain an optimal task assignment without disclosing the features of either the workers or requesters.

1.6.2 Instance-Privacy Preservation (Chapter 4)

We present a utility-privacy trade-off analyzer (UPTA) and a case study of the analysis on an instance-clipping (IC) protocol. UPTA evaluates the utility and privacy of the instance-privacy preserving (IPP) protocols, given the definitions of a task and privacy. Because a worker performs the task and at the same time invades privacy in an IPP protocol, it is necessary to quantify both utility and privacy, taking the participation of human processors into account. Our idea for quantification is to model both the task execution and privacy invasion processes as sampling of a result and sensitive value from probability distributions. These models can be empirically estimated using crowdsourcing, and given these models, divergence-based measures can be computed. As a case study of UPTA, we develop the IC protocol and investigate its properties. The IC protocol is a generalization of the method proposed by Little and Sun (2011). It clips instances with a fixed-size window and hands the clipped instances to workers for task execution while hindering privacy invasion. We apply UPTA to analyze the properties of the IC protocol and to determine the clipping window size. We further study the validity of our performance measures theoretically and empirically.

1.6.3 Worker-Privacy Preservation (Chapter 5)

We present a worker-private latent class (WPLC) protocol, which preserves the sensitive information contained in results. Noticing that in some cases,

results are processed by a quality control method and that its output is much less sensitive than the original results, our idea is to simulate the quality control method using cryptography, *i.e.*, to aggregate the multiple results acquired from multiple workers into one synthetic result and deliver the aggregated results to the requester. In specific, we develop a privacy-preserving variant of the latent class method (Dawid and Skene, 1979) using the Paillier cryptosystem. The security of the protocol is guaranteed such that it is impossible to infer any original result from the aggregated results.

1.7 Roadmap

This thesis is organized into a preliminary chapter (Chapter 2), three main research results (Chapters 3, 4, and 5), related work (Chapter 6), and the conclusion (Chapter 7).

In Chapter 2, we introduce the notation we use throughout this dissertation, the Paillier cryptosystem, which is used in Chapters 3 and 5, and the privacy assumptions in crowdsourcing. We then present the research results of this thesis in three chapters. Chapter 3 addresses the privacy issues in task assignment, which is under the review of *Knowledge and Information Systems* (Kajino et al., 2015). Chapter 4 addresses the privacy issues in task submission, which was published and presented in the proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing (Kajino et al., 2014b). Chapter 5 addresses the privacy issues in result collection from workers, which was published in *Data Mining and Knowledge Discovery* (Kajino et al., 2014a) and was presented at the Seventh European Machine Learning and Data Mining Conference in 2014.

Chapter 2

Preliminaries

This chapter defines the basic notation and cryptographic techniques we use in this dissertation. We first define the notation commonly used throughout this dissertation. Then, we introduce a cryptographic approach to preserving privacy. In specific, we introduce the notion of public-key encryption and its security and the Paillier cryptosystem (Paillier, 1999), which is used as a cryptographic building block in our protocols. Finally, we introduce privacy assumptions of our crowdsourcing model that define the behavior of each entity in a protocol.

2.1 Notation

We summarize the notation we use commonly in this thesis.

Let $\mathbb{Z}_+$ be the set of non-negative integers, and let $\mathbb{Z}_n := \{0, \ldots, n-1\}$ for any $n \geq 1$. Let $[K] = \{1, 2, \ldots, K\}$ $(K \geq 1)$. Given vector $\mathbf{x} \in \mathbb{R}^D$, x_d $(d \in [D])$ denotes the d-th dimension of vector $\mathbf{x}$. Given a set of real numbers $\{a_{i,j} \mid i \in [I],\ j \in [J]\}$, let $\left[a_{i,j}\right]_{i\in[I],j\in[J]}$ be an $I \times J$ matrix whose (i, j) element corresponds to $a_{i,j}$. A multiset is defined as a set whose elements may be duplicated. Let $\{a\}^b$ denotes a multiset consisting of b duplicated elements of a.

2.2 Public-Key Encryption

We introduce a basic notion of public-key encryption and its security in this section. A large part of this section is extracted from a textbook on modern cryptography (Katz and Lindell, 2007), otherwise indicated.

2.2.1 Public-Key Encryption Scheme

Let us consider a two-party setting for simplicity, in which we have a sender and a receiver, and the sender wants to send a message to the receiver in a private way. A simple cryptographic solution is to use a secret key; the two parties share a secret key and use it for both encryption and decryption. However, this can be infeasible in many realistic settings, because it requires that the two parties must share the secret key somehow, *e.g.*, by physically gathering at a private room.

A public-key encryption scheme successfully addresses the key sharing issue by generating a public key for encryption and a secret key[1] for decryption. The key idea is that encryption requires the public key only, whereas decryption requires the secret key. If a receiver distributes a public key and keeps a secret key private, a sender obtaining the public key can encrypt his message, and only the receiver holding the secret key can decrypt and read the message. We give the definition of a public-key encryption scheme in Definition 2.1. We especially utilize a variation of public-key encryption schemes called the Paillier cryptosystem (Paillier, 1999), which will be introduced in Section 2.3.

Definition 2.1 (Public-key encryption scheme)**.** *A public-key encryption scheme is a tuple of probabilistic polynomial-time algorithms* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfying the following four conditions:*

1. *Key generation*
   Algorithm Gen *takes as input a security parameter* 1^n *and outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*, where* pk *is the public key, and* sk *is the secret key.*

2. *Encryption*
   Algorithm Enc *takes as input the public key* pk *and a message* m *from a pre-defined plaintext space, and outputs a ciphertext* c*. This process is written as* $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m)$*. Note that encryption may entail randomness, i.e., a ciphertext may change at every time of encryption.*

3. *Decryption*
   Algorithm Dec *takes as input the secret key* sk *and a ciphertext* c*, and outputs a message* m *or a special symbol* $\perp$ *denoting failure. Assuming that decryption is deterministic, this process is written as* $m := \mathsf{Dec}_{\mathsf{sk}}(c)$*.*

4. *Relationship between Encryption and Decryption*
   For every n*, every* $(\mathsf{pk}, \mathsf{sk})$ *generated by* $\mathsf{Gen}(1^n)$*, and every message* m

[1] We use the term, a secret key, instead of a private key.

in the plaintext space, it holds that

$$\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m.$$

A typical scenario is as follows. A receiver, setting the security parameter, generates a pair of keys $(\mathsf{pk}, \mathsf{sk})$ and distributes the public key pk while keeping sk secret. A sender who wants to send a message m to the receiver uses the public key pk and the encryption function Enc to generate a ciphertext c, and send it to the receiver. The receiver uses the secret key sk and the decryption function Dec to decrypt the ciphertext c to obtain the original message m. If decryption without the secret key is hard, this process does not leak any information of the message to the third party eavesdropping the communication.

2.2.2 Security

We then discuss how to guarantee the security of a cryptosystem. A public-key encryption basically relies on the computational security; we initially make an assumption that some low-level problem, such as factoring the product of two large prime numbers, is hard to solve, and the security of the cryptosystem is theoretically guaranteed by reducing an attack on it to solving the low-level problem. In order to give a rigorous definition, we rely on the asymptotic approach, where a cryptosystem is defined to be secure if the probability that an efficient adversary succeeds in decryption is asymptotically negligible as the security parameter n increases. The goal of this section is to define this security notion rigorously.

First, we define a model of an adversary, who tries to break the encryption to read the original message. The model consists of its computation power and its attacking process.

The computation power of an adversary is assumed that it can run only a probabilistic algorithm in time polynomial in n. This assumption is necessary for computational security, because the security relies on the computational difficulty of a low-level problem that is difficult to solve in polynomial time, but is solvable in exponential time.

The definition of its attacking process has some options reflecting the strength of security. In this thesis, we consider the chosen-plaintext attack (CPA) only. The chosen-plaintext attack is modeled by the CPA indistinguishability experiment $\mathsf{PubK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n)$ (Protocol 2.1), which is performed by an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The CPA indistinguishability experiment is a realistic setting in that the adversary has as much information as possible, *i.e.*, the public key and the pair of messages in plaintext. Intuitively,

Protocol 2.1 CPA indistinguishability experiment $\mathsf{PubK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n)$.

Parties: Adversary $\mathcal{A}$ and challenger $\mathcal{C}$.

Input: Security parameter n and cryptosystem Π.

1: Challenger $\mathcal{C}$ runs $\mathsf{Gen}(1^n)$ to obtain a pair of keys $(\mathsf{pk}, \mathsf{sk})$.
2: Adversary $\mathcal{A}$ is given the public key pk and oracle access to $\mathsf{Enc}_{\mathsf{pk}}(\cdot)$. Adversary $\mathcal{A}$ generates a pair of messages m_0, m_1 with the same bit from the plaintext space, and sends them to challenger $\mathcal{C}$.
3: Challenger $\mathcal{C}$ generates a uniformly random bit $b \in \{0, 1\}$, and encrypts message m_b to generate $c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_b)$. Challenger $\mathcal{C}$ sends the ciphertext to adversary $\mathcal{A}$.
4: Adversary $\mathcal{A}$ guesses whether $m_b = m_0$ or $m_b = m_1$, and sends his/her guess bit b' to challenger $\mathcal{C}$.
5: Challenger $\mathcal{C}$ outputs 1 if $b' = b$ and 0 otherwise, which is the output of the experiment.

if the adversary cannot make a correct guess even if s/he knows the original messages, then the cryptosystem is defined as secure.

Then, given the adversary model defined above, we provide the asymptotic statement of the security in Definition 2.2. A cryptosystem satisfying Definition 2.2 is referred as IND-CPA. In order to allow some tolerance, a negligible function (Definition 2.3) is used.

Definition 2.2 (IND-CPA)**.** *Public-key encryption scheme* $\Pi = (\mathit{Gen}, \mathit{Enc}, \mathit{Dec})$ *has indistinguishable encryptions under chosen-plaintext attacks (or is CPA secure) if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\Pr[\mathit{PubK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathit{negl}(n).$$

Definition 2.3 (Negligible function)**.** *A function* f *is negligible if for every polynomial function* $p(\cdot)$*, there exists an integer* N *such that for all integers* $n > N$*,* $f(n) < \frac{1}{p(n)}$ *holds.*

Although Definition 2.2 deals with a single encryption setting where the adversary can use a pair of single messages, m_0 and m_1, it can be shown that an IND-CPA cryptosystem is secure under the multiple encryptions setting where the adversary generate a pair of multiple messages, $(m_0^1, \ldots, m_0^t)$ and $(m_1^1, \ldots, m_1^t)$ (Theorem 10.10 (Katz and Lindell, 2007)).

IND-CPA implies that a party cannot distinguish between ciphertexts of sensitive information and ciphertexts of random values. In this thesis, we

define a party learns nothing after executing a protocol if s/he only obtains a sequence of uniformly random variables and ciphertexts that are IND-CPA during the protocol.

2.3 Paillier Cryptosystem

The Paillier cryptosystem (Paillier, 1999) is known as a probabilistic public-key encryption scheme with the additive homomorphic property. We use the Paillier cryptosystem as a building block to preserve privacy; in Chapter 3, the Paillier cryptosystem is used to encrypt all the information of the requesters and workers, and in Chapter 5, the Paillier cryptosystem encrypts task results generated by the workers.

We first give a brief overview of the Paillier cryptosystem in Section 2.3.1 and its properties in Section 2.3.2. Then, for completeness, we review the correctness of the encryption and decryption algorithms and its security in Sections 2.3.3 and 2.3.4. Finally, we provide experimental results on the computation time of basic operations of the Paillier cryptosystem in Section 2.3.5.

2.3.1 Overview

We describe the overview of the Paillier cryptosystem by illustrating the key generation, encryption, and decryption algorithms. A party who encrypts a plaintext is called an encryptor, and a party who decrypts a ciphertext is called an decryptor.

Notation

Let $N = pq$ where p and q are large primes. Let $\gcd(a, b)$ and $\mathrm{lcm}(a, b)$ be the greatest common divisor and the least common multiple of two integers a and b, respectively. For example, $\gcd(4, 6) = 2$ and $\mathrm{lcm}(4, 6) = 12$. Let $\mathbb{Z}_N := \{0, \ldots, N-1\}$ and $\mathbb{Z}_N^* := \{z \in \mathbb{Z}_N \mid \gcd(z, N) = 1\}$.

Key Generation

The cryptosystem first generates a pair consisting of a public key and a secret key $(\mathsf{pk}, \mathsf{sk})$ as follows:

1. Generate two large prime numbers p and q.

2. Generate secret key sk as $\mathsf{sk} = \lambda\ (= \mathrm{lcm}(p-1, q-1))$.

3. Generate public key $\mathsf{pk} = (N, g)$, where $N = pq$ and g is a uniformly random sampling from $\mathbb{Z}_{N^2}^*$ such that the order of g is a nonzero multiple of N.

The public key, which is used for both encryption and decryption, is shared among all the parties, whereas the secret key, which is necessary for decryption, is privately held by the decryptor. Note that conventionally g is set as $N + 1$ for simplicity, whose order is proven to be N. This does not ruin the security of the Paillier cryptosystem because the security does not depend on g but only on N.

Encryption

Let us denote a plaintext by $m \in \mathbb{Z}_N$. The encryption algorithm generates a ciphertext of m, which behaves as a uniformly random variable over $\mathbb{Z}_{N^2}^*$. The algorithm proceeds as follows:

1. The encryptor generates a uniformly random variable $r \in \mathbb{Z}_N^*$.
2. The encryptor computes $c = \mathsf{Enc}_{\mathsf{pk}}(m; r) = g^m r^N \mod N^2$.

As proven in Section 2.3.4, the Paillier cryptosystem is IND-CPA under some assumptions, *i.e.*, any party who does not have the secret key learns nothing about the plaintext from the ciphertext. Note that ciphertext c depends on both plaintext m and random variable r, which is randomly chosen every time the encryptor encrypts. Therefore, ciphertexts of the same plaintext do not necessarily take the same value, which prevents a party from learning that two ciphertexts come from the same plaintext. Note also that, in the remainder of this paper, we often omit random variable r and public key pk, and we denote the encryption of plaintext m by $\mathsf{Enc}(m)$.

Decryption

Given ciphertext $c \in \mathbb{Z}_{N^2}^*$, public key $\mathsf{pk} = (N, g)$, and secret key $\mathsf{sk} = \lambda$, the decryption algorithm outputs plaintext $m \in \mathbb{Z}_N$ of the ciphertext. The algorithm proceeds as follows:

1. The decryptor computes

$$m = \frac{L(c^\lambda \mod N^2)}{L(g^\lambda \mod N^2)} \mod N,$$

where $L(u) := \dfrac{u-1}{N}$.

Computational Example

For better understanding of the Paillier cryptosystem, we provide a computation example. Let $(p, q) = (3, 5)$ and $N = pq = 15$. Then,

$$\begin{aligned}
\mathbb{Z}_N &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}, \\
\mathbb{Z}_N^* &= \{1, 2, 4, 7, 8, 11, 13, 14\}, \\
\mathbb{Z}_{N^2}^* &= \{1, 2, 4, 7, 8, 11, 13, 14, 16, 17, 19, 22, 23, 26, 28, 29, 31, 32...\}.
\end{aligned}$$

Let the public key be $\mathsf{pk} = (N, g) = (15, 16)$, where the order of $g \in \mathbb{Z}_{N^2}^*$ should be a nonzero multiple of N. $g = 16$ satisfies the condition because $16^{15} = 1 \mod 225$ holds, and $16^m \neq 1 \mod 225$ for all $m \in \{1, 2, \ldots, 14\}$. For example, $g = 19$ is not appropriate because $19^{10} = 1 \mod 225$, *i.e.*, the order of 19 is 10. The secret key is $\mathsf{sk} = \lambda = \mathrm{lcm}(p-1, q-1) = \mathrm{lcm}(2, 4) = 4$. Let the plaintext be $m = 10$. Generating a random variable $r = 8$ from $\mathbb{Z}_N^*$, the encryptor computes the ciphertext as

$$\begin{aligned}
c &= \mathsf{Enc}_{\mathsf{pk}}(m; r) \\
&= g^m r^N \mod N^2 \\
&= 16^{10} \cdot 8^{15} \mod 225 \\
&= 3868562622766813359059763 2 \mod 225 \\
&= 182 \in \mathbb{Z}_{N^2}^*.
\end{aligned}$$

The decryption algorithm results in the following computation:

$$\begin{aligned}
m &= \frac{L(c^\lambda \mod N^2)}{L(g^\lambda \mod N^2)} \mod N \\
&= \frac{L(182^4 \mod 225)}{L(16^4 \mod 225)} \mod 15 \\
&= \frac{L(151)}{L(61)} \mod 15 \\
&= \frac{(151-1)/15}{(61-1)/15} \mod 15 \\
&= \frac{10}{4} \mod 15 \\
&= 10 \cdot 4 \mod 15 = 10,
\end{aligned}$$

which corresponds to the original plaintext. In the above computation, we use $4 \cdot 4 = 1 \mod 15$, which implies $4^{-1} = 4 \mod 15$.

2.3.2 Properties

We enumerate several properties of the Paillier cryptosystem, which we utilize to design protocols.

Probabilistic Public-key Encryption

Any party who does not have the secret key learns nothing about the plaintext from the ciphertext mainly because of the following three reasons. First, since the Paillier cryptosystem is proven to be IND-CPA under some assumptions, decryption without the secret key is computationally hard. Therefore, the plaintext cannot be obtained from the ciphertext without the secret key. Second, since the Paillier cryptosystem employs a probabilistic encryption scheme, even if two encryptors encrypt the same plaintext, the resultant ciphertexts usually take different values if they choose different random variables r_1 and r_2 for encryption. Third, from the property of a public key cryptosystem, accumulating multiple ciphertexts does not provide any information about the plaintexts.

Additive Homomorphic Property

In addition to the probabilistic public-key encryption properties, the Paillier cryptosystem has the additive homomorphic property; the addition of two encrypted plaintexts can be computed without decryption. Given $\mathsf{Enc}_{\mathsf{pk}}(m_1; r_1)$ and $\mathsf{Enc}_{\mathsf{pk}}(m_2; r_2)$, $\mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2; r)$ can be computed as

$$\mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2; r) = \mathsf{Enc}_{\mathsf{pk}}(m_1; r_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(m_2; r_2) \mod N^2,$$

where r is uniformly distributed if either r_1 or r_2 is uniformly distributed. This can be confirmed by the following basic arithmetic:

$$\begin{aligned}
&\mathsf{Enc}_{\mathsf{pk}}(m_1; r_1) \cdot \mathsf{Enc}_{\mathsf{pk}}(m_2; r_2) \mod N^2 \\
=&g^{m_1} r_1^N \cdot g^{m_2} r_2^N \mod N^2 \\
=&g^{m_1+m_2} \cdot (r_1 r_2)^N \mod N^2 \\
=&\mathsf{Enc}_{\mathsf{pk}}(m_1 + m_2 \mod N; r_1 r_2 \mod N),
\end{aligned}$$

where $r_1 r_2$ distributes uniformly over $\mathbb{Z}^*_{N^2}$ (see, Lemma 2.1 below). Subtraction can also be implemented. Given a ciphertext of plaintext m, $c = g^m r^N$, it is easy to compute $c^{-1} \in \mathbb{Z}^\star_{N^2}$ by the Euclidean algorithm; the algorithm can find integers x and y such that $cx + N^2 y = 1$, which implies that x is the inverse of c in $\mathbb{Z}^\star_{N^2}$. Noticing that

$$c \cdot c^{-1} = 1 = g^0 1^N$$

holds in the space of ciphertexts,

$$m + \mathsf{Dec}(c^{-1}) = 0$$

must hold in the space of plaintexts, which indicates that multiplying c^{-1} in the space of ciphertexts corresponds to the subtraction of m in the space of plaintexts. Negative values can also be handled by shifting the plaintext space appropriately.

Lemma 2.1 (Lemma 10.18 (Katz and Lindell, 2007))**.** *Let G be a finite group with operation $\cdot$, and let $m \in G$ be an arbitrary element of G. If we choose uniformly random element $g \leftarrow G$ and set $g' := m \cdot g$, g' distributes uniform-randomly over G, i.e.,*

$$\Pr[m \cdot g = \hat{g}] = 1/|G|$$

holds for any $\hat{g} \in G$.

Proof. Let $\hat{g} \in G$ be a fixed arbitrary element of G. Then,

$$\Pr[m \cdot g = \hat{g}] = \Pr[g = m^{-1} \cdot \hat{g}]$$

holds where the probability is taken over the distribution of g. Since g is uniform-randomly distributed over G, $\Pr[g = m^{-1} \cdot \hat{g}] = 1/|G|$ holds for any fixed $\hat{g} \in G$, which concludes the proof. □

2.3.3 Correctness of the Paillier Cryptosystem

We explain the correctness of the encryption and decryption algorithms of the Paillier cryptosystem by showing that (i) the encryption function is bijective, and (ii) the decryption function correctly recovers the original plaintext.

Basic Properties of $\mathbb{Z}_m^*$

We state several important properties of $\mathbb{Z}_m^*$. To do so, we first introduce two number theoretic functions: Euler's totient function ϕ (Definition 2.4) and Carmichael's function λ (Definition 2.5).

Definition 2.4 (Euler's totient function)**.** *Let $\phi(x)$ be Euler's totient function that counts the positive integers less than or equal to x that are relatively prime to x.*

Definition 2.5 (Carmichael's function)**.** *Let $\lambda(x)$ be Carmichael's function, which is defined as the smallest positive integer m such that $a^m = 1 \mod x$ for every integer a that is coprime to x.*

In our case where $N = pq$ (p and q are odd prime numbers), $\phi(N) = (p-1)(q-1)$ and $\lambda(N) = \mathrm{lcm}(p-1, q-1)$, which are consequences of Carmichael's theorem. Noticing that $\lambda(N^2) = N\lambda(N)$, the definition of Carmichael's function implies that for any $w \in \mathbb{Z}^*_{N^2}$,

$$w^{\lambda(N)} = 1 \mod N,\ w^{N\lambda(N)} = 1 \mod N^2 \tag{2.1}$$

hold. In the following, we simply denote $\lambda(N)$ by λ.

Then, we summarize important properties of $\mathbb{Z}^*_m$ below:

(a) $|\mathbb{Z}^*_m| = \phi(m)$,

(b) $\mathbb{Z}^*_m$ forms a group together with multiplication,

(c) $\mathbb{Z}^*_m \subset \mathbb{Z}^*_{m^2}$.

The first statement as to the number of elements in $\mathbb{Z}^*_m$ is self-evident from the definition of Euler's totient function (Definition 2.4). The second statement that $\mathbb{Z}^*_m$ forms a group can be proven by utilizing the fact that two integers a and b are coprime if and only if there exist two integers x and y such that $ax+by = 1$. The third statement holds because any element in $\mathbb{Z}^*_m$ is coprime to m by definition, which implies that it is also coprime to m^2.

(i) Bijectivity of the Encryption Function

Now we are able to prove that the encryption function is bijective. We first re-introduce the encryption function using a different notation, $\mathcal{E}_g(m, r)$ with parameter g in Definition 2.6, and define a set of valid parameters $\mathcal{B}_1$ in Definition 2.7 that plays an important role to make the encryption function bijective from (m, r) to a ciphertext c. Note that as we saw in the computational example, $\mathcal{B}_1 \subsetneq \mathbb{Z}^*_{N^2}$ holds, because the order of some $b \in \mathbb{Z}^*_{N^2}$ may be smaller than N.

Definition 2.6 (Encryption function)**.** *Let m be a plaintext and r be a random variable. For $g \in \mathbb{Z}^*_{N^2}$, we define the encryption function $\mathcal{E}_g : \mathbb{Z}_N \times \mathbb{Z}^*_N \to \mathbb{Z}^*_{N^2}$ as $\mathcal{E}_g(m, r) = g^m r^N \mod N^2$.*

Definition 2.7 (Base set)**.** *For $N = pq$, the base set $\mathcal{B}_1 \subset \mathbb{Z}^*_{N^2}$ is defined as*

$$\mathcal{B}_1 = \{b \in \mathbb{Z}^*_{N^2} \mid \mathrm{ord}(b) = N\},$$

where $\mathrm{ord}(b)$ *($b \in \mathbb{Z}^*_{N^2}$) denotes the order of b, i.e., the smallest positive integer m such that $b^m = 1 \mod N^2$.*

Then, Lemma 2.2 states the bijectivity of the encryption function.

Lemma 2.2 (Lemma 3 (Paillier, 1999)). *If the order of $g \in \mathbb{Z}_{N^2}^*$ is a nonzero multiple of N, then $\mathcal{E}_g$ is bijective.*

Proof. We first show that $\mathbb{Z}_N \times \mathbb{Z}_N^*$ and $\mathbb{Z}_{N^2}^*$ have the same number of elements, $N\phi(N)$. This holds because $|\mathbb{Z}_N \times \mathbb{Z}_N^*| = N\phi(N)$ and $|\mathbb{Z}_{N^2}^*| = \phi(N^2) = N\phi(N)$ hold. Then, we only have to prove that $\mathcal{E}_g$ is injective, *i.e.*,

$$\forall (m_1, r_1), (m_2, r_2) \in \mathbb{Z}_N \times \mathbb{Z}_N^*,$$
$$(m_1, r_1) \neq (m_2, r_2) \Longrightarrow \mathcal{E}_g(m_1; r_1) \neq \mathcal{E}_g(m_2; r_2)$$

holds.

Suppose that $g^{m_1} r_1^N = g^{m_2} r_2^N \mod N^2$ holds. By taking the λ-th power of the both sides and combining it with Equation (2.1), we obtain

$$g^{\lambda \cdot (m_1 - m_2)} = 1 \mod N^2.$$

The assumption that the order of g is a nonzero multiple of N and the fact that $\gcd(\lambda, N) = 1$ imply that $m_1 - m_2$ is a non-zero multiple of N. Since m_1 and m_2 are elements of $\mathbb{Z}_N$, $m_1 = m_2$ must hold.

As for r_1 and r_2, we have $r_1^N = r_2^N \mod N$ by applying $m_1 = m_2$. Since $f(r) = r^N \mod N$ is bijective, the above equation has the unique solution $r_1 = r_2$ (see, Corollary 7.17 (Katz and Lindell, 2007)).

In summary, we have proven that $g^{m_1} r_1^n = g^{m_2} r_2^n \mod N^2$ implies $m_1 = m_2$ and $r_1 = r_2$, *i.e.*, $\mathcal{E}_g$ is injective, which concludes the proof. □

(ii) Correctness of the Decryption Function

We show that the decryption process successfully recovers the original plaintext using the bijectivity of the encryption function; the inverse function of it can be well-defined as the N-th residuosity class (Definition 2.8).

Definition 2.8 (N-th residuosity class). *For $c \in \mathbb{Z}_{N^2}^*$, the N-th residuosity class of c with respect to $g \in \mathcal{B}_1$ is defined as the unique integer $m \in \mathbb{Z}_N$ for which there exists $r \in \mathbb{Z}_N^*$ such that*

$$\mathcal{E}_g(m, r) = c.$$

We denote the N-th residuosity class of c with respect to g by $[c]_g$, i.e., $\mathcal{E}_g([c]_g, r) = c$ for some $r \in \mathbb{Z}_N^*$.

The N-th residuosity class has the following formulae (Lemma 2.3, Corollary 2.1, and Lemma 2.4), which are used for decryption. Lemma 2.3 and Corollary 2.1 state the change-of-base formulae, and Lemma 2.4 states a formula of function L used in decryption.

Lemma 2.3. *For any $c \in \mathbb{Z}_{N^2}^*$ and $g_1, g_2 \in \mathcal{B}_1$,*

$$[c]_{g_1} = [c]_{g_2}[g_2]_{g_1} \mod N.$$

Proof. Given that the encryption function is bijective, for (g_1, c), (g_2, c), and (g_1, g_2), there respectively exist r_1, r_2, and r_3 such that

$$\mathcal{E}_{g_1}([c]_{g_1}, r_1) = g_1^{[c]_{g_1}} r_1^N \mod N^2 = c, \tag{2.2}$$
$$\mathcal{E}_{g_2}([c]_{g_2}, r_2) = g_2^{[c]_{g_2}} r_2^N \mod N^2 = c, \tag{2.3}$$
$$\mathcal{E}_{g_1}([g_2]_{g_1}, r_3) = g_1^{[g_2]_{g_1}} r_3^N \mod N^2 = g_2. \tag{2.4}$$

From Equation (2.4), we obtain

$$\begin{aligned} g_2^{[c]_{g_2}} \mod N^2 &= \left(g_1^{[g_2]_{g_1}} r_3^N\right)^{[c]_{g_2}} \mod N^2 \\ &= g_1^{[g_2]_{g_1}[c]_{g_2}} r_3^N \mod N^2 \\ &= \mathcal{E}_{g_1}([g_2]_{g_1}[c]_{g_2} \mod N, r_3), \end{aligned}$$

where we use the fact that the order of $g_1 \in \mathcal{B}_1$ is N. By combining this with Equations (2.2) and (2.3), we obtain

$$\begin{aligned} c &= \mathcal{E}_{g_2}([c]_{g_2}, r_2) \\ &= g_1^{[c]_{g_2}[g_2]_{g_1}} r_3^N r_2^N \mod N^2 \\ &= \mathcal{E}_{g_1}([c]_{g_2}[g_2]_{g_1} \mod N, r_2 r_3 \mod N) \\ &= \mathcal{E}_{g_1}([c]_{g_1}, r_1). \end{aligned}$$

The last two lines of above equations and the bijectivity of the encryption function indicate that $[c]_{g_1} = [g_2]_{g_1}[c]_{g_2} \mod N$, which concludes the proof. □

Corollary 2.1. *For any $g_1, g_2 \in \mathcal{B}_1$, $[g_1]_{g_2} = [g_2]_{g_1}^{-1}$.*

Proof. By setting $c = g_1$ in Lemma 2.3, we have

$$[g_1]_{g_1} = [g_2]_{g_1}[g_1]_{g_2} \mod N.$$

Noticing $\mathcal{E}_{g_1}(1, 1) = g_1^1 1^N = g_1$, we have $[g_1]_{g_1} = 1$, which concludes the statement. □

Lemma 2.4. *For any $c \in \mathbb{Z}_{N^2}^*$, $L(c^\lambda \mod N^2) = \lambda[c]_{1+N} \mod N$.*

Proof. Since $c \in \mathbb{Z}_{N^2}^*$ and $1 + N \in \mathcal{B}_1$, there exist m $(= [c]_{1+N}) \in \mathbb{Z}_N$ and $r \in \mathbb{Z}_N^*$ such that

$$(1+N)^m r^N = c \mod N^2$$

hold. By taking the λ-th power of the above equation, we have

$$\begin{aligned} c^\lambda &= (1+N)^{m\lambda} r^{N\lambda} \mod N^2 \\ &= (1+N)^{m\lambda} \mod N^2 \\ &= 1 + Nm\lambda \mod N^2, \end{aligned}$$

which yields the statement in Lemma 2.4. □

By using these formulae, the decryption process works as follows:

$$\begin{aligned} \frac{L(c^\lambda \mod N^2)}{L(g^\lambda \mod N^2)} \mod N &= \frac{\lambda[c]_{1+N}}{\lambda[g]_{1+N}} \mod N \text{ (Lemma 2.4)} \\ &= \frac{[c]_{1+N}}{[g]_{1+N}} \mod N \\ &= [c]_{1+N}[1+N]_g \mod N \text{ (Corollary 2.1)} \\ &= [c]_g \mod N \text{ (Lemma 2.3).} \end{aligned}$$

This shows that the decryption process can recover the original plaintext by using the public key $\mathsf{pk} = (N, g)$ and the secret key $\mathsf{sk} = \lambda$.

2.3.4 Security

The security of the Paillier cryptosystem relies on the intractability hypothesis of the Composite Residuosity Class Problem, which is called the Decisional Composite Residuosity Assumption (DCRA). DCRA roughly states that it is hard to decide a presented element is the N-th residue or not, where the N-th residuosity is defined as Definition 2.9.

Definition 2.9 (N-th residuosity)**.** *A number $z \in \mathbb{Z}_{N^2}^*$ is said to be an N-th residue modulo N^2 if there exists a number $y \in \mathbb{Z}_{N^2}^*$ such that*

$$z = y^N \mod N^2.$$

We define an adversary $\mathcal{A}(N, r)$ assumed in DCRA as follows. Let n be a security parameter, and let $N = pq$ where p and q are n-bit odd primes. The adversary is given N and a random r, which can be either a random over $\mathbb{Z}_{N^2}^*$ (*i.e.*, $r \leftarrow \mathbb{Z}_{N^2}^*$) or the N-th power of a random over $\mathbb{Z}_{N^2}^*$ (*i.e.*, $r = r'^N$

mod N^2 where $r' \leftarrow \mathbb{Z}^*_{N^2}$). The adversary is allowed to use any probabilistic polynomial-time algorithm to guess whether r is the N-th residuosity or not. The output of the adversary is 1 if the adversary guesses that r is the N-th residuosity, and 0 otherwise.

Let $\mathsf{GenModulus}$ be a polynomial-time algorithm that, given a security parameter n, outputs (N, p, q) where $N = pq$, and p and q are n-bit odd primes except with probability negligible in n. Then, DCRA is stated as Conjecture 2.1.

Conjecture 2.1 (Decisional composite residuosity assumption (DCRA))**.** *Let N be generated by $\mathsf{GenModulus}(1^n)$. The decisional composite residuosity assumption is that for all probabilistic polynomial-time algorithms $\mathcal{A}$, there exists a negligible function negl such that*

$$\left|\Pr[\mathcal{A}(N, r^N \mod N^2) = 1] - \Pr[\mathcal{A}(N, r) = 1]\right| \leq \mathsf{negl}(n),$$

where the probability is taken over the experiment in which (N, p, q) is generated by $\mathsf{GenModulus}(1^n)$, and a random $r \leftarrow \mathbb{Z}^*_{N^2}$ is chosen.

Given DCRA, the security of the Paillier cryptosystem is stated in Theorem 2.1.

Theorem 2.1 (Security of the Paillier cryptosystem)**.** *Assuming that DCRA holds, the Paillier cryptosystem is IND-CPA.*

Proof. Let n be a security parameter, let $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be the Paillier cryptosystem, let $\mathcal{A}$ be any probabilistic polynomial-time adversary, and let $\mathsf{PubK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n)$ be a CPA indistinguishability experiment associated with the Paillier cryptosystem. Let us consider the following experiment $D(N, y)$.

1. Challenger $\mathcal{C}$ runs $\mathsf{GenModulus}(n)$ to obtain (N, p, q), and sets $\mathsf{pk} = (N, N + 1)$.

2. Adversary $\mathcal{A}$ is given the public key pk, and oracle access to $\mathsf{Enc}_{\mathsf{pk}}(\cdot)$. Adversary $\mathcal{A}$ generates a pair of messages m_0 and m_1 from $\mathbb{Z}_N$, and sends them to challenger $\mathcal{C}$.

3. Challenger $\mathcal{C}$ chooses a uniformly-random bit $b \in \{0, 1\}$, sets

$$c := (1 + N)^{m_b} \cdot y \mod N^2,$$

   and sends c to adversary $\mathcal{A}$.

4. Adversary $\mathcal{A}$ guesses whether $m_b = m_0$ or $m_b = m_1$ without knowing the random bit b, and sends his/her guess bit $b' \in \{0, 1\}$ to challenger $\mathcal{C}$.

5. Challenger $\mathcal{C}$ outputs 1 if $b' = b$, and 0 otherwise, which is the output of the experiment.

We analyze the experiment in two cases where (i) y is the N-th power of a random over $\mathbb{Z}_{N^2}^*$ or (ii) y is a random over $\mathbb{Z}_{N^2}^*$.

(i) y is the N-th power of a random from $\mathbb{Z}_{N^2}^*$
Let $y = r^N \mod N^2$ where $r \leftarrow \mathbb{Z}_{N^2}^*$. Since the view of adversary $\mathcal{A}$ is the same as the adversary in the CPA indistinguishability experiment of the Paillier cryptosystem, it holds that

$$\Pr[D(N, r^N) = 1] = \Pr[\mathsf{PubK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n) = 1].$$

(ii) y is a random from $\mathbb{Z}_{N^2}^*$
Since y is uniform-randomly distributed in $\mathbb{Z}_{N^2}^*$, the challenge ciphertext $c = (1+N)^{m_b} \cdot y \mod N^2$ is also uniform-randomly distributed in $\mathbb{Z}_{N^2}^*$, independent of m_b (see, Lemma 2.1). Therefore, adversary $\mathcal{A}$ cannot but guess the bit uniform-randomly, *i.e.*,

$$\Pr[D(N, y) = 1] = \frac{1}{2}$$

holds.

Since DCRA states that for all probabilistic polynomial-time algorithms $\mathcal{A}'$, there exists a negligible function negl satisfying

$$\left|\Pr[\mathcal{A}'(N, r^N \mod N^2) = 1] - \Pr[\mathcal{A}'(N, r) = 1]\right| \leq \mathsf{negl}(n),$$

the probabilistic polynomial-time algorithm D must also satisfy the inequality above, implying that

$$\begin{aligned}
&\left|\Pr[D(N, r^N \mod N^2) = 1] - \Pr[D(N, r) = 1]\right| \\
=&\left|\Pr[\mathsf{PubK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n) = 1] - \frac{1}{2}\right| \\
\leq&\mathsf{negl}(n),
\end{aligned}$$

which concludes the proof. □

2.3.5 Computation Time

Finally, we empirically examine the computation time for atomic cryptographic operations of the Paillier cryptosystem: key generation, encryption,

Table 2.1: Computation time of basic operations of the Paillier cryptosystem.

	Time [ms]
Key generation T_{keygen}	45.2
Encryption T_{enc}	10.14
Decryption T_{dec}	9.63
Addition T_{add}	0.01
Subtraction T_{sub}	0.52

decryption, secure addition, and secure subtraction. We implemented the Paillier cryptosystem in Java 1.8.0_45 and ran the algorithms on a Mac laptop with 2.6 GHz Intel Core i5 and 8 GB 1600 MHz DDR3 memory. We fix the key length[2] $k = 1024$ bits. We repeatedly execute the algorithms 100 times and report their mean values. The experimental results are summarized in Table 2.1. These results will be used in Chapters 3 and 5 to estimate the computation time of the protocols presented therein. Since the computation time of addition is much smaller than the other operations, we ignore it in the computation time estimation.

2.4 Privacy Assumptions in Crowdsourcing

We conclude the preliminary section by defining the behavior of the entities during executing a protocol and discussing the validity of the assumptions.

We assume that all entities (the platform, requesters, and workers) are semi-honest and non-collusive, *i.e.*, they follow a protocol and do not share their records they obtain during the protocol, but they may try to infer other entities' sensitive information using their own records. We also assume that all the communication channels are encrypted to ensure that an attacker cannot eavesdrop the communication between any pair of entities.

The semi-honest assumption is reasonable assuming that the main purpose of all the entities using crowdsourcing complies with the model defined in Section 1.5.2. If an entity did not follow the protocol, the result of a protocol would be meaningless, which would be inconvenient for all of those involved. For example, in Chapter 3, the result of a protocol is a task assignment, and if some entity does not follow the protocol, the resultant task assignment can be infeasible. In Chapter 4, the result of a protocol is task results. If task results are corrupted, the requester does not benefit from them at all, and workers will not be able to receive the reward. In Chapter 5, the

[2]In our case, the key length corresponds to the bit length of prime numbers p and q.

result of a protocol is a set of aggregated labels, which can be corrupted if some entity does not follow the protocol. As a result, the requester cannot obtain his/her target, the platform can lose faith from other entities, and workers will not be rewarded.

The non-collusion assumption can be validated as follows. The platform does not collude to any other entities because if the platform would try to collude, but fail, the platform would develop a bad reputation that would spread to all the entities, and the platform would suffer from it severely. Other two entities cannot collude if they do not know each other in reality, because the communication between them is defined by a software performing a protocol, and no direct communication path exists. We believe it is also possible to prevent two entities from sharing their information by engineering a software to run a protocol.

Chapter 3

Privacy-Preserving Task Assignment

3.1 Introduction

Many existing platforms are currently dealing with simple tasks, which do not require workers to have special features such as skills and other attributes, Recently, there has been growing interest in how to crowdsource feature-dependent tasks, which can only be processed only by workers with specific features. For example, an English-French translation task requires both English and French skills. As another example, let us consider a task involving reporting whether an indicated building has any damage after a disaster. In this case, it would be difficult for a worker far from the building to complete the task, whereas a worker based closer to the building would be able to perform the task (in this example, the worker's location corresponds to the feature). Since feature-dependent tasks include a number of practical tasks, an increasing number of studies have been conducted to enable us to handle such tasks to be easily processed via crowdsourcing.

One of the vital challenges a crowdsourcing platform specializing in feature-dependent tasks needs to overcome is to improve the throughput of the platform. The current practice of using open call assignments is not appropriate for this purpose in that a worker with a special skill may choose a simple task; thus a feature-dependent task requiring this worker's skill remains unassigned, which can be a sub-optimal assignment. An obvious approach to addressing this issue would be to compute optimal task assignment based on the feature sets of the workers and the feature requirements of the tasks. A platform with such knowledge would be able to globally maximize the throughput of its crowdsourcing system.

In this study, we first point out the privacy issues affecting both the workers and requesters in the task assignment strategy mentioned above. Workers are requested to report their skills such as language abilities and programming skills as well as their attributes such as their locations, minimum wages, and working hours, and they may even have to disclose additional information of a more personal nature. These features can be used to identify the workers, infer sensitive information about them, expose them to physical danger by revealing their location information, and introduce unfairness by excessively favoring highly skilled and hardworking workers. Furthermore, the privacy of requesters will also be compromised. Requesters have to report the corresponding feature requirements, which can be used to identify them and to reveal the contents of their tasks. Although several previous studies have investigated privacy issues surrounding task assignments (Kazemi and Shahabi, 2011, 2012b; To et al., 2014), their applicability is limited because (i) they focus on location-based tasks and (ii) they aim to preserve the privacy of workers only. This indicates the need for the development of a task assignment system for *general* tasks in which the privacy of *both* workers and requesters is preserved.

We address the privacy issues by developing a private task assignment (PTA) protocol. We set our goal at maximizing the number of feasible tasks without asking each entity to disclose their features. We first note that the task assignment problem without privacy requirements is reduced to a maximum flow problem. Given this observation, PTA first constructs an assignment network in which a maximum flow coincides with an optimal assignment. Then, PTA computes the maximum flow on the assignment network to derive the optimal task assignment. We utilize the Paillier cryptosystem (Paillier, 1999) to execute these two steps in a private way. Based on the security offered by the Paillier cryptosystem, after execution of PTA, the platform learns the optimal task assignment only, whereas none of the other entities learn anything.

PTA has three main advantages compared to the previous approaches. First, PTA outputs an optimal assignment without sacrificing the privacy guarantee because it employs a cryptographic approach, rather than a perturbation approach (Kazemi and Shahabi, 2011, 2012a; To et al., 2014), to guarantee privacy preservation. Therefore, we do not have to be concerned about the privacy-utility trade-off of the perturbation approaches. Second, PTA provides a practical cryptographic role assignment strategy. Although most of the existing cryptographic studies do not specify how to assign roles, we devise a general strategy to prepare the parties who play the roles so as to elaborate the ability of PTA to be practical in crowdsourcing. The key idea is to crowdsource some of the cryptographic roles of PTA without compromis-

ing any privacy. Third, we dedicate a privacy-preserving maximum flow part of PTA specifically to task assignment setting, which is theoretically proven to run eight times faster than the existing privacy-preserving maximum flow protocol (Aly et al., 2013).

In summary, this research makes three main contributions. First, we point out the privacy issues of both workers and requesters in a crowdsourcing task assignment environment. Second, we develop a cryptographic solution called PTA in which neither privacy nor accuracy is sacrificed. Third, we provide a general strategy to assign cryptographic roles, which are mandatory to deploy PTA in a real setting.

3.2 Private Task Assignment Problem

We first specify our crowdsourcing model based on the abstraction of crowdsourcing introduced in Section 1.5.2 and then formulate the private task assignment problem, which is our main concern.

3.2.1 Crowdsourcing Model

Entities

As described in Section 1.5.2, our model consists of three types of entities: requesters, workers, and a platform. We provide details about them that are not described in Section 1.5.2 in the following paragraphs.

A requester uses crowdsourcing to have a task processed, which consists of a job instruction and multiple instances. Assume for simplicity that each requester submits one task, *i.e.*, each task is associated with one requester. Let $\mathcal{T} = \{t_i \mid i \in [I]\}$ be a set of tasks, where t_i is a task ID. We abuse the notation to represent the requester who has task t_i as requester t_i. In Figure 3.1, task t_1 is an English-German translation task, and task t_2 is an English-French translation task, in which an instance corresponds to an English text. We further assume that requester t_i has two parameters: requirement vector $\mathbf{r}_i \in \{0,1\}^D$ ($D \geq 1$) and capacity $L_i \in \mathbb{Z}_+$. The d-th dimension of $\mathbf{r}_i$ indicates whether task t_i requires feature d for completion ($r_{i,d} = 1$) or not ($r_{i,d} = 0$). In Figure 3.1, dimensions 1, 2, and 3 are the requirements of the English, French, and German skills, respectively. Since task t_1 is an English-German translation task, $\mathbf{r}_1 = \begin{bmatrix}1 & 0 & 1\end{bmatrix}$ holds. Capacity L_i is the number of instances of task t_i, indicating that the requester is willing to process L_i instances. Let us denote the set of all the parameters of the tasks by $\mathcal{P}_{\mathcal{T}} = \{(t_i, \mathbf{r}_i, L_i) \mid i \in [I]\}$.

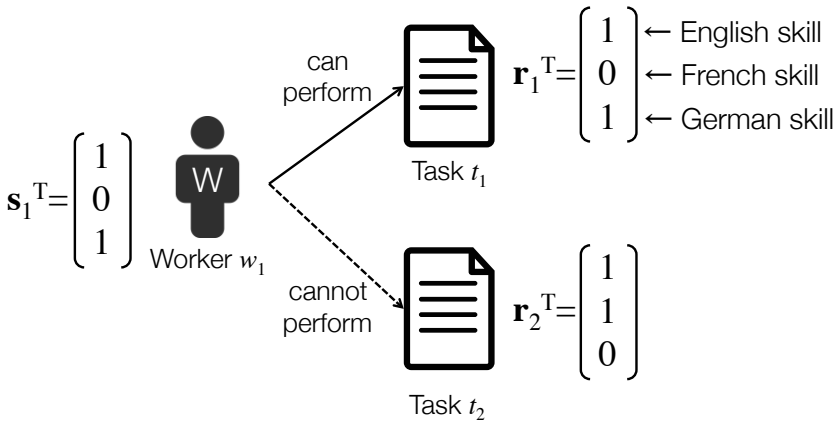


Figure 3.1: Illustration of task feasibility. Worker w_1 with both English and German skills can complete task t_1, but cannot complete task t_2, which requires French skills.

A worker performs the assigned tasks in exchange for reward. For example, if a worker is assigned the translation task from English to German, the worker receives English sentences from the requester, translates them into German sentences, and sends them back to the requester to be rewarded. Let $\mathcal{W} = \{w_j \mid j \in [J]\}$ be a set of workers, where w_j is a worker ID. Assume that each worker w_j is associated with two parameters: feature vector $\mathbf{s}_j \in \{0,1\}^D$ and capacity $M_j \in \mathbb{Z}_+$. The d-th element of $\mathbf{s}_j$ indicates whether worker w_j has feature d ($s_{j,d} = 1$) or not ($s_{j,d} = 0$). In Figure 3.1, worker w_1 is fluent in English and German, but is not familiar with French. Note that requirement and feature vectors share the semantics of each dimension. For example, the first dimension of both vectors indicates the English skill in Figure 3.1. Capacity M_j is the maximum number of tasks that the worker can afford to accept. Let us denote the set of parameters of all the workers by $\mathcal{P}_{\mathcal{W}} = \{(w_j, \mathbf{s}_j, M_j) \mid j \in [J]\}$.

Task Feasibility

Task feasibility is defined by the parameters of the tasks and the workers. We assume that worker w_j can complete task t_i if $\mathbf{s}_j \geq \mathbf{r}_i$ holds element-wise and if capacity conditions hold. In Figure 3.1, worker w_1 can complete task t_1, but cannot complete task t_2, because worker w_1 does not have the French skill. Definition 3.1 gives the definitions of a task assignment and a feasible task assignment.

Definition 3.1 (Task assignment)**.** *Task assignment of size K is defined as a multiset $\mathcal{A} := \{(w_{j_k}, t_{i_k}) \mid k \in [K], i_k \in [I], j_k \in [J]\}$. If $\mathcal{A}$ satisfies*

1. $|\{k \mid i_k = i\}| \leq L_i$ *for all* $i \in [I]$,

2. $|\{k \mid j_k = j\}| \leq M_j$ *for all* $j \in [J]$,

3. $\mathbf{s}_{j_k} \geq \mathbf{r}_{i_k}$ *for all* $k \in [K]$,

then the task assignment is referred to as a feasible task assignment.

Non-binary and Missing Features

Here we discuss how we enable our model to address non-binary and missing features.

Non-binary features. Non-binary features can largely be converted into binary features. We demonstrate this by converting two popular non-binary features into binary features.

(i) A real number

Some may be willing to represent a feature by a real number instead of a binary number to represent the degree of expertise. For example, since a language skill can be measured by assigning a score based on a test, this demands the use of real numbers for features. Let us denote the degree of a skill of worker w_j by $x_j \in [0, 1]$. Then, quantizing $[0, 1]$ into multiple bins (four bins in this example), the continuous skill can be represented as follows:

$$\begin{aligned} s_{j,1} &= 1 \text{ if } x_j \in [0, 1.0], \\ s_{j,2} &= 1 \text{ if } x_j \in [0.25, 1.0], \\ s_{j,3} &= 1 \text{ if } x_j \in [0.5, 1.0], \\ s_{j,4} &= 1 \text{ if } x_j \in [0.75, 1.0]. \end{aligned}$$

For example, a worker with skill $x_j = 0.7$ sets his/her feature vector $\mathbf{s}_j = \begin{bmatrix} 1 & 1 & 1 & 0 \end{bmatrix}$. A requester r_i who wants to hire a worker with skill $x_j \geq 0.5$ only has to set his/her requirement vector as

$$\begin{bmatrix} r_{i,1} & r_{i,2} & r_{i,3} & r_{i,4} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}.$$

(ii) Location information

The location information can also be encoded into binary features by quantization. Assume that a target area (*e.g.*, a country and a city) is divided into N districts. Then, by assigning the d-th dimension of

the feature vector to the d-th district, the feature vector of worker w_j is composed as

$s_{j,d} = 1$ if worker w_j can go to the d-th district for work ($\forall d \in [N]$).

A requester r_i who is willing to submit a task associated with the l-th district only has to set his/her requirement vector as

$$r_{i,d} = \begin{cases} 1 & \text{if } d = l, \\ 0 & \text{if } d \neq l. \end{cases}$$

Missing features. Although a real crowdsourcing platform such as Upwork provides us with an explicit feature representation of a worker's skills, the list of displayed skills may be incomplete, either in part or in total. One approach to overcoming such incomplete or missing features is to estimate them using workers' work histories. Given results returned by multiple workers processing the same task, a series of quality control methods (Dawid and Skene, 1979; Whitehill et al., 2009; Welinder et al., 2010; Lease, 2011) allow us to infer the ability of each worker to carry out the task, x_j. Among a number of quality control methods, a method developed by Kajino *et al.* (Kajino et al., 2014a) is appropriate for this purpose because it can execute the above inference privately. By applying the conversion technique above, the estimated real-valued ability can be represented by a binary feature vector.

3.2.2 Problem Setting

A private task assignment problem (Problem 3.1) aims to obtain a maximum feasible assignment without revealing the parameters of each worker and requester under the privacy assumptions in crowdsourcing (Section 2.4). A task assignment problem refers to the same problem setting without the privacy requirements.

Problem 3.1 (Private task assignment problem)**.** *Given $\mathcal{P}_\mathcal{T}$ and $\mathcal{P}_\mathcal{W}$, the private task assignment problem entails obtaining a maximum feasible assignment between tasks and workers without allowing any entity to infer the parameters of the others, under the privacy assumptions in Section 2.4.*

3.3 Solution in a Non-Private Setting

In this section, we present a solution to the task assignment problem, a non-privacy-preserving variant of our problem setting. Our approach reduces the

task assignment problem to a maximum flow problem and solves it by applying the push-relabel algorithm (Goldberg and Tarjan, 1988). The solution described in this section is used as a basis for developing PTA.

3.3.1 Maximum Flow Problem

Let $N = (V, E, C)$ be a network, where V is a set of vertices including source s and sink t, E is a set of directed edges, and C is a set of capacities. Capacity $c_{u,v} \in \mathbb{Z}_+$ is the upper limit of a flow from u to v ($u, v \in V$). If $(u, v) \notin E$, we set $c_{u,v} = 0$. A flow on a network is defined as Definition 3.2. Note that we consider an integer flow in this chapter.

Definition 3.2 (Flow)**.** *Flow F on network $N = (V, E, C)$ is a set of integers $\{f_{u,v} \in \mathbb{Z}_+ \mid (u, v) \in V \times V\}$ satisfying the following conditions:*

1. $f_{u,v} \leq c_{u,v}$, $\forall (u, v) \in V \times V$,
2. $f_{u,v} = -f_{v,u}$, $\forall (u, v) \in V \times V$,
3. $\sum_{u:(u,v)\in E} f_{u,v} = \sum_{u:(v,u)\in E} f_{v,u}$, $\forall v \in V \backslash \{s, t\}$.

The value of flow F is defined as $|F| = \sum_{v:(s,v)\in E} f_{s,v}$.

Then, the maximum flow problem is defined as Problem 3.2.

Problem 3.2 (Maximum flow problem)**.** *Given network N, the maximum flow problem is to obtain flow F whose value is maximal.*

3.3.2 Reduction to a Maximum Flow Problem

We reduce the task assignment problem to a maximum flow problem by constructing assignment network $N = (V, E, C)$ such that a maximum flow on N coincides with a maximum task assignment. The following three steps describe how to construct an assignment network given an instance of the task assignment problem $(\mathcal{P}_\mathcal{T}, \mathcal{P}_\mathcal{W})$.

1. Let the set of vertices be $V := \{s, t\} \cup \mathcal{T} \cup \mathcal{W}$, where s is the source and t is the sink.
2. Let the set of edges be $E = E_1 \cup E_2 \cup E_3$, in which we define

$$
\begin{aligned}
E_1 &:= \{(s, w_j) \mid w_j \in \mathcal{W}\}, \\
E_2 &:= \{(w_j, t_i) \mid w_j \in \mathcal{W},\ t_i \in \mathcal{T}\}, \\
E_3 &:= \{(t_i, t) \mid t_i \in \mathcal{T}\}.
\end{aligned}
$$

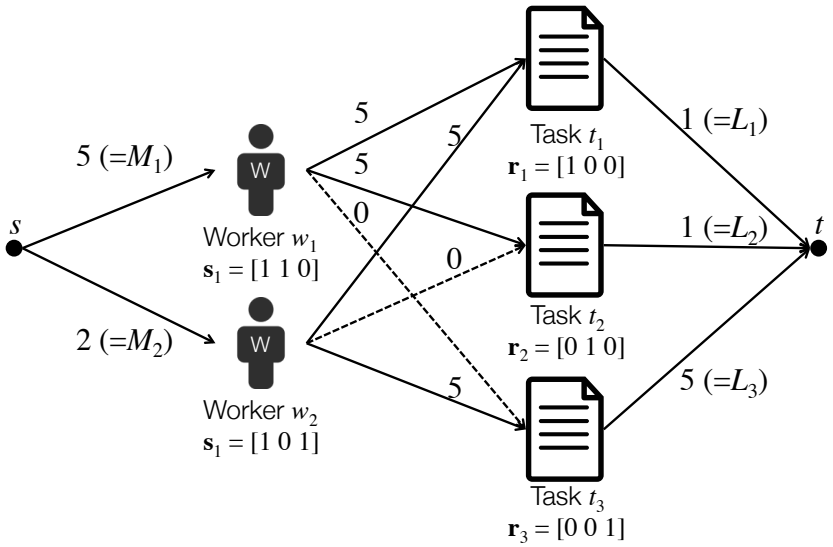


Figure 3.2: Network representation of the task assignment problem instance, which we refer to as an assignment network.

3. We set the capacity of each edge $(s, w_j) \in E_1$ as $c_{s,w_j} := M_j$, the capacity of each edge $(w_j, t_i) \in E_2$ as

$$c_{w_j,t_i} := \begin{cases} M_\top & (\mathbf{s}_j \geq \mathbf{r}_i), \\ 0 & (\mathbf{s}_j < \mathbf{r}_i), \end{cases}$$

where $M_\top = \max\{\max_{i\in[I]} L_i, \max_{j\in[J]} M_j\}$, and the capacity of each edge $(t_i, t) \in E_3$ is $c_{t_i,t} := L_i$. Let the set of all the capacities be C.

Figure 3.2 illustrates the assignment network construction. Suppose that there are three tasks $\mathcal{T} = \{t_1, t_2, t_3\}$ with the following parameters,

$$\mathbf{r}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \mathbf{r}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}, \mathbf{r}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}, (L_1, L_2, L_3) = (1, 1, 5),$$

and that there are two workers $\mathcal{W} = \{w_1, w_2\}$ with the following parameters,

$$\mathbf{s}_1 = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}, \mathbf{s}_2 = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}, (M_1, M_2) = (5, 2).$$

We first set edges from source s to workers w_1 and w_2 with capacities M_1 and M_2, respectively. Then, we set edges from each worker to each task with capacity $M_\top$ $(= 5)$ if the worker has all the features required by the task; otherwise, the capacity is set to 0. Finally, we set edges from tasks t_1, t_2, and t_3 to sink t with capacities L_1, L_2, and L_3, respectively.

Algorithm 3.1 Extracting an assignment from a flow.

Input: integer flow F.
Output: assignment $\mathcal{A}$.

1: $\mathcal{A} \leftarrow \emptyset$
2: **for** each $w_j \in \mathcal{W}$ and $t_i \in \mathcal{T}$ **do**
3: **if** $f_{w_j,t_i} > 0$ **then**
4: $\mathcal{A} \leftarrow \mathcal{A} \cup \{(w_j, t_i)\}^{f_{w_j,t_i}}$
5: **end if**
6: **end for**
7: **return** $\mathcal{A}$

A maximum assignment of $(\mathcal{P}_\mathcal{T}, \mathcal{P}_\mathcal{W})$ can be derived from a maximum flow $F^\star$ on the corresponding assignment network using Algorithm 3.1. Proposition 3.1 claims the correspondence. The proof of Proposition 3.1 appears in Appendix A.1.

Proposition 3.1. *Given a problem instance of the task assignment problem $(\mathcal{P}_\mathcal{T}, \mathcal{P}_\mathcal{W})$, the corresponding assignment network N, and a maximum flow $F^\star$ on N, a solution to the task assignment problem is given by transforming the maximum flow into a feasible assignment using Algorithm 3.1.*

3.3.3 Push-Relabel Algorithm

The push-relabel algorithm (Goldberg and Tarjan, 1988) is a variant of maximum flow algorithms. It serves as an essential building block of our private task assignment protocol. The push-relabel protocol updates two variables, a preflow and heights, using the push and relabel operations. We first introduce these internal variables and operations, and then, describe the generic push-relabel algorithm.

Internal Variables

A preflow (Definition 3.3) is a relaxed flow in which for each vertex, the total flow into the vertex can exceed the total flow out of the vertex.

Definition 3.3 (Preflow)**.** *Preflow F on network $N = (V, E, C)$ is a set of integers $\{f_{u,v} \in \mathbb{Z}_+ \mid (u, v) \in V \times V\}$ satisfying the following three conditions:*

1. $f_{u,v} \leq c_{u,v}$, $\forall (u, v) \in V \times V$,
2. $f_{u,v} = -f_{v,u}$, $\forall (u, v) \in V \times V$,

Operation 3.2 $\mathsf{Push}(v, w)$.

1: **if** v is active, $(v, w) \in E_F$, and $h_w = h_v - 1$ **then**
2: $\quad \delta \leftarrow \min\{e_v, c_{v,w} - f_{v,w}\}$.
3: $\quad f_{v,w} \leftarrow f_{v,w} + \delta$, $f_{w,v} \leftarrow f_{w,v} - \delta$, $e_v \leftarrow e_v - \delta$, and $e_w \leftarrow e_w + \delta$.
4: **end if**

3. $\sum_{u:(u,v)\in E} f_{u,v} \geq \sum_{u:(v,u)\in E} f_{v,u}$, $\forall v \in V \backslash \{s, t\}$.

Excess e_v of vertex $v \in V$ is the amount of overflow at vertex v, *i.e.*,

$$e_v := \sum_{u \in V \backslash \{v\}} f_{u,v}.$$

A set of excesses is denoted by $\mathbf{e} \in \mathbb{Z}_+^{|V|}$. If F is a preflow, $e_v \geq 0$ holds for every vertex $v \in V \backslash \{s\}$. Vertex $v \in V \backslash \{s, t\}$ is active if $e_v > 0$, indicating vertex v is overflowing. Edge $(u, v) \in V \times V$ is residual if $c_{u,v} - f_{u,v} > 0$. The set of residual edges is denoted by E_F.

Height h_v of vertex $v \in V$ is a non-negative integer label, managing the applicability of push and relabel operations. A set of heights is denoted by $\mathbf{h} \in \mathbb{Z}_+^{|V|}$. Heights on network N with preflow F are valid if they satisfy the following three conditions:

1. $h_u \leq h_v + 1$ for each residual edge $(u, v) \in E_F$,
2. $h_s = |V|$,
3. $h_t = 0$.

While in operation, the algorithm maintains the heights such that they are valid. Intuitively, the valid heights retain approximate distances to the sink or source. If $h_v < |V|$, h_v is a lower bound on the distance from v to the sink in the residual network, and if $h_v \geq |V|$, $h_v - |V|$ is a lower bound on the distance from v to the source in the residual network. The algorithm harnesses the heights to distribute the overflow on an active vertex to the neighboring vertex that seems to be the closest to the sink or the source.

Push and Relabel Operations

A push operation $\mathsf{Push}(v, w)$ $(v, w \in V)$ transfers the overflow on vertex v to vertex w as much as possible. Since it transfers the overflow on vertex v, vertex v must be active ($e_v > 0$), and edge (v, w) must be residual ($c_{v,w} - f_{v,w} > 0$). In addition, the height condition, $h_w = h_v - 1$, must be satisfied;

Operation 3.3 Relabel(v).

1: **if** v is active, and $h_v \leq h_w$ holds for all $w \in V$ such that $(v, w) \in E_F$ **then**
2: $\quad h_v \leftarrow \min\{h_w + 1 \mid (v, w) \in E_F\}$
3: **end if**

the algorithm tries to transfer the overflow to the source or the sink through the shortest path. Such vertex w is expected to be the shortest because the validity of the heights ensures that vertex u such that $h_u < h_v - 1$ does not exist. Operation 3.2 describes the push operation.

A relabel operation Relabel(v) ($v \in V$) increases the height of vertex v as much as possible when the push operation cannot be applied to the vertex because of the height condition. Application of the relabel operation requires vertex v to be active and that for all residual edges $(v, w) \in E_F$, $h_v \leq h_w$ holds. Operation 3.3 describes the relabel operation.

Generic Algorithm

The generic push-relabel algorithm first initializes the preflow and the heights as

$$\begin{aligned} f_{s,v} &\leftarrow c_{s,v}, f_{v,s} \leftarrow -c_{v,s}\ (\forall v \in V), \\ f_{u,v} &\leftarrow 0\ (\forall u, v \in V \backslash \{s\}), \\ h_s &\leftarrow |V|, h_v \leftarrow 0\ (\forall v \in V \backslash \{s\}), \end{aligned}$$

and initializes the excess using the preflow. It then repeatedly applies the push and relabel operations in an arbitrary order until there exists no active vertex. The resultant preflow is proven to be a maximum flow (Goldberg and Tarjan, 1988).

3.4 Cryptographic Building Blocks

Next, we introduce cryptographic building blocks of PTA. All the cryptographic building blocks are based on the Paillier cryptosystem (Paillier, 1999), which has been introduced in Section 2.3. In order to implement PTA, we newly develop a conditional test, which executes an if-else statement of the algorithm in a privacy-preserving way.

Table 3.1: Private inputs and outputs of the conditional test COND.

Party	**Private input**	**Private output**
Decryptor	sk	–
Operator	$\mathsf{Enc}(m_1)$, $\mathsf{Enc}(m_2)$, $\mathsf{Enc}(c)$	$\mathsf{Enc}(m_1)$ if $c > 0$, $\mathsf{Enc}(m_2)$ otherwise.
Mixer	–	–

3.4.1 Data Structure

An encrypted network (Definition 3.4) stores the encryption of a network and a flow using two matrices of size $|V| \times |V|$. In particular, the encryption of an assignment network with a flow is called an encrypted assignment network. It represents the topological structure of a network by the capacities of edges; a zero-capacity edge indicates that the edge does not exist. Therefore, without the secret key, it neither leaks any information about the structure of the network nor about its capacities, except for the size of the network.

Definition 3.4 (Encrypted network)**.** *Encrypted network* $\mathsf{Enc}(N)$ *is defined as* $\mathsf{Enc}(N) := (\mathsf{Enc}(\mathbf{C}), \mathsf{Enc}(\mathbf{F}))$*, where* $\mathsf{Enc}(\mathbf{C}) := \left[\mathsf{Enc}(c_{u,v})\right]_{u,v \in V}$*, and* $\mathsf{Enc}(\mathbf{F}) := \left[\mathsf{Enc}(f_{u,v})\right]_{u,v \in V}$ *are encryptions of the capacities and flow.*

3.4.2 Conditional Test

We present a *conditional test* as a key sub-protocol of PTA, which outputs one of two ciphertexts based on the encrypted condition without leaking any information. Table 3.1 summarizes the parties and their private inputs and outputs. Assume that there are three parties, an operator, a decryptor, and a mixer. The operator has three ciphertexts $\mathsf{Enc}(m_1)$, $\mathsf{Enc}(m_2)$, and $\mathsf{Enc}(c)$, and only the decryptor retains the secret key. The private output of the operator after executing the conditional test is

$$\mathsf{COND}(\mathsf{Enc}(m_1), \mathsf{Enc}(m_2), \mathsf{Enc}(c)) = \begin{cases} \mathsf{Enc}(m_1) & (c > 0), \\ \mathsf{Enc}(m_2) & (c \leq 0). \end{cases}$$

The conditional test can be used to implement a *min protocol* MIN:

$$\mathsf{MIN}(\mathsf{Enc}(m_1), \mathsf{Enc}(m_2)) = \begin{cases} \mathsf{Enc}(m_2) & (m_1 > m_2), \\ \mathsf{Enc}(m_1) & (m_1 \leq m_2). \end{cases}$$

The operator only has to set $\mathsf{Enc}(c) := \mathsf{Enc}(m_2 - m_1)$ in the conditional test protocol. A *max protocol* MAX can be defined similarly. These protocols can

be extended to multiple ciphertexts $\{\mathsf{Enc}(m_1), \ldots, \mathsf{Enc}(m_l)\}$; the parties only have to repeat the protocol $l-1$ times.

In the following, we first review an inequality test protocol (Golle, 2006), which serves as a building block of COND, before presenting the detailed protocol of COND. Then, the correctness and security of COND are discussed.

Building Block: Inequality Test

The inequality test protocol INEQ (Golle, 2006) is executed by two parties: the operator and the decryptor. The private input of the operator consists of two ciphertexts $\mathsf{Enc}(m_1)$ and $\mathsf{Enc}(m_2)$, and that of the decryptor is secret key sk. We assume that they know that both plaintexts satisfy $0 \leq m_1, m_2 \leq L-1$ for some $L > 0$. The private output of the operator is defined as

$$\mathsf{INEQ}(\mathsf{Enc}(m_1), \mathsf{Enc}(m_2)) = \begin{cases} 1 & (m_1 > m_2), \\ 0 & (m_1 \leq m_2). \end{cases}$$

We review the procedure of INEQ intuitively. First, the operator creates an ordered set $S = \{\mathsf{Enc}(s_{\sigma(1)}(m_{\pi(1)} - m_{\pi(2)} - \sigma(1))), \ldots, \mathsf{Enc}(s_{\sigma(L-1)}(m_{\pi(1)} - m_{\pi(2)} - \sigma(L-1)))\}$ using random permutations $\sigma : [L-1] \to [L-1]$ and $\pi : [2] \to [2]$ and uniformly random variables $\{s_i\}_{i=1}^{L-1}$ chosen by the operator. Second, the operator sends S to the decryptor, and the decryptor decrypts the ciphertexts. If $m_{\pi(1)} > m_{\pi(2)}$ holds, the decryptor obtains one 0 and $L-2$ uniformly random variables from the decryptions; otherwise, the decryptor obtains $L-1$ uniformly random variables. The decryptor sends 1 to the operator if the decryptor obtains one 0; otherwise, s/he sends 0, and the operator computes his/her private output by using the message from the decryptor and π. After execution of the protocol, the decryptor learns nothing, and the operator only learns whether $m_1 > m_2$.

Note that the inequality test protocol is grounded on the equality test protocol EQTEST (Jakobsson and Schnorr, 1999; Lipmaa, 2003), which enables the parties to determine whether $m_1 = m_2$ holds without leaking any other information. By modifying the message sent by the operator in INEQ with $\mathsf{Enc}(s(m_1 - m_2))$, the parties learn whether $m_1 = m_2$. Since the inequality test protocol can be seen as $L-1$ repetitions of the equality test protocol, we employ the equality test protocol to estimate the computation time of the inequality test protocol in Section 3.4.3.

Protocol

The conditional test is implemented based on the inequality test as follows.

Table 3.2: Truth table of the conditional test in case of (a) $c > 0$ and (b) $c \leq 0$. The private output of the operator depends only on c, and is independent of permutation σ.

(a) $c > 0$

	$(\sigma(1), \sigma(2)) = (1, 2)$	$(\sigma(1), \sigma(2)) = (2, 1)$
a	1	0
Private output of the operator	$\mathsf{Enc}(m_{\sigma(1)}; r'_{\sigma(1),1}) = \mathsf{Enc}(m_1)$	$\mathsf{Enc}(m_{\sigma(1)}; r'_{\sigma(2),1}) = \mathsf{Enc}(m_1)$

(b) $c \leq 0$

	$(\sigma(1), \sigma(2)) = (1, 2)$	$(\sigma(1), \sigma(2)) = (2, 1)$
a	0	1
Private output of the operator	$\mathsf{Enc}(m_{\sigma(1)}; r'_{\sigma(2),1}) = \mathsf{Enc}(m_2)$	$\mathsf{Enc}(m_{\sigma(1)}; r'_{\sigma(1),1}) = \mathsf{Enc}(m_2)$

1. The operator creates an ordered set S of two vectors as follows and sends S to the mixer:

   $$S = \left\{ \left[\mathsf{Enc}(m_1; r_{1,1}) \quad \mathsf{Enc}(l_1; r_{1,2}) \right], \left[\mathsf{Enc}(m_2; r_{2,1}) \quad \mathsf{Enc}(l_2; r_{2,2}) \right] \right\},$$
   where $l_1 = 2c - 1,\ l_2 = 0$.

2. The mixer, receiving S, creates a shuffled and re-encrypted ordered set S' as follows and sends S' to the operator:

   $$S' = \left\{ \left[\mathsf{Enc}(m_{\sigma(1)}; r'_{\sigma(1),1}) \quad \mathsf{Enc}(l_{\sigma(1)}; r'_{\sigma(1),2}) \right], \left[\mathsf{Enc}(m_{\sigma(2)}; r'_{\sigma(2),1}) \quad \mathsf{Enc}(l_{\sigma(2)}; r'_{\sigma(2),2}) \right] \right\},$$

   where $\sigma : [2] \to [2]$ is a random permutation.

3. The operator and decryptor compute

   $$a \leftarrow \mathsf{INEQ}\left(\mathsf{Enc}(l_{\sigma(1)}; r'_{\sigma(1),2}), \mathsf{Enc}(l_{\sigma(2)}; r'_{\sigma(2),2}) \right).$$

4. The operator keeps $\mathsf{Enc}(m_{\sigma(1)}; r'_{\sigma(1),1})$ if $a = 1$ and keeps $\mathsf{Enc}(m_{\sigma(2)}; r'_{\sigma(2),1})$ if $a = 0$, which is the private output of the operator.

Table 3.3: Computation time of the cryptographic building blocks of PTA. L is the maximum of a plaintext, *i.e.*, a plaintext m must satisfy $0 \leq m \leq L-1$.

	Time [ms]
Equality test T_{EQTEST}	19.98
Inequality test T_{INEQ}	$19.98(L-1)$
Conditional test T_{COND}	$40.56 + 19.98(L-1)$
Min T_{MIN}	$41.08 + 19.98(L-1)$

Correctness and Security

The correctness of the protocol can be validated by referring to Table 3.2. The security of the protocol is stated as Proposition 3.2 based on the privacy definition in cryptography (Goldreich, 2004). Its proof is given in Appendix A.2.

Proposition 3.2 (Security of the conditional test)**.** *Let f_{COND} be a functionality whose inputs and outputs are defined as in Table 3.1. Assume that the Paillier cryptosystem is secure, i.e., the DCR assumption holds. Then, the conditional test protocol COND privately computes f_{COND}. In addition, the operator cannot distinguish his/her private output from a random ciphertext.*

3.4.3 Computation Time

We finally conclude this section by providing the computation time of the cryptographic building blocks, *i.e.*, EQTEST, INEQ, COND, and MIN. Since all of the building blocks depend on EQTEST and the basic operations of the Paillier cryptosystem, we first measure the computation time of EQTEST, and then, estimate the computation time of the other operations based on the computation time of EQTEST and the basic operations shown in Table 2.1. The results are summarized in Table 3.3.

Equality Test (EQTEST)

EQTEST receives two ciphertexts as its input and outputs whether they encrypt the same plaintext or not. The empirical estimate of its computation time is

$$T_{\mathsf{EQTEST}} = 19.98 \text{ ms},$$

which was evaluated in the same setting as Section 2.3.5.

Inequality Test (INEQ)

INEQ receives two ciphertexts and one plaintext L restricting the range of plaintexts and outputs which ciphertext is larger in the plaintext space. Given the computation time of EQTEST, we are able to estimate that of INEQ as

$$\begin{aligned} T_{\mathsf{INEQ}} &= (L-1)T_{\mathsf{EQTEST}} \\ &= 19.98(L-1) \text{ ms}, \end{aligned}$$

because INEQ executes EQTEST $L-1$ times.

Conditional Test (COND)

COND receives three ciphertexts $\mathsf{Enc}(m_1)$, $\mathsf{Enc}(m_2)$, and $\mathsf{Enc}(c)$ and outputs either $\mathsf{Enc}(m_1)$ or $\mathsf{Enc}(m_2)$ depending on whether $c > 0$ or not. Since COND consists of four encryptions (for re-encryption) and one INEQ, the computation time of COND is estimated as

$$\begin{aligned} T_{\mathsf{COND}} &= 4T_{\text{enc}} + T_{\mathsf{INEQ}} \\ &= 40.56 + 19.98(L-1) \text{ ms}. \end{aligned}$$

Min (MIN)

MIN receives two ciphertexts and outputs the smaller one, which is implemented with COND. The computation time is estimated as

$$\begin{aligned} T_{\mathsf{MIN}} &= T_{\mathsf{COND}} + T_{\text{sub}} \\ &= 41.08 + 19.98(L-1) \text{ ms}. \end{aligned}$$

3.5 Private Task Assignment (PTA) Protocol

We present our *private task assignment (PTA) protocol*, which solves Problem 3.1. Table 3.4 summarizes the private inputs and outputs. PTA consists of three parts.

1. The cryptosystem is initialized by the platform (who serves as a decryptor), and two cryptographic roles, an operator and a mixer, are recruited using crowdsourcing (Section 3.5.1).

2. The encrypted assignment network is constructed by all the parties (Section 3.5.2).

Table 3.4: Private inputs and outputs of PTA.

Party	**Private input**	**Private output**
Requester t_i	$\mathbf{r}_i$, L_i	–
Worker w_j	$\mathbf{s}_j$, M_j	–
Decryptor (=platform)	sk	a maximum assignment $\mathcal{A}^\star$
Operator	–	–
Mixer	–	–

3. The operator, the mixer, and the platform run a private push-relabel protocol to obtain a maximum flow, and the platform extracts a maximum task assignment from the maximum flow (Section 3.5.3).

Sections 3.5.1, 3.5.2, and 3.5.3 present the detailed protocols as well as the correctness of them, and Section 3.5.4 presents the security of the protocol.

Technical Contributions. Our protocol has three main technical contributions.

First, PTA is theoretically guaranteed to output an optimal task assignment to the crowdsourcing platform without any privacy invasion by making full use of cryptography. On the other hand, the existing task assignment methods for spatial crowdsourcing (Kazemi and Shahabi, 2011, 2012b; To et al., 2014) rely on perturbation approaches, and therefore, they do not usually output an optimal task assignment and/or preserve privacy perfectly.

Second, we invented a general strategy to assign cryptographic roles using crowdsourcing, taking the security of it into consideration. As a result, most of the entities are allowed to be offline in the main protocol. Contrary to this, a standard cryptographic study often dismisses such an assignment of roles despite its importance, because it is not easy to design the role assignment such that the semi-honest and non-collusive assumptions are reasonable.

Third, we theoretically prove that the private push-relabel part of PTA is eight times faster compared to the existing method (Aly et al., 2013), which is intended for general networks. The key idea is that PTA specializes in an assignment network only, rather than in general networks.

These contributions highlight the unique advantages of our protocol over existing task assignment protocols and the existing private push-relabel protocol.

Table 3.5: Private inputs and outputs of the initialization of PTA.

Party	Private input	Private output
Requester t_i	$\mathbf{r}_i$, L_i	$\mathbf{r}_i$, $\mathsf{Enc}(\mathbf{r}_i)$, $\mathsf{Enc}(L_i)$
Worker w_j	$\mathbf{s}_j$, M_j	$\mathsf{Enc}(\mathbf{s}_j)$, $\mathsf{Enc}(M_j)$
Decryptor (=platform)	sk	–
Operator	–	–
Mixer	–	–

3.5.1 Initialization

The first part of PTA assigns three cryptographic roles, a *decryptor*, an *operator*, and a *mixer*, and sets up the Paillier cryptosystem.

Cryptographic Role Assignment

We first present our strategy to recruit the three cryptographic roles. The decryptor is responsible for setting up the Paillier cryptosystem and decryption. We assign the decryptor to the platform, because the decryptor, who holds the secret key, should be assigned to the most reliable entity. The operator is responsible for most of the computation in PTA except decryption, and the mixer executes the conditional test protocol with the operator. We assign these two roles to workers via crowdsourcing. Because workers are assumed not to be able to communicate with each other, crowdsourcing these two roles does not violate the non-collusion assumption. In addition, since the computation is performed automatically by a computer program, it is quite unlikely that these entities would act in an adversarial way.

Protocol

The private inputs and outputs of the parties are summarized in Table 3.5. Given the cryptographic role assignment, the decryptor generates the keys $(\mathsf{pk}, \mathsf{sk})$ and broadcasts the public key pk to all the entities while keeping the secret key sk. Each entity encrypts his/her parameters to generate $(\mathsf{Enc}(\mathbf{s}_j), \mathsf{Enc}(M_j))$ for worker w_j and $(\mathsf{Enc}(\mathbf{r}_i), \mathsf{Enc}(L_i))$ for requester t_i.

3.5.2 Private Network Construction

The second part of PTA constructs an encrypted assignment network.

Table 3.6: Private inputs and outputs of the private network construction.

Party	Private input	Private output
Requester t_i	$\mathbf{r}_i$, $\mathsf{Enc}(\mathbf{r}_i)$, $\mathsf{Enc}(L_i)$	–
Worker w_j	$\mathsf{Enc}(\mathbf{s}_j)$, $\mathsf{Enc}(M_j)$	–
Decryptor	sk	–
Operator	–	$(\mathsf{Enc}(\mathbf{C}), \mathsf{Enc}(\mathbf{F}))$, $(\mathsf{Enc}(\mathbf{h}), \mathsf{Enc}(\mathbf{e}))$
Mixer	–	–

Protocol

The private inputs and outputs are summarized in Table 3.6. Steps 1, 2, and 3 construct the left, right, and middle parts, respectively, of the assignment network in Figure 3.2.

Step 0: Initialization

Let $V = \{s, t\} \cup \mathcal{W} \cup \mathcal{T}$. The operator initializes the encrypted assignment network and the internal variables of the push-relabel algorithm as $\mathsf{Enc}(\mathbf{C}) \leftarrow [\mathsf{Enc}(0)]_{v,w \in V}$, $\mathsf{Enc}(\mathbf{F}) \leftarrow [\mathsf{Enc}(0)]_{v,w \in V}$, $\mathsf{Enc}(\mathbf{h}) \leftarrow [\mathsf{Enc}(0)]_{v \in V}$, and $\mathsf{Enc}(\mathbf{e}) \leftarrow [\mathsf{Enc}(0)]_{v \in V}$.

Step 1: Edges from source s to workers $\mathcal{W}$

The first step is to set edges from source s to each worker w_j. The encrypted edges from source s to workers $\mathcal{W}$ can be computed by each worker w_j independently. Each worker w_j sends $\mathsf{Enc}(M_j)$ to the operator, and the operator updates the encrypted capacity of edge (s, w_j) as

$$\mathsf{Enc}(c_{s,w_j}) \leftarrow \mathsf{Enc}(M_j) \ (\forall w_j \in \mathcal{W}).$$

Step 2: Edges from tasks $\mathcal{T}$ to sink t

The second step is to set edges from each task t_i to sink t. The encrypted edges from tasks $\mathcal{T}$ to sink t can be computed by each requester t_i independently. Each requester t_i sends $\mathsf{Enc}(L_i)$ to the operator, and the operator updates the encrypted capacity of edge (t_i, t) as

$$\mathsf{Enc}(c_{t_i,t}) \leftarrow \mathsf{Enc}(L_i) \ (\forall t_i \in \mathcal{T}).$$

Step 3: Edges from workers $\mathcal{W}$ to tasks $\mathcal{T}$

The third step is to set edges from each worker w_j to each task t_i. This step requires the cooperation of the workers and the requesters.

We employ the following lemma to compute the capacity in a privacy-preserving way. The proof is provided in Section 3.5.2.

Lemma 3.1. *Given a skill vector* $\mathbf{s}_j \in \{0,1\}^D$ *of worker* w_j *and a requirement vector* $\mathbf{r}_i \in \{0,1\}^D$ *of task* t_i*, the value* $\|\mathbf{r}_i\|_2^2 - \mathbf{s}_j \cdot \mathbf{r}_i$ *equals the number of features that worker* w_j *does not have, but are required by task* t_i*.*

Then, the capacity of (w_j, t_i) can be computed as follows. First, the parties compute $\mathsf{Enc}(M_\top)$ using MAX. Second, each worker w_j sends $\mathsf{Enc}(\mathbf{s}_j)$ to all the requesters.[1] Third, each requester t_i, upon receiving $\{\mathsf{Enc}(\mathbf{s}_j)\}_{j\in[J]}$, computes

$$\mathsf{Enc}(\|\mathbf{r}_i\|_2^2 - \mathbf{s}_j \cdot \mathbf{r}_i) = \prod_{d=1}^{D} \mathsf{Enc}(r_{i,d}^2) \cdot \prod_{d=1}^{D} \mathsf{Enc}(s_{j,d})^{-r_{i,d}} \tag{3.1}$$

for each $j \in [J]$, and sends them to the operator.[2] Finally, for each $w_j \in \mathcal{W}$ and $t_i \in \mathcal{T}$, the operator, decryptor, and mixer update capacity c_{w_j,t_i} as

$$\mathsf{Enc}(c_{w_j,t_i}) \leftarrow \mathsf{COND}(\mathsf{Enc}(0), \mathsf{Enc}(M_\top), \mathsf{Enc}(\|\mathbf{r}_i\|_2^2 - \mathbf{s}_j \cdot \mathbf{r}_i)). \tag{3.2}$$

After these three steps, the operator initializes the encrypted internal variables of the push-relabel algorithm, $\mathsf{Enc}(\mathbf{F})$, $\mathsf{Enc}(\mathbf{e})$, and $\mathsf{Enc}(\mathbf{h})$ as

$$\begin{aligned}
&\mathsf{Enc}(f_{s,v}) \leftarrow \mathsf{Enc}(c_{s,v}),\ \mathsf{Enc}(e_v) \leftarrow \mathsf{Enc}(c_{s,v})\ (\forall v \in V \backslash \{s\}),\\
&\mathsf{Enc}(h_s) \leftarrow \mathsf{Enc}(|V|),\ \mathsf{Enc}(h_t) \leftarrow \mathsf{Enc}(0),\\
&\mathsf{Enc}(h_{w_j}) \leftarrow \mathsf{Enc}(2)\ (\forall w_j \in \mathcal{W}),\ \mathsf{Enc}(h_{t_i}) \leftarrow \mathsf{Enc}(1)\ (\forall t_i \in \mathcal{T}).
\end{aligned}$$

Correctness

We prove the correctness of the private network construction. Because Steps 1 and 2 set edges E_1 and E_3, respectively, of the encrypted assignment network in an obvious way, we only validate Step 3, which sets edges E_2 of the encrypted assignment network. First, we prove Lemma 3.1 in the following.

[1] Subsequent to this procedure, the workers are permitted to go offline.
[2] Subsequent to this procedure, the requesters may also go offline.

Table 3.7: Truth table of $(r_{i,d} - s_{j,d}) \cdot r_{i,d}$.

	$s_{j,d} = 0$	$s_{j,d} = 1$
$r_{i,d} = 0$	0	0
$r_{i,d} = 1$	1	0

Proof of Lemma 3.1. Given that

$$\|\mathbf{r}_i\|_2^2 - \mathbf{s}_j \cdot \mathbf{r}_i = (\mathbf{r}_i - \mathbf{s}_j) \cdot \mathbf{r}_i = \sum_{d=1}^{D} (r_{i,d} - s_{j,d}) \cdot r_{i,d}, \tag{3.3}$$

we only have to evaluate the summand of Equation (3.3). The truth table of the summand (Table 3.7) shows that the summand equals 1 if and only if worker w_j does not have the d-th feature whereas task t_i requires it. Therefore, $\|\mathbf{r}\|_2^2 - \mathbf{s} \cdot \mathbf{r}$ equals the number of skills that the worker does not have, but the task requires. □

The validation of Step 3 finishes by showing that a requester can compute Equation (3.1) and that Equation (3.2) correctly sets capacity c_{w_j,t_i}. Requester t_i can compute Equation (3.1) because s/he has plaintext $\mathbf{r}_i$ and ciphertexts $\{\mathsf{Enc}(\mathbf{s}_j)\}_{j\in[J]}$. In Equation (3.2), c_{w_j,t_i} is set to 0 if $\|\mathbf{r}_i\|_2^2 - \mathbf{s}_j \cdot \mathbf{r}_i > 0$, *i.e.*, worker w_j cannot complete task t_i, and c_{w_j,t_I} is set to $M_\top$ if worker w_j can complete task t_i. Therefore, the private network construction is proven to be correct.

3.5.3 Private Push-Relabel Protocol

Our private push-relabel protocol is executed by the decryptor, operator, and mixer to compute an encrypted maximum flow based on the push-relabel algorithm. We first introduce two techniques to convert the push-relabel algorithm to preserve privacy, and then present building blocks of the private push-relabel protocol. Finally, we present the main protocol as well as its correctness.

Two Techniques to Convert Algorithms to Preserve Privacy

We present two techniques to compile push and relabel operations to be private.

The first technique is to introduce null operations. In the original operations, the operation halts if any of the applicability conditions (*e.g.*, v is active) does not hold; otherwise, the operation proceeds. However, this

branch leaks substantial information. For example, if the push operation from vertex v to vertex w does not halt, it indicates that there exists edge (v, w), vertex v is active, and so on. We avoid such information leakage by performing a null operation if applicability conditions do not hold. In specific, if the applicability conditions do not hold, our private push protocol sends $\mathsf{Enc}(0)$ flow to a neighboring vertex, and our private relabel protocol increases the height by $\mathsf{Enc}(0)$.

The second technique is to replace the if-else structure by COND. In detail, the following if-else statement combined with the null operation technique,

1: **if** $a > 0$ **then**
2: $\quad b \leftarrow c$
3: **else**
4: $\quad b \leftarrow b$
5: **end if**

can be made private using COND as

1: $\mathsf{Enc}(b) \leftarrow \mathsf{COND}(\mathsf{Enc}(c), \mathsf{Enc}(b), \mathsf{Enc}(a))$.

Building Blocks

Here, we present the building blocks of the private push-relabel protocol, referred to as the *private push protocol* and *private relabel protocol*.

The private push protocol $\mathsf{Push}_p(v, w)$ (Protocol 3.4) executes the push operation $\mathsf{Push}(v, w)$ (Operation 3.2) in a privacy-preserving way. Line 2 computes the temporary flow value from vertex v to w, $\mathsf{Enc}(f')$, assuming that the applicability conditions hold. Then, following the techniques introduced above, line 3 updates $\mathsf{Enc}(f_{v,w})$ to $\mathsf{Enc}(f')$ or keeps $\mathsf{Enc}(f_{v,w})$ depending on the height condition $h_v = h_w + 1$. Note that other applicability conditions, $e_v > 0$ and $(v, w) \in E_F$, can be disregarded; if either one of them or neither of them hold, $f' = f_{v,w}$ also holds, which coincides with the null push operation. Finally, line 4 updates other variables $\mathsf{Enc}(f_{w,v})$, $\mathsf{Enc}(e_v)$, and $\mathsf{Enc}(e_w)$.

The private relabel protocol $\mathsf{Relabel}_p(v)$ (Protocol 3.5) executes the relabel operation $\mathsf{Relabel}(v)$ (Operation 3.3) in a privacy-preserving way. Lines 1–6 compute the temporary height of vertex v, $\mathsf{Enc}(h'_v)$, assuming the applicability conditions hold. We compute $\mathsf{Enc}(h'_v)$ by creating a set of encrypted heights, S_v, such that $\mathsf{MIN}(S_v) = \mathsf{Enc}(h'_v)$ holds. We make use of the fact that $0 \leq h_v \leq |V| + 2$ always holds for all $v \in V$ in case of assignment networks (see Lemma A.1 in Appendix A.3). For each vertex $w \in \mathcal{W} \cup \{t\}$, the protocol adds $\mathsf{Enc}(h_w + 1)$ to S_v if $(v, w) \in E_F$, and adds $\mathsf{Enc}(|V| + 3)$

Protocol 3.4 $\mathsf{Push}_p(v, w)$.

Parties: operator, decryptor, and mixer.

1: Operator sets $\mathsf{Enc}(f_{\text{old}}) \leftarrow \mathsf{Enc}(f_{v,w})$.

2: Parties compute $\mathsf{Enc}(f')$ as

$$\mathsf{Enc}(f') \leftarrow \mathsf{MIN}(\mathsf{Enc}(c_{v,w}), \mathsf{Enc}(f_{v,w} + e_v)).$$

3: Parties compute

$$\begin{aligned}\mathsf{Enc}(f'') &\leftarrow \mathsf{COND}(\mathsf{Enc}(f_{v,w}), \mathsf{Enc}(f'), \mathsf{Enc}(h_v - h_w - 1)),\\ \mathsf{Enc}(f_{v,w}) &\leftarrow \mathsf{COND}(\mathsf{Enc}(f_{v,w}), \mathsf{Enc}(f''), \mathsf{Enc}(h_w - h_v + 1)).\end{aligned}$$

4: Operator updates

$$\begin{aligned}\mathsf{Enc}(e_v) &\leftarrow \mathsf{Enc}(e_v - f_{v,w} + f_{\text{old}}),\\ \mathsf{Enc}(e_w) &\leftarrow \mathsf{Enc}(e_w + f_{v,w} - f_{\text{old}}),\\ \mathsf{Enc}(f_{w,v}) &\leftarrow \mathsf{Enc}(-f_{v,w}).\end{aligned}$$

Table 3.8: Private inputs and outputs of the private push-relabel protocol.

Party	Private input	Private output
Decryptor	sk	$\mathcal{A}^\star$
Operator	$(\mathsf{Enc}(\mathbf{C}), \mathsf{Enc}(\mathbf{F}))$, $(\mathsf{Enc}(\mathbf{h}), \mathsf{Enc}(\mathbf{e}))$	–
Mixer	–	–

otherwise; adding $\mathsf{Enc}(|V| + 3)$ to S_v does not have any influence on the output of $\mathsf{MIN}(S_v)$. After computing the temporal height $\mathsf{Enc}(h'_v)$, line 7 checks the applicability conditions to update $\mathsf{Enc}(h_v)$. The first operation rejects the update unless $e_v > 0$, and the second operation rejects the update unless $h_v + 1 \leq h'_v$, *i.e.*, unless $h_v \leq h_w$ for all $w \in V$ such that $(v, w) \in E_F$.

Main Protocol

Protocol 3.6 describes the main protocol of our private push-relabel protocol. Table 3.8 summarizes the private inputs and outputs.

First, the operator shuffles the order of $\mathcal{W}$ and $\mathcal{T}$ arbitrarily. Then, the parties sequentially choose worker $w_j \in \mathcal{W}$ according to the order of $\mathcal{W}$, apply the private relabel protocol to w_j, apply the private push protocol from worker w_j to source s, and apply the private push protocol from worker w_j to each task $t_i \in \mathcal{T}$. Then, the parties sequentially choose task $t_i \in \mathcal{T}$ according

Protocol 3.5 $\mathsf{Relabel}_p(v)$.

Parties: operator, decryptor, and mixer.

1: Operator initializes set $S_v \leftarrow \emptyset$.
2: **for** each $w \in \mathcal{W} \cup \{t\}$ **do**
3: Parties compute $\mathsf{Enc}(h')$ as

$$\mathsf{COND}(\mathsf{Enc}(h_w + 1), \mathsf{Enc}(|V| + 3), \mathsf{Enc}(c_{v,w} - f_{v,w})).$$

4: Operator updates $S_v \leftarrow S_v \cup \{\mathsf{Enc}(h')\}$.
5: **end for**
6: Parties compute $\mathsf{Enc}(h'_v) \leftarrow \mathsf{MIN}(S_v)$.
7: Parties compute $\mathsf{Enc}(h_v)$ as

$$\mathsf{Enc}(h''_v) \leftarrow \mathsf{COND}(\mathsf{Enc}(h'_v), \mathsf{Enc}(h_v), \mathsf{Enc}(e_v)),$$
$$\mathsf{Enc}(h_v) \leftarrow \mathsf{COND}(\mathsf{Enc}(h_v), \mathsf{Enc}(h''_v), \mathsf{Enc}(h_v - h''_v + 1)).$$

to the order of $\mathcal{T}$, apply the private relabel protocol to t_i, apply the private push protocol from task t_i to sink t, and apply the private push protocol from task t_i to each worker $w_j \in \mathcal{W}$. The protocol stops after repeating the above procedures

$$K = |\mathcal{W}||\mathcal{T}| \left(\left\lceil \frac{|V| - 1}{2} \right\rceil + 1 \right) (|V| + 2) + |\mathcal{W}|(2|V| + 1) + |\mathcal{T}|(2|V| + 3)$$

times regardless of the input network. Finally, the operator sends $\mathsf{Enc}(\mathbf{F})$ to the platform, and the platform extracts an assignment from it.

Correctness

We prove the correctness of our private push-relabel protocol. As the private push and relabel protocols are essentially the same as the original operations, we only have to prove that the number of iterations K in Protocol 3.6 is sufficient for convergence.

We derive the sufficient number of iterations by first evaluating the upper bounds on the numbers of push and relabel operations on assignment networks. Taking advantage of the structure of assignment networks, we derive new upper bounds in Corollaries A.1 and A.2 that are tighter than those proved by Goldberg and Tarjan (Goldberg and Tarjan, 1988) for general networks. Corollary A.1 states that the upper bound on the number of relabel

operations is

$$|\mathcal{W}|(|V|-1)+|\mathcal{T}|(|V|+1).$$

Corollary A.2 states that the upper bound on the number of push operations is

$$(|V|+2)\left[|\mathcal{W}||\mathcal{T}|\left(\left\lceil\frac{|V|-1}{2}\right\rceil+1\right)+|\mathcal{W}|+|\mathcal{T}|\right].$$

Given these corollaries, K iterations are sufficient to obtain a maximum flow, because at each iteration, if maximum flow is not achieved, there exists at least one vertex to which either the non-null push or relabel operation is applicable (see Lemma 2.1 (Goldberg and Tarjan, 1988)).

The number of iterations required by PTA is much smaller than that of the existing protocol (Aly et al., 2013), which requires

$$(2|V|-1)(|V|-2)+2|V||E|+4|V|^2|E|$$

iterations in the main loop of Protocol 3.6. Comparing the coefficient of the leading terms, $|V|^2|E|$, the existing one has 4, whereas our protocol has 1/2, indicating that our protocol is eight times faster.

3.5.4 Security

This section discusses the security of PTA based on the cryptographic definition (Goldreich, 2004). Given the functionality of PTA described in Table 3.4, the security is stated as Theorem 3.1.

Theorem 3.1 (Security of PTA)**.** *Let f_{PTA} be a functionality whose inputs and outputs are defined as Table 3.4. PTA privately computes f_{PTA}.*

PTA consists of three parts; thus, we employ the following lemmata, each of which states the security of each individual part.

Lemma 3.2 (Security of the initialization)**.** *Let f_{init} be a functionality whose inputs and outputs are defined as Table 3.5. The initialization of PTA privately computes f_{init}.*

Lemma 3.3 (Security of private network construction (PNC))**.** *Let f_{PNC} be a functionality whose inputs and outputs are defined as Table 3.6. The private network construction privately computes f_{PNC}.*

Lemma 3.4 (Security of private push-relabel (PPR) protocol)**.** *Let f_{PPR} be a functionality whose inputs and outputs are defined as Table 3.8. The private push-relabel protocol privately computes f_{PPR}.*

Given these lemmata, it is straightforward to prove Theorem 3.1 because the private outputs of the initialization and PNC are ciphertexts that do not leak any information without the secret key. The proofs of the lemmata are provided in Appendix A.2.2.

3.6 Computation Time of PTA and Acceleration Methods

Finally, we conclude this section by making an estimate of the computation time of PTA and discuss several acceleration methods.

3.6.1 Computation Time of PTA

We evaluate the cryptographic computational overhead of PTA, which dominates the computation time of the whole procedure. We utilize the estimates shown in Tables 2.1 and 3.3. Our evaluation of the computation time depends on the range of the plaintext space L, the number of features D, and the number of workers and tasks, $|\mathcal{W}|$ and $|\mathcal{T}|$. In the following, we substitute all of these parameters by 100 for clarification purposes.

Initialization

The initialization step involves one key generation and $(D+1)(|\mathcal{W}|+|\mathcal{T}|)$ encryptions. Therefore, the computation time required by the initialization step is

$$\begin{aligned} T_{\text{init}} &= T_{\text{keygen}} + (D+1)(|\mathcal{W}|+|\mathcal{T}|)T_{\text{enc}} \\ &= 207 \text{ seconds} \approx 3.5 \text{ minutes}. \end{aligned}$$

Private Network Construction

The private network construction involves $|\mathcal{W}||\mathcal{T}|$ operations of COND and $2|V|^2 + 2|V|$ encryptions. Therefore, the computation time required by the private network construction is

$$\begin{aligned} T_{\text{PNC}} &= |\mathcal{W}||\mathcal{T}|T_{\mathsf{COND}} + (2|V|^2 + 2|V|)T_{\text{enc}} \\ &= 2.10 \times 10^4 \text{ seconds} \approx 5.83 \text{ hours}. \end{aligned}$$

Private Push-Relabel Protocol

The private push-relabel protocol consists of the private push and private relabel protocols. We first estimate the computation time of these components individually.

- Private Push Protocol ($\mathsf{Push}_p(v, w)$)
  The private push protocol consists of one MIN, five subtraction operations, and two COND operations. Therefore, the computation time is

$$\begin{aligned} T_{\mathsf{Push}} &= T_{\mathsf{MIN}} + 2T_{\mathsf{COND}} + 5T_{\mathrm{sub}} \\ &= 6.06 \text{ seconds.} \end{aligned}$$

- Private Relabel Protocol ($\mathsf{Relabel}_p(v)$)
  The private relabel protocol consists of $|\mathcal{W}| + 3$ COND, $|\mathcal{W}|$ MIN, and $|\mathcal{W}| + 2$ subtract operations. Therefore, the computation time is

$$\begin{aligned} T_{\mathsf{Relabel}} &= (|\mathcal{W}| + 3)T_{\mathsf{COND}} + |\mathcal{W}|T_{\mathsf{MIN}} + (|\mathcal{W}| + 2)T_{\mathrm{sub}} \\ &= 4.10 \times 10^2 \text{ seconds} \approx 7 \text{ minutes.} \end{aligned}$$

Then, we estimate the computation time of the private push-relabel protocol in total. One iteration of the private push-relabel protocol consists of $2|\mathcal{W}||\mathcal{T}|+|\mathcal{W}|+|\mathcal{T}|$ $(= 20200)$ push operations and $|\mathcal{W}|+|\mathcal{T}|$ $(= 200)$ relabel operations. Therefore, one iteration of PTA requires 2.04×10^5 seconds, which approximately equals 57 hours. Since the number of iterations required by the private push-relabel protocol is $K = 2.08 \times 10^8$ times, the private push-relabel protocol requires 4.24×10^{13} seconds $\approx 1.34 \times 10^4$ centuries.

3.6.2 Acceleration Techniques

As we have seen in the previous section, the full computation of the private push-relabel protocol is infeasible. This section discusses two approaches to accelerating the protocol: (i) to reduce the computation time of one iteration and (ii) to reduce the number of iterations.

Reduction of the Computation Time of One Iteration

The computation time of one iteration of the private push-relabel protocol can be reduced by introducing parallel computing. Because we recruit cryptographic roles using crowdsourcing, the number of parties in the protocol is highly-scalable. Considering that the decryptor (*i.e.*, the platform) is likely

to have much computational power, it is possible to parallelize the conditional test protocol itself into multiple parties and execute it multiple times at the same time. In the following, we discuss these two ideas of parallelization.

The first idea is to parallelize the inequality test in the conditional test protocol. Assume that the decryptor has multiple CPUs and we have the same number of operators. The parallel conditional test protocol is defined as follows. Each of the operators independently creates each part of the ordered set S and sends it to the representative operator. The representative operator constructs S and sends it to the decryptor. Finally, the decryptor decrypts each encryption in S using the multiple CPUs in parallel and returns bit a to the operator. For example, if we have ten CPUs and ten operators, the computation time of the parallel conditional test protocol will be

$$\tilde{T}_{\mathsf{COND}} = 1.98 \times 10^2 \text{ ms}.$$

The second idea is to execute the conditional test protocol multiple times at the same time. Assume that we have multiple sets of parties. Then, lines 2–5 of Protocol 3.5 can be independently computed by multiple sets of parties. In addition, the min protocol (line 6) can be computed in parallel using the divide-and-conquer approach. For example, if we have ten sets of parties, the computation time of the private relabel protocol will be

$$\tilde{T}_{\mathsf{Relabel}} = 11T_{\mathsf{COND}} + 15T_{\mathsf{MIN}} + 12T_{\mathrm{sub}}.$$

By combining these two ideas, the computation time of the private push and private relabel protocols with 100 CPUs, 100 operators, and 10 mixers will be

$$\tilde{T}_{\mathsf{Push}} = \tilde{T}_{\mathsf{MIN}} + 2\tilde{T}_{\mathsf{COND}} + 5T_{\mathrm{sub}} = 5.97 \times 10^2 \text{ ms},$$
$$\tilde{T}_{\mathsf{Relabel}} = 11\tilde{T}_{\mathsf{COND}} + 15\tilde{T}_{\mathsf{MIN}} + 2T_{\mathrm{sub}} = 5.16 \text{ seconds},$$

and the computation time of one iteration of the private push-relabel protocol will be 218 minutes.

Reduction of the Number of Iterations

The number of iterations can be reduced by means of the following two ideas. The first idea is to examine the convergence of the push-relabel protocol at the end of every iteration (between lines 16 and 17 in Protocol 3.6), allowing a little breach of privacy. The convergence is examined by executing the following inequality test:

$$\mathsf{INEQ}\left(\mathsf{Enc}\left(\sum_{w_j \in \mathcal{W}} e_{w_j} + \sum_{t_i \in \mathcal{T}} e_{t_i}\right), \mathsf{Enc}(0)\right).$$

The push-relabel protocol converges if and only if the output is 0. Note that the encryption of the sum of excesses can be easily computed by the operator. The second idea is to halt the protocol earlier than K iterations. After halting, the platform can obtain a feasible assignment by executing Protocol 3.7.

Both ideas are rooted in the expectation that the number of iterations required for convergence is actually much smaller than the upper bound on it, K. We experimentally validate this expectation by examining the approximation ratio[3] at each iteration on randomly generated assignment networks; our ideas are validated if the approximation ratio approaches 1.0 much earlier than K.

The experimental procedure is as follows. We first set the size of an assignment network as $(|\mathcal{W}|, |\mathcal{T}|) = (100, 100)$ or $(|\mathcal{W}|, |\mathcal{T}|) = (500, 500)$ and generate a random assignment network as follows:

1. For each worker w_j, we draw M_j from Uniform($[100]$).

2. For each task t_i, we draw L_i from Uniform($[100]$).

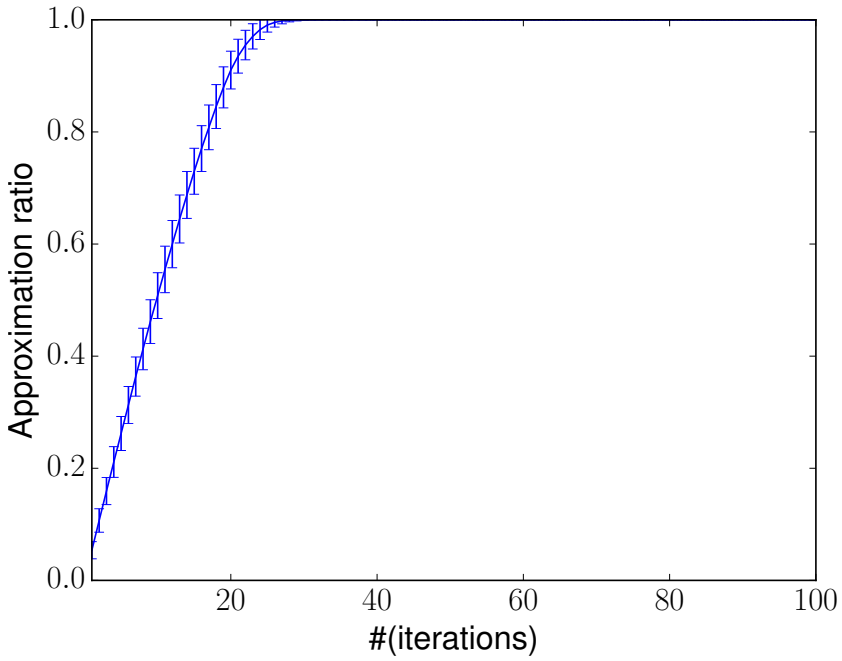
3. For each worker w_j and task t_i, we set edge (w_j, t_i) with probability 0.6.

Then, we execute the push-relabel protocol and examine the size of the feasible assignment at each iteration. We repeat this procedure $1,000$ times to compute the mean and standard deviation of the approximation ratio at each iteration.

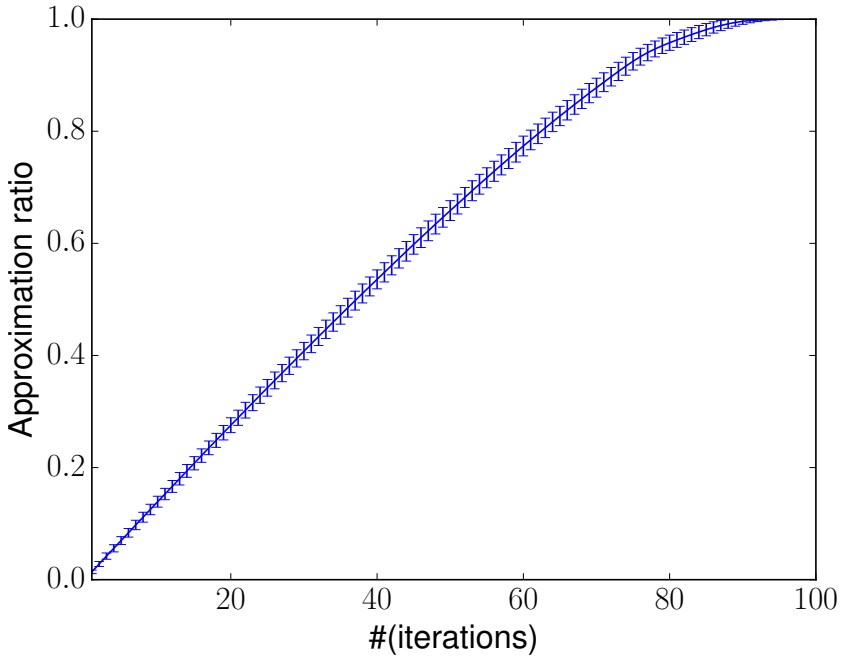
Figure 3.3 shows the experimental results. It suggests that 30 iterations are sufficient to obtain the maximum flow if $|\mathcal{W}| = |\mathcal{T}| = 100$ and 100 iterations if $|\mathcal{W}| = |\mathcal{T}| = 500$, both of which are significantly smaller than K. In addition, the small standard deviations imply that this result does not much depend on the realization of the random assignment network, which suggests that preliminary experiments with synthetic data enable us to guess the number of iterations necessary for convergence. Based on the experimental results, we conclude that both of our ideas are effective solutions to make PTA feasible.

By combining the parallization and early-stopping techniques, the 4.5-day computation of PTA will be enough to obtain an optimal task assignment for the network with $|\mathcal{W}| = |\mathcal{T}| = 100$.

[3]Letting O be the size of the task assignment given by an approximation algorithm and letting $O^\star$ be that given by the exact algorithm, the approximation ratio is defined as $O/O^\star$.



(a) $|\mathcal{W}| = |\mathcal{T}| = 100$.



(b) $|\mathcal{W}| = |\mathcal{T}| = 500$.

Figure 3.3: Approximation ratio at each iteration. The x- and y-axes correspond to the number of iterations and the approximation ratio, respectively. The error bars represent the standard deviations.

3.7 Summary and Future Work

We point out the privacy issues affecting both workers and requesters in crowdsourcing task assignment. Assigning tasks to appropriate workers necessitates collecting the features of the tasks as well as those of the workers, whose privacy may be invaded. We present a private task assignment protocol that outputs an optimal assignment while preserving their privacy. Noting that the task assignment problem can be reduced to the maximum flow problem, our protocol first constructs a network whose maximum flow coincides with an optimal assignment, and then computes a maximum flow on the network, both of which are executed in a privacy-preserving way. After the execution of our protocol, the crowdsourcing platform only learns the optimal assignment, whereas none of the other parties learn anything.

An interesting direction for future work would be to incorporate a more sophisticated task assignment algorithm. For example, Bragg et al. 2014 have introduced a probabilistic crowdsourcing model in which workers return correct answers with a probability proportional to their skills. In this model, the objective function differs from ours, in that it requires another algorithm to be made private to compute an optimal assignment.

Another promising research direction would be to integrate our protocol with other privacy-preserving crowdsourcing protocols. For example, the research in Chapter 5 provides a privacy-preserving method to estimate skills of workers in crowdsourcing, which can be used as feature vectors of workers in our setting.

Protocol 3.6 $\mathsf{Push\text{-}Relabel}_p(\mathsf{Enc}(N))$.

Parties: operator, decryptor, and mixer.

Private output of platform: maximum task assignment $\mathcal{A}^\star$.

Operator shuffles the order of workers and tasks $\mathcal{W}, \mathcal{T}$ and sets

$$K \leftarrow |\mathcal{W}||\mathcal{T}| \left(\left\lceil \frac{|V|-1}{2} \right\rceil + 1 \right) (|V|+2) + |\mathcal{W}|(2|V|+1) + |\mathcal{T}|(2|V|+3).$$

for $k = 1, \ldots, K$ **do** ▷ Main loop of our private push-relabel protocol.
  for each $w_j \in \mathcal{W}$ **do**
    Parties execute $\mathsf{Relabel}_p(w_j)$.
    Parties execute $\mathsf{Push}_p(w_j, s)$.
    for each $t_i \in \mathcal{T}$ **do**
      Parties execute $\mathsf{Push}_p(w_j, t_i)$.
    end for
  end for
  for each $t_i \in \mathcal{T}$ **do**
    Parties execute $\mathsf{Relabel}_p(t_i)$.
    Parties execute $\mathsf{Push}_p(t_i, t)$.
    for each $w_j \in \mathcal{W}$ **do**
      Parties execute $\mathsf{Push}_p(t_i, w_j)$.
    end for
  end for
end for
Operator sends $\mathsf{Enc}(\mathbf{F})$ to platform.
Platform initializes $\mathcal{A}^\star \leftarrow \emptyset$. ▷ Extract an assignment from the flow.
for each edge $(w_j, t_i) \in \mathcal{W} \times \mathcal{T}$ **do**
  Platform decrypts $\mathsf{Enc}(f_{w_j,t_i})$.
  Platform updates $\mathcal{A}^\star \leftarrow \mathcal{A}^\star \cup \{(w_j, t_i)\}^{f_{w_j,t_i}}$.
end for

Protocol 3.7 Post-processing for the early-stopping approach

Parties: operator, decryptor, and mixer.

Private output of platform: feasible task assignment $\mathcal{A}^\star$.

for each $t_i \in \mathcal{T}$ **do** ▷ Convert an assignment to be feasible.
  for each $w_j \in \mathcal{W}$ **do**
    Operator sets $\mathsf{Enc}(f_{\text{old}}) \leftarrow \mathsf{Enc}(f_{t_i,w_j})$.
    Parties compute $\mathsf{Enc}(f')$ as

$$\mathsf{Enc}(f') \leftarrow \mathsf{MIN}(\mathsf{Enc}(c_{t_i,w_j}), \mathsf{Enc}(f_{t_i,w_j} + e_{t_i})).$$

    Operator updates

$$\begin{aligned} \mathsf{Enc}(e_{t_i}) &\leftarrow \mathsf{Enc}(e_{t_i} - f_{t_i,w_j} + f_{\text{old}}), \\ \mathsf{Enc}(e_{w_j}) &\leftarrow \mathsf{Enc}(e_{w_j} + f_{t_i,w_j} - f_{\text{old}}), \\ \mathsf{Enc}(f_{w_j,t_i}) &\leftarrow \mathsf{Enc}(-f_{t_i,w_j}). \end{aligned}$$

  end for
end for
Operator sends $\mathsf{Enc}(\mathbf{F})$ to platform.
Platform initializes $\mathcal{A}^\star \leftarrow \emptyset$. ▷ Extract an assignment.
for each edge $(w_j, t_i) \in \mathcal{W} \times \mathcal{T}$ **do**
  Platform decrypts $\mathsf{Enc}(f_{w_j,t_i})$.
  Platform updates $\mathcal{A}^\star \leftarrow \mathcal{A}^\star \cup \{(w_j, t_i)\}^{f_{w_j,t_i}}$.
end for

Chapter 4

Instance-Privacy Preservation

4.1 Introduction

Crowdsourcing entails the invasion of instance privacy[1] as pointed out in Section 1.5.3. A worker must access task instances in order to complete a task, which enables the worker to extract sensitive information from them. For example, consider a task to transcribe audio recordings of business meetings, where an audio recording corresponds to an instance. The content of such a recording includes confidential information of companies. Besides this example, there exist many other tasks whose instances contain sensitive information, such as a task to digitize hand-written texts and a task to detect objects in images. Therefore, there is a strong demand on developing a method to submit tasks with instance-privacy preserved.

There are two research lines to cope with instance privacy: theoretical and practical studies. Varshney (2012) studied a theoretical aspect of instance-privacy, aiming to establish a mathematical model of the random perturbation approach. Little and Sun (2011) proposed a practical protocol tailored for a human OCR task, whose objective is to digitize handwritten medical forms. They leverage a template of a medical form to decompose each form into items such as a name, an address, and a medical history. This decomposition prevents a worker from linking a personal identifier (*e.g.*, name) and its property (*e.g.*, medical history).

To summarize the existing work, a general and practical instance-privacy-preserving (IPP) protocol[2] is still underexplored, and a lack of performance measures for IPP protocols hinders us from investigating generally-applicable

[1] Instance privacy is defined as a state that sensitive information contained in an instance is not revealed.

[2] An IPP protocol poses preprocessed instances to workers so as to preserve instance privacy whilst enabling a worker to perform a task on instances.

IPP protocols. Since privacy preservation usually comes at a price of utility, the quantification of the trade-off between them is essential to examine the applicability of an IPP protocol to a specific pair of a task and a privacy definition, to tune parameters of it, and to compare the performance of multiple IPP protocols. However, there exists no performance evaluation scheme for IPP protocols in a crowdsourcing setting, as far as we know. To this end, we set our research goal to develop a practical performance measure for an IPP protocol.

The main difficulty in quantifying utility and privacy of IPP protocols is that any algorithm evaluating them must incorporate humans in it. In a crowdsourcing setting, instances are used to generate results as specified by a task instruction, and only a human can execute the task and invade instance privacy. Since the output is specified, the utility should be measured by the quality of results, which requires responses by humans. Further, the invasion of instance privacy is also performed only by humans, and therefore, the intervention of humans is inevitable. In this light, the performance evaluation of IPP protocols requires responses from workers, which makes it impossible to employ existing evaluation algorithms.

In this chapter, we present an utility-privacy trade-off analyzer (UPTA) for IPP protocols. Given an IPP protocol, a set of instances, and the definitions of a task and privacy, UPTA measures the utility and the privacy of the IPP protocol. The utility is quantified by the quality degradation of results after applying an IPP protocol, and the privacy is quantified by the amount of the sensitive information contained in instances preprocessed by the IPP protocol. Our main idea is to model the task execution and the privacy invasion as sampling of a result and sensitive information from probability distributions. The models can be empirically estimated by simulating the task execution and the privacy invasion using a real crowdsourcing platform. Given the models, we are able to quantify both utility and privacy using divergence-based measures such as the Kullback-Leibler divergence.

As a case study of UPTA, we investigate the properties of an instance clipping (IC) protocol, employing a task to detect heads in an image and defining the activity of a person in an image as instance privacy. The IC protocol is a generalization of the protocols developed by Little and Sun (2011). It preserves instance privacy by clipping instances with a window of a fixed size. We design two experiments to demonstrate the effectiveness and validity of UTA. The first experiment applies UPTA to investigate the properties of the IC protocol. The experimental result shows that the IC protocol can preserve privacy without much degrading the quality of results. In specific, it reduces the amount of information leakage to 0.6 times of that without privacy preservation while the quality loss of applying the IC

Protocol 4.1 Crowdsourcing with privacy invasion.

Input: instance i.
Output of a requester: result r.
Output of a worker: sensitive information s.

1: The requester submits a task with instance i.
2: A worker samples result r from $p_t(R \mid I = i)$.
3: The worker returns result r to the requester.
4: The worker extracts sensitive information s from $p_p(S \mid I = i)$.

protocol is 1.1 times of that without the IC protocol. We further discuss how to determine the parameter of the IC protocol. In the second experiment, we examine the validity of UPTA by comparing the utility computed by UPTA and standard measures that can be computed in this special case. Since both scores have similar trends including outlier-like behavior, we conclude that the score provided by UPTA is valid.

4.2 Crowdsourcing Model

We specify the crowdsourcing model used in this chapter based on the abstraction of crowdsourcing introduced in Section 1.5.2. At the initial state, a requester has instance $i \in \mathcal{I}$ and is willing to obtain result $r \in \mathcal{R}$ of performing a task on instance i, where $\mathcal{I}$ and $\mathcal{R}$ are sets of possible instances and results, respectively. For example, when a task is to give yes if an image contains a face, and no otherwise, the image corresponds to an instance, and the label {yes/no} corresponds to a result. The task request procedure in crowdsourcing is modeled as Protocol 4.1. The protocol starts when the requester submits a task with instance $i \in \mathcal{I}$ (line 1 in Protocol 4.1). It involves the following two processes: task execution and privacy invasion, each of which is described in the following sections.

4.2.1 Task Execution

The worker, receiving instance i, performs the task on instance i to generate result r and returns the result to the requester (lines 2 and 3 in Protocol 4.1). This process is modeled by the task execution model (Definition 4.1). Let us represent a task by a conditional probability distribution over the set of results given an instance, $p_t(R \mid I)$, which we call a task execution model.[3]

[3]We use capital letters for random variables and the corresponding lowercase letters for realizations of them.

Then, the task execution process given instance i is modeled as sampling from $p_t(R \mid I = i)$. We assume that only a human can sample results from the model.

Definition 4.1 (Task execution model)**.** *Let instance I and result R be random variables whose ranges are $\mathcal{I}$ and $\mathcal{R}$. Let us represent a task by a conditional probability distribution $p_t(R \mid I)$, which we call a task execution model. Then, the execution of the task given instance $i \in \mathcal{I}$ is modeled as sampling from $p_t(R \mid I = i)$.*

4.2.2 Privacy Invasion

Aside from the task execution, the worker extracts sensitive information from the instance (line 4 in Protocol 4.1). Assuming that the definition of sensitive information is given, let us denote a set of possible sensitive values as $\mathcal{S}$ and a sensitive value of instance i as $s \in \mathcal{S}$. For example, if an instance is an image, $\mathcal{S}$ may correspond to a set of possible actions that the person in the image is engaged in. Noticing that privacy invasion can be interpreted as task execution by regarding a sensitive value as a result, this process is modeled as shown in Definition 4.2 in the same way as Definition 4.1.

Definition 4.2 (Privacy invasion model)**.** *Let instance I and sensitive value S be random variables whose ranges are $\mathcal{I}$ and $\mathcal{S}$. Let us represent a privacy invasion by a conditional probability distribution $p_p(S \mid I)$, which we call a privacy invasion model. Then, the privacy invasion given instance $i \in \mathcal{I}$ is modeled as sampling from $p_p(S \mid I = i)$.*

4.2.3 Validity of the Models

Modeling the process as sampling from a probability distribution is justified considering that the quality of a result varies depending on the ability of the worker and the difficulty of the instance, as often stated in the literature of the quality control problem in crowdsourcing (Lease, 2011). We introduce a probability distribution to capture the uncertainty. Furthermore, our model is essential in case of a subjective task such as a questionnaire task, which does not necessarily have a single ground truth.

Our model is more general than standard probabilistic models of task execution, *e.g.*, the model proposed by Dawid and Skene (1979), in that the details of a process such as the ability of a worker are not explicitly modeled. Since the performance measures introduced in the next section are built based on our models, the generality is indispensable so as to keep the applicability of our performance measures.

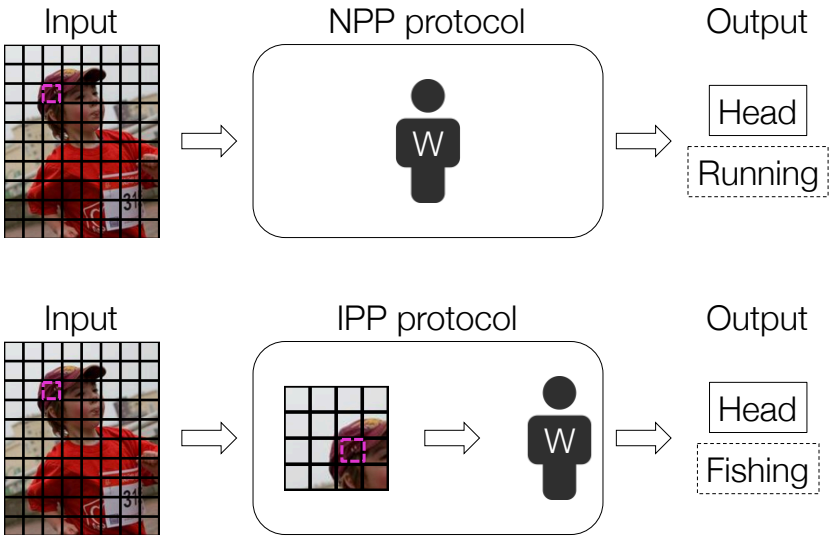


Figure 4.1: Illustration of a non-privacy-preserving (NPP) protocol and an instance-privacy preserving (IPP) protocol. Both protocols share the input and outputs (a label framed by solid lines is a result, and that framed by broken lines is a sensitive value). The IPP protocol involves preprocessing of an instance to preserve instance privacy (clipping, in this case), which alters both task execution and privacy invasion models.

4.3 Utility-Privacy Trade-Off Analyzer (UPTA)

We present our framework called the utility-privacy trade-off analyzer (UPTA), which evaluates the performance on both utility and privacy of an instance-privacy-preserving (IPP) protocol. Given an IPP protocol and the definitions of a task and privacy, UPTA computes the quality degradation of a result (utility) and the amount of information leakage (privacy).

4.3.1 Instance-Privacy-Preserving (IPP) Protocol

The standard crowdsourcing process we have introduced in Protocol 4.1 takes an instance $i \in \mathcal{I}$ as its input and outputs a result $r \in \mathcal{R}$ and a sensitive value $s \in \mathcal{S}$. We call Protocol 4.1 a non-privacy-preserving (NPP) protocol to emphasize that it is ignorant of the instance-privacy problem. An instance-privacy preserving (IPP) protocol has the same input and output with the NPP protocol as shown in Figure 4.1. Due to the instance-privacy-preserving function in the IPP protocol (clipping, in Figure 4.1), it has different task execution and privacy invasion models from those of the NPP protocol. In the following, the privacy invasion model and task execution model of an

NPP protocol are denoted by $p_t(R \mid I)$ and $p_p(S \mid I)$, and those of an IPP protocol are denoted by $p'_t(R \mid I; \theta)$ and $p'_p(S \mid I; \theta)$, where $\theta \in \Theta$ represents the parameter of the IPP protocol.

4.3.2 Task Information Loss

A task information loss (Definition 4.3) quantifies the quality by measuring how much the quality of a result degrades when we apply the IPP protocol instead of the NPP protocol. A small task information loss indicates that the task performance is preserved even after applying a privacy preservation technique.

Definition 4.3 (Task information loss)**.** *Given a task execution model of the NPP protocol $p_t(R \mid I)$ and that of the IPP protocol $p'_t(R \mid I; \theta)$, the task information loss is defined as*

$$L_t(p_t, p'_t) := \mathbb{E}_{p(I)}[\mathrm{KL}(p_t(R \mid I) \parallel p'_t(R \mid I; \theta))],$$

where $\mathrm{KL}(p \parallel q)$ *is the KL divergence of q from p, and $p(I)$ is a probability distribution over $\mathcal{I}$.*

The task information loss has two main advantages over standard measures such as precision, recall, and accuracy scores. First, the task information loss can be applied to various types of task results with little modification. A task result can be either a multi-class label, an integer, or a real number depending on a task definition. The task information loss can be calculated for them simply by changing the probability distributions. On contrary, a standard measure is basically task-specific, and therefore, it is not applicable generally. Second, the task information loss can be applied to even a subjective task, for example, a survey task whose objective is to collect subjective opinions of people. On contrary, standard measures cannot be applied to a subjective task because the ground truths for subjective opinions cannot be defined.

4.3.3 Privacy Information Gain

A privacy information gain (Definition 4.4) quantifies the privacy by measuring how much sensitive information can be extracted from instances when the IPP protocol is used. It captures the uninformativeness of an instance posed to a worker in the IPP protocol. A small privacy information gain indicates that the instance and the sensitive value are almost independent, and therefore, the sensitive information cannot be learned from the instance.

Definition 4.4 (Privacy information gain)**.** *Given a privacy invasion model of the IPP protocol $p'_p(S \mid I; \theta)$, the privacy information gain is defined as*

$$L_p(p'_p) := \mathbb{E}_{p(I)}[\mathrm{KL}(p'_p(S \mid I; \theta) \parallel p'_p(S; \theta))],$$

which is the mutual information of S and I.

The privacy information gain is the first criterion to evaluate the amount of privacy leakage in a crowdsourcing setting, to the best of our knowledge. The standard measures are not suitable for this purpose, because they consider that privacy is preserved even when a worker extracts an incorrect sensitive value from an instance, implying that they do not satisfy the uninformativeness principle (Machanavajjhala et al., 2007), which is used in privacy preserving data publishing (Fung et al., 2010). The uninformative principle roughly states that published data should give little additional information beyond the background knowledge of an attacker. The privacy information gain reflects this principle by employing the mutual information. The privacy information gain can penalize the case when workers extract an incorrect sensitive value from an instance; a malicious worker can harm others using even an incorrect sensitive value.

4.3.4 Empirical Estimation

These performance measures can be estimated empirically. We apply the plug-in estimation using the empirical estimation of probability distributions. For each instance i, we repeatedly execute the protocol M times to obtain M samples $\{r^{(m)}\}_{m \in [M]}$ from $p_t(R \mid I = i)$. Then, for each instance i and result r, we calculate an empirical estimation of $p_t(R \mid I = i)$ using an additive smoothing as

$$\hat{p}_t(R = r \mid I = i) \propto \left|\left\{m \in [M] \mid r = r^{(m)}\right\}\right| + \tau, \tag{4.1}$$

where τ (> 0) is a smoothing parameter. Given a set of instances at hand $\{i_1, \ldots, i_N\}$, the distribution over a set of possible instances $p(I)$ is estimated as

$$\hat{p}(I) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left[I = i_n\right]. \tag{4.2}$$

Other probability distributions can also be empirically calculated in the same way.

4.3.5 Properties

Assume that a requester has an option to use an IPP protocol to have his/her task processed or not to use it with the task remaining unprocessed. If s/he uses the IPP protocol, s/he will acquire a certain amount of utility from task results, and at the same time, will suffer some loss because of privacy invasion by workers. In this section, we estimate both quantities based on the task information loss and privacy information gain, showing that minimizing the task information loss implies maximizing the lower-bound of the utility and minimizing the privacy information gain implies minimizing the upper-bound of the privacy loss. Since these statements hold for any utility and loss functions, which are often unknown to the requester, our performance measures are useful proxies valid for various purposes.

Assume that there exists a utility function $U_t(R, I)$ (> 0), which encodes the utility the requester obtains when the result of processing instance I is R. Then, the expected utility of using the IPP protocol is defined as

$$\mathbb{E}_{p(I)}\mathbb{E}_{p'_t(R|I;\theta)}\left[U_t(R, I)\right]. \tag{4.3}$$

Equation (4.3) is lower-bounded as follows:

$$\begin{aligned}
&\mathbb{E}_{p(I)}\mathbb{E}_{p'_t(R|I;\theta)}\left[U_t(R, I)\right] \\
=&\mathbb{E}_{p(I)}\mathbb{E}_{p_t(R|I)}\left[U_t(R, I)\frac{p'_t(R \mid I;\theta)}{p_t(R \mid I)}\right] \\
=&\mathbb{E}_{p(I)}\mathbb{E}_{p_t(R|I)}\left[\exp\left(\log\left(U_t(R, I)\frac{p'_t(R \mid I;\theta)}{p_t(R \mid I)}\right)\right)\right] \\
\geq&\exp\left(\mathbb{E}_{p(I)}\mathbb{E}_{p_t(R|I)}\left[\log\left(U_t(R, I)\frac{p'_t(R \mid I;\theta)}{p_t(R \mid I)}\right)\right]\right) \\
=&\exp\left(\mathbb{E}_{p(I)}\mathbb{E}_{p_t(R|I)}\left[\log U_t(R, I)\right]\right) \cdot e^{-L_t(p_t(R|I), p'_t(R|I;\theta))},
\end{aligned}$$

which implies that, for any utility function, minimizing the task information loss by tweaking θ indirectly maximizes the utility.

In a similar way, assume that there exists a privacy loss function $U_p(S, I)$ (> 0), which encodes the loss to the requester when workers disclose that the sensitive information of instance I is S. The expected additional privacy loss the requester suffers by using the IPP protocol is

$$\mathbb{E}_{p(I)}\mathbb{E}_{p'_p(S|I;\theta)}\left[U_p(S, I)\right] - \mathbb{E}_{p(I)}\mathbb{E}_{p'_p(S;\theta)}\left[U_p(S, I)\right], \tag{4.4}$$

assuming that workers have the background knowledge of the sensitive infor-

mation $p'_p(S;\theta)$. Equation (4.4) is upper-bounded as follows:

$$\begin{aligned}
&\mathbb{E}_{p(I)}\mathbb{E}_{p'_p(S|I;\theta)}\left[U_p(S,I)\right] - \mathbb{E}_{p(I)}\mathbb{E}_{p'_p(S;\theta)}\left[U_p(S,I)\right] \\
=&\sum_{S,I} p'_p(S \mid I;\theta)p(I)\left(1 - \frac{p'_p(S;\theta)p(I)}{p'_p(S \mid I;\theta)p(I)}\right)U_p(S,I) \\
\leq&\max_{S,I} U_p(S,I) - \exp\left(\mathbb{E}_{p'_p(S|I;\theta)p(I)}\left[\log U_p(S,I)\right]\right)e^{-L_p(p'_p)} \\
\leq&\max_{S,I} U_p(S,I) - \left(\min_{S,I} U_p(S,I)\right)e^{-L_p(p'_p)},
\end{aligned}$$

which implies that minimizing the privacy information gain indirectly minimizes the additional privacy loss the requester undergoes.

4.3.6 Breaking the Trade-Off

Given a set of IPP protocols (including protocols with different parameters), it is often a serious problem to determine the protocol we use. This section discusses how to use UPTA to break the trade-off and choose the best one. Among a number of approaches to breaking the trade-off, we especially focus on one of the simplest approaches called a threshold-based approach, which minimizes a privacy risk while preserving a utility loss within a certain threshold. We introduce several formulations of the threshold-based approach based on UPTA.

The basic formulation is to solve the following optimization problem:

$$\begin{aligned}
\text{minimize}_{\theta\in\Theta} \quad & L_p(p'_p(S,I;\theta)) \\
\text{subject to} \quad & L_t(p_t(R \mid I), p'_t(R \mid I;\theta)) \leq \bar{L}_t,
\end{aligned} \tag{4.5}$$

where we denote the maximum tolerable task information loss given by the requester by $\bar{L}_t$. This formulation allows us to obtain the best parameter $\theta_1^\star$ that is not dependent on the definitions of the utility and privacy loss functions, $U_t(R,I)$ and $U_p(S,I)$.

Another formulation is to incorporate the utility function $U_t(R,I)$ the requester has, which boils down to the following optimization problem:

$$\begin{aligned}
\text{minimize}_{\theta\in\Theta} \quad & L_p(p'_p(S,I;\theta)) \\
\text{subject to} \quad & \mathbb{E}_{p(I)}\mathbb{E}_{p_t(R|I)}\left[U_t(R,I)\right] - \mathbb{E}_{p(I)}\mathbb{E}_{p'_t(R|I;\theta)}\left[U_t(R,I)\right] \leq \bar{U}_t,
\end{aligned} \tag{4.6}$$

where we denote the maximum tolerable task information loss given by the requester by $\bar{U}_t$. The optimization problem (4.6) is advantageous to the optimization problem (4.5) when the requester has a solid definition of the

utility function. Further, if the requester is familiar with the utility function rather than the task information loss, the formulation (4.6) is easier for the requester to determine the threshold $\bar{U}_t$ than $\bar{L}_t$. Note that it is possible to substitute the expected utility by its lower-bound, which leads to the first optimization problem. This substitution provides an easy way to set the maximum tolerable task information loss $\bar{L}_t$ from $\bar{U}_t$.

In addition to the threshold-based approach, it is possible to select the protocol by optimizing the profit of the requester:

$$\begin{aligned}\text{maximize}_{\theta\in\Theta}\ \ & \mathbb{E}_{p(I)}\mathbb{E}_{p'_t(R|I;\theta)}\left[U_t(R,I)\right] \\ & - \left(\mathbb{E}_{p(I)}\mathbb{E}_{p'_p(S|I;\theta)}\left[U_p(S,I)\right] - \mathbb{E}_{p(I)}\mathbb{E}_{p'_p(S;\theta)}\left[U_p(S,I)\right]\right).\end{aligned} \tag{4.7}$$

Considering that the objective function (4.7) is lower-bounded by

$$\begin{aligned}&\exp\left(\mathbb{E}_{p(I)}\mathbb{E}_{p_t(R|I)}\left[\log U_t(R,I)\right]\right)\cdot e^{-L_t(p_t(R|I),p'_t(R|I;\theta))} \\ &- \left(\max_{S,I} U_p(S,I) - \left(\min_{S,I} U_p(S,I)\right)e^{-L_p(p'_p)}\right),\end{aligned} \tag{4.8}$$

it is also possible to substitute the objective function (4.7) by the lower-bound (4.8).

The use of lower-bounds has two advantages. First, the lower-bounds have much fewer parameters than the original objective functions, and therefore, it is easy for the requester to configure the parameters. Second, we are able to estimate the objective function just given the task information loss and privacy information gain. This advantage allows the requester to select the IPP protocols preliminarily given the task information loss and privacy information gain other researchers have computed.

4.4 Instance Clipping Protocol

We present an instance clipping (IC) protocol, which is used in a case study of UPTA. The IC protocol preserves instance privacy by clipping an instance with a fixed-size window, instead of relying on a blank template (Little and Sun, 2011). In this sense, the IC protocol is regarded as a generalization of the method proposed by Little and Sun (2011). Figure 4.2 illustrates the IC protocol along with the terminology we use. This section gives the formal description of the IC protocol as well as the qualitative analysis of the protocol, which will be examined in the experiment using UPTA.

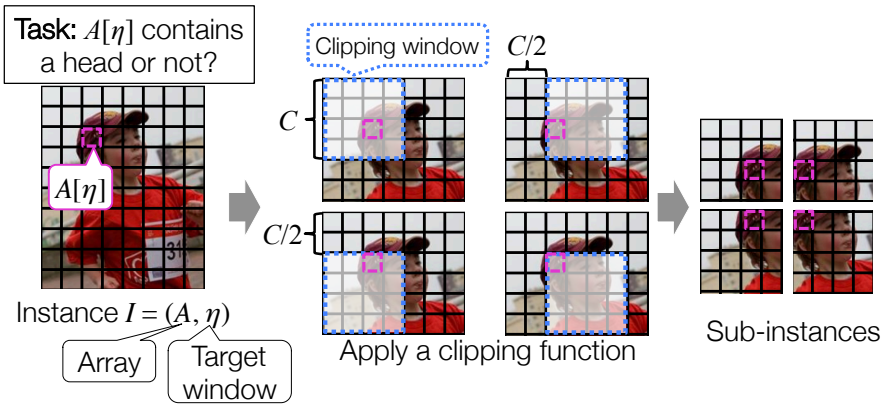


Figure 4.2: Illustration of the IC protocol. **(Left)** An instance consists of a pair of array A and target window η. A worker is asked to check whether $A[\eta]$ (the rectangle framed by magenta broken lines) contains a head or not. **(Middle)** Given instance $I = (A, \eta)$, a clipping window of size C (a rectangle framed by blue dotted lines) is moved in steps of $C/2$ so long as the clipping window contains the target window. A clipping function clips the instance with each clipping window to generate sub-instances. **(Right)** All the sub-instances are used to submit a task.

4.4.1 Task Assumption: Array-Labeling Task

The target task of the IC protocol is an array-labeling task: a task to examine whether part of an array indicated by a window satisfies a particular condition or not. A number of tasks belong to this class of tasks, including a task to transcribe an audio recording, a task to detect objects in an image, and a task to digitize hand-written documents. As an illustration, Figure 4.2 **(left)** depicts a task to detect human heads in an image.

In the following, we give the formal definition of the array-labeling task. We assume that an instance $i = (A, \eta)$ consists of a D-dimensional array A and a window η on the array, which is defined as a set of indices of the array. We call A an array and η a target window. We further assume that a result R is a label on the sub-array $A[\eta]$, where $A[\eta]$ denotes the sub-array clipped by η. In Figure 4.2 **(left)**, an array corresponds to an image, a target window to a specific region of the image (the rectangle framed by magenta broken lines), and a result to a label indicating whether $A[\eta]$ contains a head or not.

4.4.2 Main Protocol

The IC protocol receives an instance (A, η) as an input and outputs a set of results. It first clips the instance using a clipping function to generate multiple sub-instances as shown in Figure 4.2. Then, it asks multiple workers to perform the task on the sub-instances. Finally, it outputs the results obtained from the workers. In the following, we first define the clipping function, and then, describe the protocol in details.

A clipping function, given instance $i = (A, \eta)$ and a clipping window of size C, generates a sub-instance by clipping the instance with the clipping window. Definition 4.5 formalizes the notion of the clipping function. We require that the clipping window includes the target window η.

Definition 4.5 (Clipping function)**.** *Given instance $i = (A, \eta)$, and a clipping window ω such that $\omega \supseteq \eta$, a clipping function $\phi(i; \omega)$ is defined as*

$$\phi(i; \omega) := (A[\omega], \eta) \ (=: i[\omega]).$$

We call $A[\omega]$ a sub-array and $i[\omega] = (A[\omega], \eta)$ a sub-instance.

Then, the IC protocol is described as shown in Protocol 4.2. At the initial state, a requester has instance $i = (A, \eta)$ and fixes the window size to C pixels. First, a clipping window of C pixels is moved over array A in steps of $C/2$ pixels, and the clipping function is applied when clipping window ω contains target window η, generating sub-instances (lines 1–6 in Protocol 4.2). Then, the requester submits the task using all of the sub-instances, and workers process them as well as extract sensitive values from them (line 7–12). Finally, the requester, receiving the results from the workers, regards all of them as the sample from the task execution model (line 14).

We move the clipping window in steps of $C/2$ so as to ensure that the resultant sub-instances have overlapped areas. Figure 4.2 **(right)** shows that all the sub-instances have overlaps around the target window; without them, a target object can be divided into two sub-instances, which can degrade the task performance severely. For example, in a head detection task, a head can be divided into two without overlaps, from which it is difficult to recognize that there exists a head.

4.4.3 Applicability

We discuss the applicability of the IC protocol from a qualitative point of view. We first provide the general guideline showing when the IC protocol works, and then, we review several examples to which the IC protocol is suitable.

Protocol 4.2 Instance Clipping (IC) Protocol.

Inputs: instance $i = (A, \eta)$ and the size of a clipping window C.
Output of a requester: a set of results.
Output of a worker: sensitive information.

1: $\Omega_C \leftarrow \emptyset$. ▷ Initialize a set of valid clipping windows.
2: **for** clipping window ω defined in steps of $C/2$ **do**
3: **if** $\omega \supseteq \eta$ **then**
4: $\Omega_C \leftarrow \Omega_C \cup \{\omega\}$.
5: **end if**
6: **end for**
7: The requester submits the task with the sub-instances $\{i[\omega] \mid \omega \in \Omega_C\}$.
8: **for** $\omega \in \Omega_C$ **do**
9: A worker is randomly selected from a pool of workers.
10: The worker samples $r^{(\omega)}$ from $p_t(R \mid I = i[\omega])$.
11: The worker returns $r^{(\omega)}$ to the requester.
12: The worker extracts a sensitive value $s^{(\omega)}$ from $p_p(S \mid I = i[\omega])$.
13: **end for**
14: The requester regards $\left\{r^{(\omega)}\right\}_{\omega \in \Omega_C}$ as samples from $p'_t(R \mid I = i; C)$.
15: The workers regard $\left\{s^{(\omega)}\right\}_{\omega \in \Omega_C}$ as samples from $p'_p(S \mid I = i; C)$.

General Guideline

From the qualitative analysis on the IC protocol, we arrive at the hypothesis that the locality of task and privacy definitions plays an important role in the performance of the IC protocol.

The locality of a task is defined as the size of the part of an array that is necessary to perform the task with satisfactory quality. For example, a task to detect a small object in an image is local because a worker can detect it even if s/he is shown an image clipped around the object, while a task to summarize a text is global because a worker has to check the whole sequence of the text. The locality of a privacy is defined similarly. If a face in an image is defined as sensitive information, the privacy definition is local because a small part of the image can leak it; if the sensitive information is the abstract of a meeting recorded in an audio file, the privacy definition is global because a worker has to check the entire sequence of the audio file.

Our hypothesis on the performance of the IC protocol is that the IC protocol is suitable to a pair of a local task and a global privacy definition. Since the IC protocol clips an instance using a small window, the task needs to be local so as not to degrade the task performance. The privacy definition needs to be global in order to preserve instance privacy; otherwise, the sensitive

information can be extracted even from a clipped instance. This hypothesis is validated quantitatively in the experiment using UPTA.

Examples

We give three examples for which the IC protocol is suitable according to the previous hypothesis. The first example is a task to detect a head in an image. Sensitive information can be defined as the association between a person in an image and his/her context including location, activities, and companions. The task is local because the area of a head is usually small compared to the size of the whole image. On contrary, the privacy definition is not local because the association of a person and his/her context often requires a large part of an image to infer compared to the head detection task. Therefore, the task and privacy definitions will be suitable to the IC protocol. We use this example in the experiment.

The second example is a task to transcribe an audio recording of a meeting, and the third one is a task to digitize a handwritten document. Sensitive information is the abstract of the meeting or the document, which cannot easily be inferred from a local part of the recording or the scanned image. On contrary, the task can be performed even with a clipped array (a segment of an audio file or a document) if it contains at least a few words.

4.5 Experiments

We conduct two experiments using a real crowdsourcing platform to demonstrate the effectiveness of UPTA. We employ the head detection task and define the activity of a person as instance privacy, and we execute the IC protocol on this pair of task and privacy definitions. The first experiment applies UPTA to analyze the properties of the IC protocol. The second experiment investigates the consistency of UPTA with standard performance measures to validate the performance measures of UPTA.

4.5.1 Task and Privacy Definitions and Dataset

We employ a dataset, a task, and a privacy definition to which the IC protocol is suitable according to the general guideline provided in the previous section.

Dataset

We use the Stanford 40 Action Dataset (Yao et al., 2011), which contains images of humans performing actions belonging to forty classes. We selected

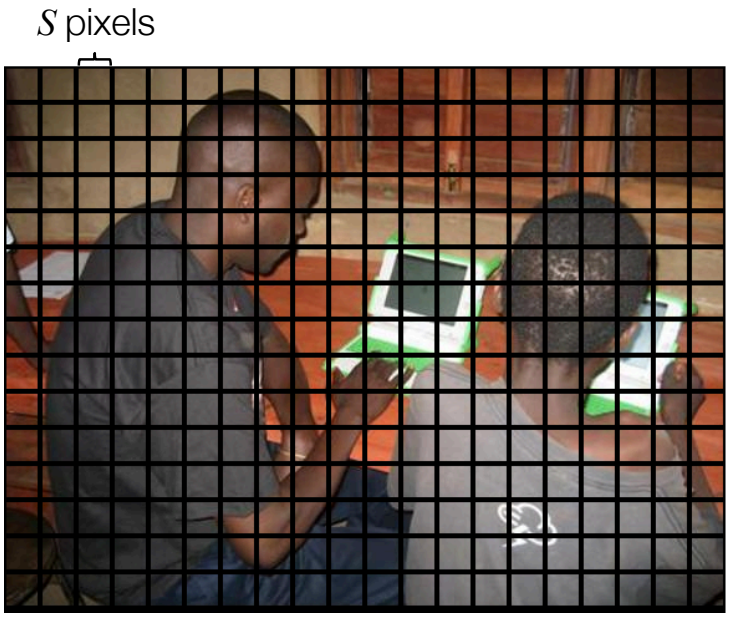


Figure 4.3: Conversion from a head detection task to an array-labeling task. An image is discretized into blocks of $S \times S$ pixels as depicted by the black lattice. Then, an array-labeling version of the task is, given the image, to check whether each block contains a head or not (each block of the lattice corresponds to a target window η).

ten classes: `cooking`, `fishing`, `running`, `throwing_frisby`, `watching_TV`, `feeding_a_horse`, `playing_guitar`, `texting_message`, `using_a_computer`, and `writing_on_a_book`. For each class, we selected fifty images in which all the humans were engaged in the same action specified by the class name. In total, we selected 500 pairs of images and action labels and used them for the experiment. All the images were resized to fit in 500×500 pixels. Note that all the photographs in this chapter are borrowed from this dataset.

Task Definition

We employ a head detection task, a task to detect areas containing human heads in a given image. In order to apply the IC protocol, we convert the head detection task into an array-labeling task as follows. First, we divide an image into blocks of $S \times S$ pixels as shown in Figure 4.3, where blocks are illustrated by lattices of black lines.[4] Then, we create a set of array-labeling tasks by regarding each block as a target window. The resultant task is, given

[4]We set $S = 25$ pixels.

an image and a target window (one $S \times S$ pixels block of the lattice), to judge whether the target window contains a human head ($R = 1$) or not ($R = 0$).

Privacy Definition

We define the sensitive information as the association between a human and his/her action. By assuming that a human is identifiable by his/her head, the instance privacy is preserved when a worker cannot infer the action context from a sub-array that contains a head. For example, if the worker cannot infer the action context `running` from an image containing a head of a running woman, then the instance privacy is preserved.

4.5.2 Experimental Setting

This section describes the detailed computation procedures of the task information loss and the privacy information gain.

Task Execution

The computation of the task information loss requires the execution of both the IC protocol and the corresponding NPP protocol. We executed the head detection task through the NPP protocol and the IC protocol with different parameters.

The IC protocol was executed with the interface shown in Figure 4.4. The image posed to a worker was composed by combining multiple sub-arrays of $C \times C$ pixels so that the size of the image was roughly 500×500 pixels as shown in Figure 4.5. The labels acquired from the worker are regarded as the outputs of the IC protocol. We allocated one worker per composed image, and therefore, large part of images were labeled by four workers.[5] A reward of 0.5 yen was given for the completion on a task with a composed image. We repeated this procedure, varying C from 50 pixels to 300 pixels in steps of 50 pixels.

We executed the NPP protocol with the interface shown in Figure 4.4, replacing the composed image with the image shown in Figure 4.3. A worker is instructed to select all the blocks of $S \times S$ pixels by clicking them. The labels acquired from the worker are regarded as the outputs of the NPP protocol. We allocated one worker per image and gave a reward of one yen per image.

The task information loss was estimated empirically using two smoothing parameters, $\tau = 0.1$ and 0.01. For each $C \in \{50, 100, \ldots, 300\}$, we

[5] As shown in Figure 4.2 **(right)**, each instance generates four sub-instances in general.

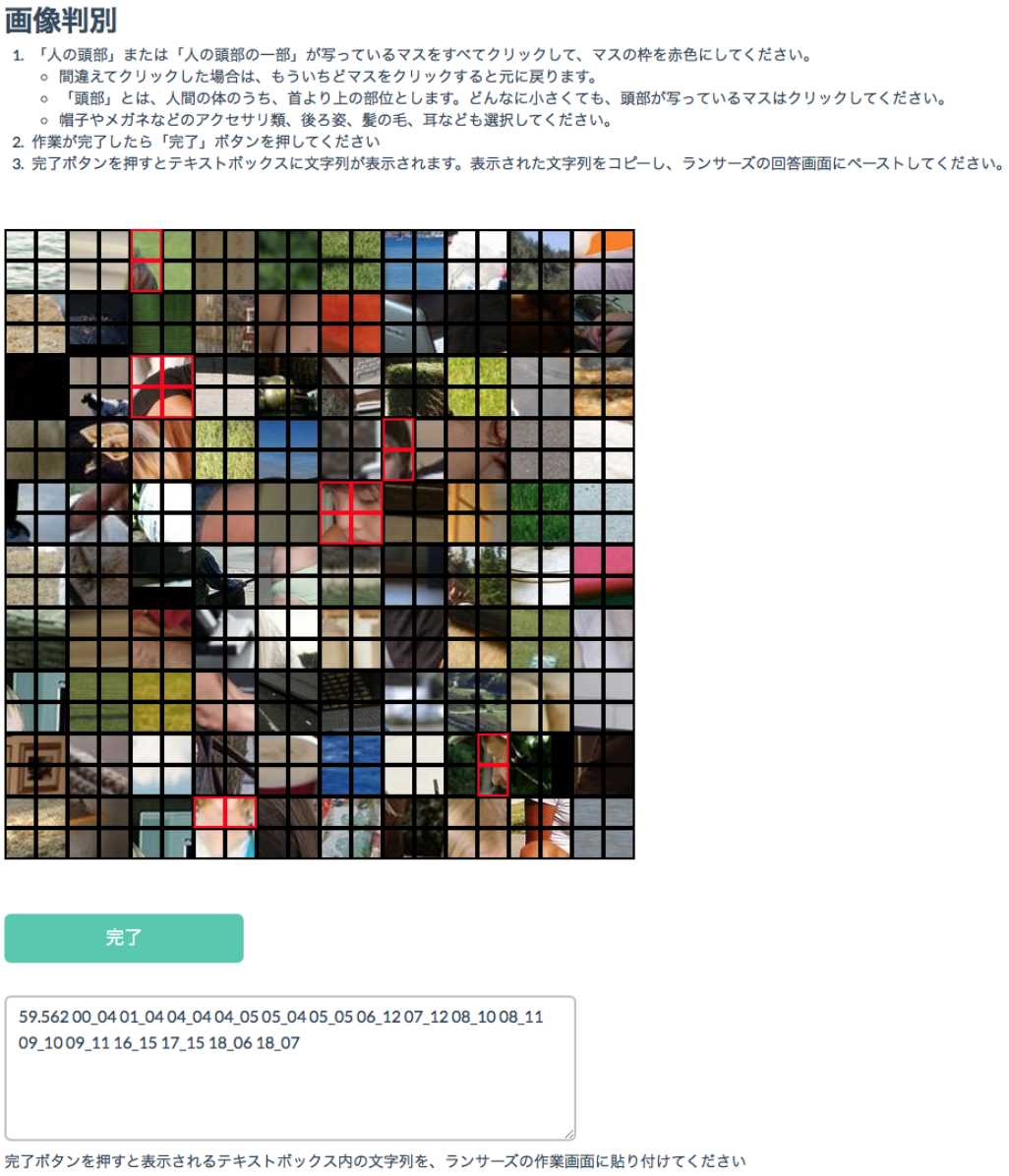


Figure 4.4: Interface for the head detection task used in a real crowdsourcing platform. The job instruction is as follows. *(1) Click all the blocks containing a head or part of it to turn them into red. If you misclick a block, click it again to unselect it. The definition of a head is part of a human body above a neck, including accessories such as a cap and glasses, hair, and ears. (2) If you finish the task, press the green button "finish". (3) Copy the text shown in the text box below, and paste it to the text box in a crowdsourcing platform.* The third step is arranged because the platform does not allow the use of JavaScript; the interface is put on our server, and the platform is used only for recruiting workers and collecting results.

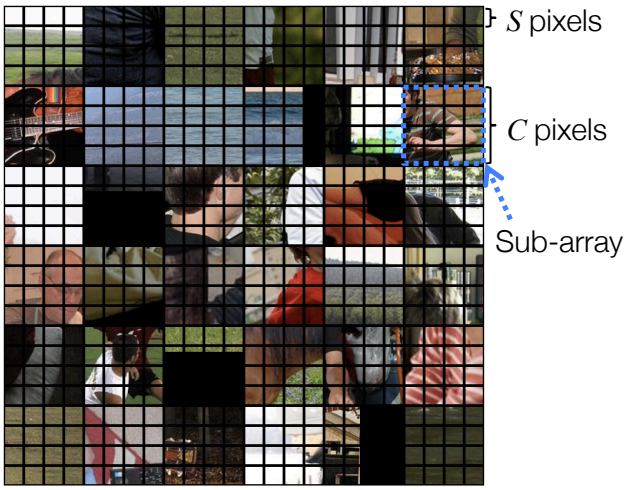


Figure 4.5: We combine multiple sub-arrays of $C \times C$ pixels to execute the IC protocol at low cost. A worker is instructed to select all the blocks of $S \times S$ pixels that contain (part of) a head by clicking them.

calculated the task information loss using all the results following Equations (4.1) and (4.2).

Privacy Invasion

The computation of the privacy information gain requires the execution of the IC protocol only. We simulated the privacy invasion procedure using ten-choice questions with the interface shown in Figure 4.6. A worker is shown a sub-array of $C \times C$ pixels as well as ten choices of the action labels and is asked to assign an appropriate label to the sub-array. For each action label, we randomly chose twenty-five sub-arrays from the sub-arrays that were judged to contain a head in the previous experiment; in total, we had 250 sub-instances. We assigned fifty workers to each question. A reward of 0.2 yen was given to a worker for answering one question. We repeated the procedure above, varying C from 50 pixels to 300 pixels in steps of 50 pixels.

The privacy information gain was empirically estimated using two smoothing parameters, $\tau = 0.1$ and 0.01. For each $C \in \{50, 100, \ldots, 300\}$, we calculated the privacy information gain using all the results following Equations (4.1) and (4.2), where we substitute $p'_p(S)$ by

$$p'_p(S) = \mathbb{E}_{p(I)}[p'_p(S \mid I)].$$

Figure 4.6: Interface for the simulated privacy invasion task used in a real crowdsourcing platform. The job instruction is as follows. *Select the most appropriate action from the drop-down list that the human in the image is engaged in. If the image should not contain any part of a human, then leave the list untouched. We basically accept all the task results except for obviously dishonest results.*

4.5.3 Utility-Privacy Trade-Off

The first experiment analyzes the trade-off between utility and privacy of the IC protocol to investigate the applicability of it and discusses the parameter selection of it based on UPTA.

Utility and Privacy

Figures 4.7 and 4.8 respectively show the task information loss scores and the privacy information gain scores on different clipping window sizes $C \in \{50, 100, \ldots, 300\}$. We have three findings from this result.

First, different smoothing parameters result in different scores for both cases, they show the same trend across the two parameters. Therefore, we

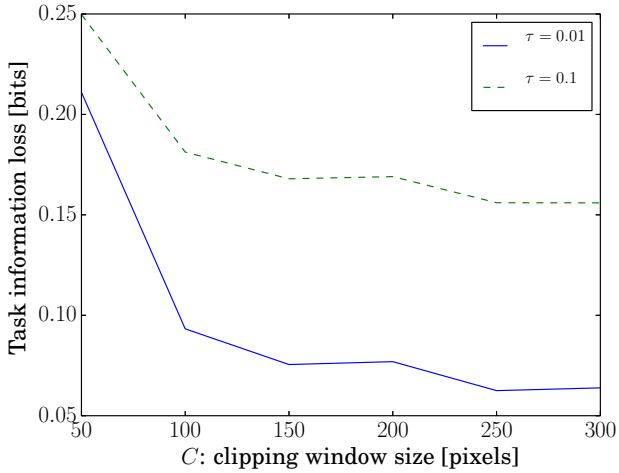


Figure 4.7: Task information loss scores for different clipping window sizes. The x-axis corresponds to the clipping window size, and the y-axis to the task information loss.

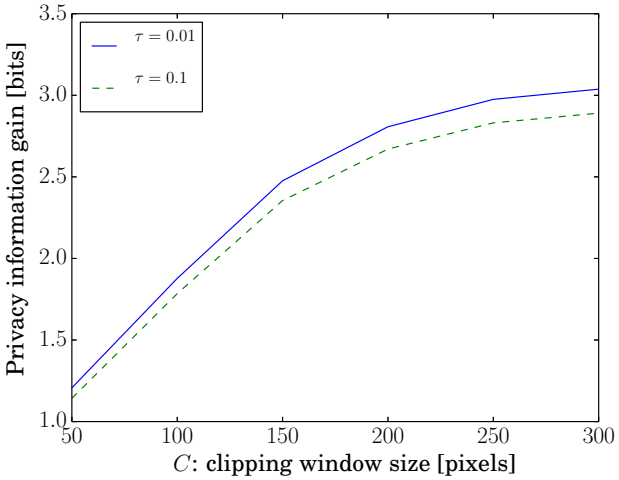


Figure 4.8: Privacy information gain scores for different clipping window sizes. The x-axis corresponds to the clipping window size, and the y-axis to the privacy information gain.

Table 4.1: Scores of the task information loss and privacy information gain ($\tau = 0.01$) and the profit of the requester. Let $\tilde{U}_t := \exp(\mathbb{E}_{p(I)}\mathbb{E}_{p_t(R|I)}[\log U_t(R, I)])$, $\bar{U}_p := \max_{S,I} U_p(S, I)$, and $\underline{U_p} := \min_{S,I} U_p(S, I)$.

C [pixels]	Task info. loss [bits]	Privacy info. gain [bits]	Profit (4.8)
50	0.211	1.21	$0.864\tilde{U}_t + 0.432\underline{U_p} - \bar{U}_p$
100	0.0932	1.88	$0.937\tilde{U}_t + 0.272\underline{U_p} - \bar{U}_p$
150	0.0755	2.48	$0.949\tilde{U}_t + 0.179\underline{U_p} - \bar{U}_p$
200	0.769	2.81	$0.948\tilde{U}_t + 0.143\underline{U_p} - \bar{U}_p$
250	0.0625	2.98	$0.958\tilde{U}_t + 0.127\underline{U_p} - \bar{U}_p$
300	0.0639	3.04	$0.957\tilde{U}_t + 0.122\underline{U_p} - \bar{U}_p$

conclude that the smoothing parameter does not have much impact on the trend of the scores. Second, as C increases, the task information loss decreases almost monotonically, and the privacy information gain increases monotonically. This matches intuition; both the task execution and privacy invasion become easier for larger sub-instances. This observation supports the validity of UPTA. Note that the outlier value of the task information loss at $C = 200$ pixels is caused by one outlier worker, which is analyzed in Section 4.5.4. Finally, we observe that the task information loss is almost the same when C is larger than 100 pixels, while the privacy information gain increases gradually as we increase C. In specific, in case of $\tau = 0.01$, the task information loss at $C = 100$ pixels is only 1.1 times larger than the score at $C = 300$ pixels, while the privacy information gain at $C = 100$ pixels is 0.6 times larger than the score at $C = 300$ pixels. Therefore, we conclude that the IC protocol can preserve the utility while preserving instance-privacy.

Parameter Selection

We provide a use case of UPTA to select the clipping window size C. For simplicity, we select it by maximizing the lower-bound of the profit (4.8). Table 4.1 summarizes the scores of the task information loss and privacy information gain when $\tau = 0.01$ as well as the profit scores. If we substitute $(\tilde{U}_t, \underline{U_p}, \bar{U}_p) = (1, 0.2, 2)$, setting $C = 100$ pixels achieves the best profit -1.01, if we substitute $(\tilde{U}_t, \underline{U_p}, \bar{U}_p) = (1, 0.5, 2)$, setting $C = 50$ pixels achieves the best profit -0.92, and if we substitute $(\tilde{U}_t, \underline{U_p}, \bar{U}_p) = (1, 0.01, 2)$

setting $C = 250$ pixels achieves the best profit -1.04. Although the results are different when we use different utility and privacy loss functions, we observe that the profit is one effective criterion to determine the parameter of the IPP protocol.

4.5.4 Consistency of UPTA with Standard Measures

The second experiment examines the consistency of the task information loss with standard measures, *i.e.*, precision, recall, and the F-measure. If the standard measures show a similar tendency to the task information loss, we can validate the task information loss. Note that, although we can apply the standard measures for a binary array-labeling task, we have to devise other measures for other tasks, and it is sometimes impossible to apply standard measures especially in case of a subjective task. In addition, the privacy information gain has no alternative. Therefore, this experiment does not devalue the performance measures of UPTA.

Analytic Method

For each $C \in \{50, 100, \ldots, 300\}$, we first aggregated multiple labels given to each instance, and then, computed precision, recall, and the F-measure on the aggregated labels of all the instances, regarding the results of the NPP protocol as the ground truths.

We used three popular label aggregation methods. Letting us denote a label given by worker j to instance i by $r_{i,j} \in \{0, 1\}$ and an aggregated label of instance i by $r_i \in \{0, 1\}$, we describe the aggregation procedure of multiple labels $\{r_{i,j}\}_{j \in \mathcal{J}}$ into r_i.

(i) OR method
Multiple labels are aggregated by taking their logical disjunction. The aggregated label is 1 if there exists at least one label 1 in the multiple labels and 0 if all the labels are 0:

$$r_i = \begin{cases} 1 & \text{if } \sum_{j \in \mathcal{J}} r_{i,j} \neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

(ii) Majority Voting (MV) method
Multiple labels are aggregated by majority vote, where ties are broken

uniformly randomly:

$$r_i = \begin{cases} 1 & \text{if } \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} r_{i,j} > \frac{1}{2}, \\ 0 & \text{if } \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} r_{i,j} < \frac{1}{2}, \\ \text{Bernoulli}\left(\frac{1}{2}\right) & \text{otherwise.} \end{cases}$$

(iii) Latent Class (LC) method
The LC method is commonly used for the purpose of quality control in crowdsourcing (Dawid and Skene, 1979). It aggregates the labels by a weighted majority voting strategy based on the ability of each worker; a label given by a low-ability worker has less influence on the aggregation than a label given by a high-ability worker. The detailed algorithm will be reviewed in Section 5.2.2.

Result

Figure 4.9 shows the standard measures on different clipping window sizes $C \in \{50, 100, \ldots, 300\}$. First, we notice that the trends of the scores of the standard measures are the same as those of the task information loss shown in Figure 4.7 in that (i) the score is almost the same when C is larger than 100 pixels (OR, MV, LC), and (ii) the score at $C = 200$ pixels is an outlier (OR, MV). Therefore, we conclude that the task information loss is consistent with the standard measures, which supports the validity of its use.

Then, we investigated the precision score failure at $C = 200$ pixels when we apply the OR and MV methods. We find that the failure is caused by the ability of the workers. Figure 4.10 shows the estimated ability parameters of workers in the LC method. The right frame in Figure 4.10 shows the probability of each worker assigning label 1 when the estimated true label is 1, which is related to the recall scores. We denote the probability of worker j by α_j. The middle frame shows the probability of each worker assigning label 0 when the estimated true label is 0, which is related to the precision scores. We denote the probability of worker j by β_j. In the middle frame, while the ability parameters of most of the workers are high, the ability of worker 19 is quite low at $C = 200$ pixels. Considering that the number of labels 0 is much bigger than the number of labels 1 in this task, worker 19 gave a significant amount of labels 1 to sub-instances that should have been labeled 0. Therefore, we conclude that the failure is caused by the low precision of worker 19. The LC method does not suffer from worker 19 because it aggregates labels taking the ability of the workers into account, while the other methods cannot handle them, which leads to the results.

4.6 Summary and Future Work

This chapter has discussed the issue of instance privacy in crowdsourcing and has introduced the utility-privacy trade-off analyzer (UPTA), which enables us to evaluate the performance of an instance-privacy preserving (IPP) protocol. Our idea to quantify the utility and privacy is to model the task execution and privacy invasion as sampling from respective probability distributions. The models can be empirically estimated by using crowdsourcing, and once we have estimated the models, we are able to compute divergence-based performance measures. As a case study, we have applied UPTA to the instance clipping (IC) protocol and have investigated the properties of the IC protocol and UPTA. The experimental results show that the IC protocol can balance the utility-privacy trade-off and UPTA is consistent with existing performance measures, which validates the formulation of UPTA.

An interesting research direction is to extend the IC protocol. It is possible to replace the clipping function with any instance transformation function, which transforms an instance to preserve privacy. For example, a function to add noise on an instance or that to blur an image instance will be suitable. It is also possible to select the clipping window size adaptively. For example, a protocol starts with a small clipping window and expands as necessary until workers can perform tasks.

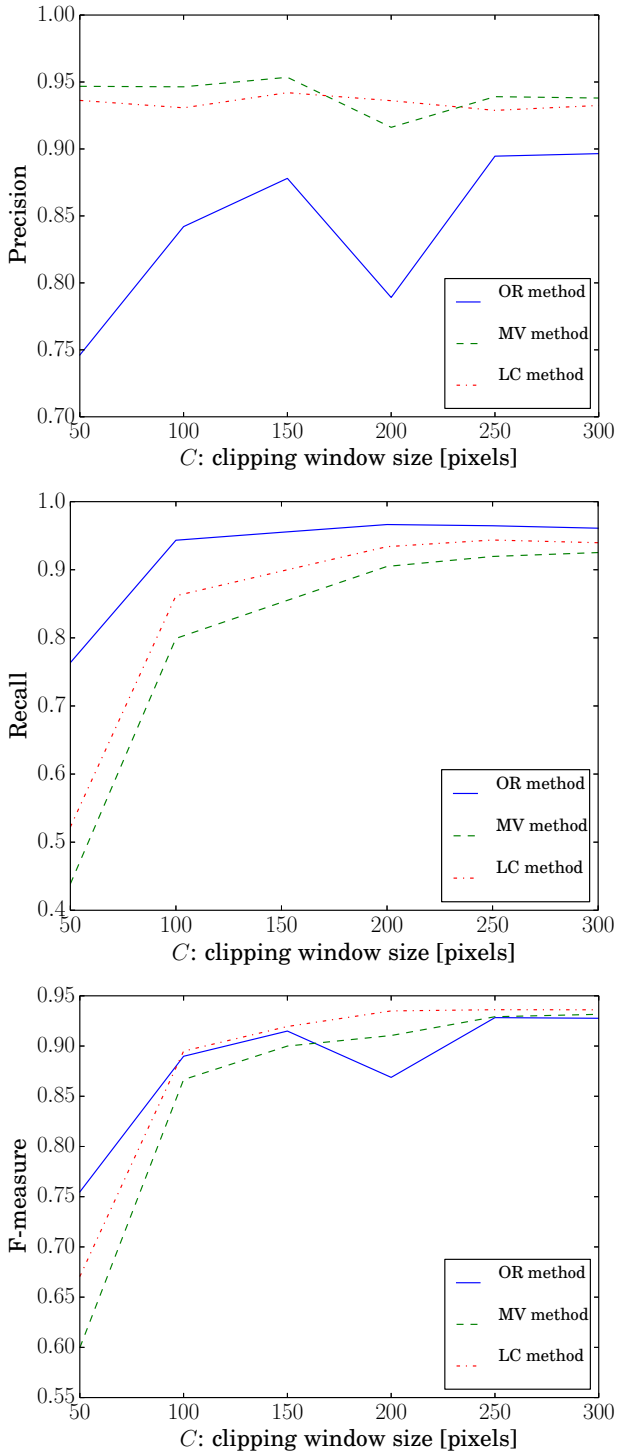


Figure 4.9: Precision **(left)**, recall **(middle)**, and the F-measure scores **(right)** for different window sizes. The x-axis corresponds to the clipping window size, and the y-axis to each score.

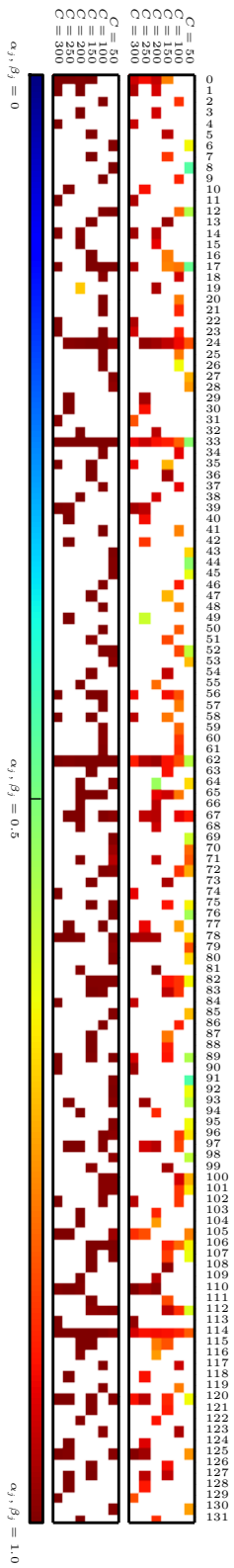


Figure 4.10: Estimated ability parameters of the workers. The right and mid frames show the ability parameters of each worker α_j and β_j. The x-axis corresponds to a result with a particular window size C, the y-axis corresponds to worker IDs j, and each element corresponds to the value of α_j **(right)** or β_j **(middle)** at each C. The left frame shows a colormap. The white elements indicate that the worker did not perform the tasks.

Chapter 5

Worker-Privacy Preservation

5.1 Introduction

The quality of a task result is one of the central issues in crowdsourcing. The quality is not guaranteed because workers may not be skilled at the task or the task design is poor. Figure 5.1 depicts a small example of such results, where three workers were engaged in the same image labeling task. Worker w_1 gave the correct labels, whereas worker w_2 sometimes failed, and worker w_3 seemed to return non-informative labels because all the labels given by worker w_3 were the same. To make full use of crowdsourcing, it is necessary to cope with the variable quality of task results.

A basic approach to the quality issue is to collect multiple redundant labels from different workers as shown in Figure 5.1 and infer the true labels by aggregating them (Sheng et al., 2008; Dawid and Skene, 1979). One of the most popular methods is the latent class method (LC method) (Dawid and Skene, 1979), where the true labels are estimated by inferring the model of labeling processes of workers. They model a worker equipped with ability parameters such that the worker outputs the true label with probability proportional to the ability parameters. The true labels are inferred using the EM algorithm in a form of weighted majority voting based on each worker's ability. In other words, by using the LC method, not only the true labels but also the abilities of workers can be estimated.

We first point out that this procedure can lead to invade the privacy of workers; the results returned from workers can be used to infer the attributes of workers, which can be sensitive information about them. If workers recognize that their privacy is at risk, some workers will be reluctant to participate in crowdsourcing, which severely damages the continuation of crowdsourcing. In order to clarify this risk, let us illustrate three examples of the privacy

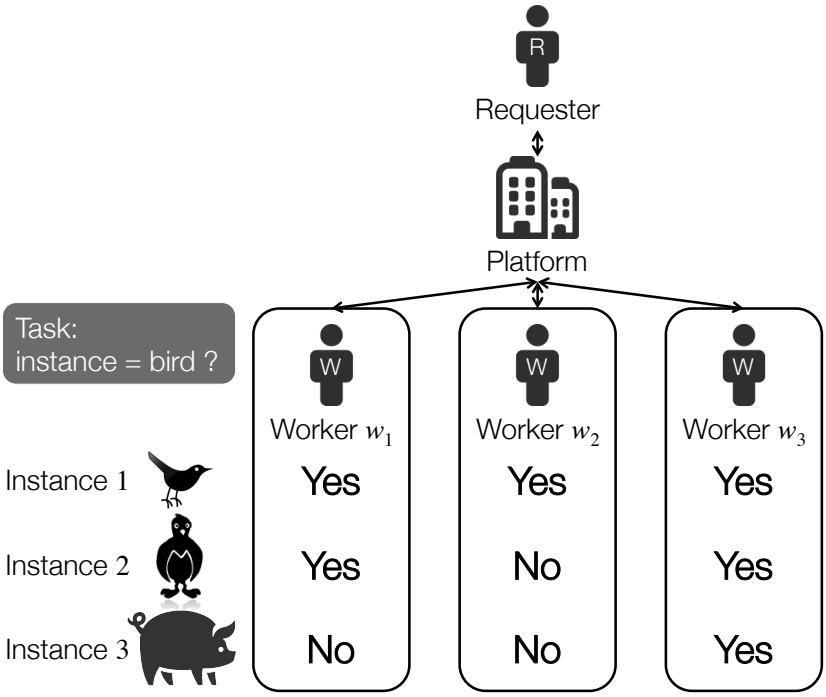


Figure 5.1: Illustrative model of crowdsourcing and our protocol. We assume an image labeling task. Each column in the table shows labels given by each worker to the images, which do not necessarily agree with unobservable ground truths. The lines between entities indicate feasible communication in crowdsourcing. We develop a protocol where a requester can estimate the true labels by communicating with each worker who secretly holds his own labels via the platform.

invasion. The first example is a location-based task where workers are asked to submit location information of specific objects. For example, AED4[1] asks workers to submit the locations of automated external defibrillators (AEDs) in order to create a location map of AEDs. Such location data possibly reveal the trajectory of each worker, and therefore, the privacy of workers can be invaded. The second example is a questionnaire task where workers are asked to fill in questionnaires. Crowdsourcing allows us to cut down the efforts and monetary and time costs to recruit many participants. The raw data contain much personal information, and thus, the privacy of workers will be invaded. Although a single answer of questionnaires contains little personal information, combining simple information can sometimes identify people uniquely as pointed out in the field of privacy-preserving data mining (Sweeney, 2002).

[1] `http://www.aed4.eu/`

The last example is a volunteer task (or a non-paid task) such as that in Galaxy Zoo and AED4, which is a popular form of crowdsourcing. In such a task, some workers will feel uncomfortable to sacrifice their privacy without any reward. These examples clearly show that the privacy of workers can be invaded from the raw results returned from the workers. Addressing the privacy issue can encourage more workers to participate in crowdsourcing who could not participate because of these privacy problems. Therefore, it is important to preserve the privacy of workers.

A main research challenge is how to resolve the conflicting demands of requesters and workers. As stated above, workers wish to avoid from handing their results to requesters; requesters wish to receive the results from workers. Our observation aiming to address this dilemma is that requesters do not necessarily have to know the raw results; it is sufficient for them to know aggregated results of high quality. Based on this observation, our idea to confront the research challenge is to develop a privacy-preserving variant of a quality control method. If a requester can obtain aggregated results for all the instances without letting any entity know the raw results, it will satisfy both conflicting requirements of requesters and workers.

In this chapter, we first formalize the research challenge above to define a worker-private quality control problem, whose goal is that a requester infers the true labels whilst preserving the privacy of workers. Then, we present our solution called a worker-private latent class protocol (WPLC protocol). The WPLC protocol allows workers to keep their labels and ability parameters private, which prevents privacy invasion. The key ideas of the WPLC protocol are twofold: decentralization of computation and introduction of secure computation. By taking workers into computation and introducing secure computation, a requester can infer the true labels while workers can hide their labels. To validate the WPLC protocol, we provide a theoretical guarantee of its security and investigate its computational efficiency and accuracy experimentally.

5.2 Quality Control Problem

We first define a quality control problem and review an existing method called the latent class method (Dawid and Skene, 1979).

5.2.1 Problem Setting

We introduce the notation we use in this chapter and then, we define the quality control problem in Problem 5.1.

We assume that participants in crowdsourcing consist of a single requester and J workers for simplicity. We also assume that the requester submits a binary labeling task, where a task is to give a binary label to an instance. The requester has a set of task instances $\mathcal{I} := \{1, \ldots, |\mathcal{I}|\}$ and is willing to know the true label $y_i \in \{0, 1\}$ for each instance $i \in \mathcal{I}$. Let $\mathcal{Y}^\star = \{y_i \mid i \in \mathcal{I}\}$ be a set of the true labels, which is unknown to both the requester and workers.

Each worker $j \in \{1, \ldots, |\mathcal{J}|\}(=: \mathcal{J})$ gives an unreliable label $y_{i,j} \in \{0, 1, \perp\}$ to each instance i, which we call a crowd label. For convenience, we introduce a label $\perp$; $y_{i,j} = \perp$ indicates that worker j does not give a label to instance i. We can assume that each worker gives labels including $\perp$ to all the instances. We call label $\perp$ a null label and labels $\{0, 1\}$ valid labels in this chapter. Since a crowd label is unreliable, $y_{i,j} = y_i$ does not always hold for valid labels. Let $\mathcal{Y} = \{y_{i,j} \mid i \in \mathcal{I}, j \in \mathcal{J}\}$ be the set of all the crowd labels.

Then, the quality control problem is defined as Problem 5.1, where the requester aims to infer the true labels from the crowd labels $\mathcal{Y}$ collected from workers.

Problem 5.1 (Quality control problem)**.** *Assume that the requester has crowd labels $\mathcal{Y}$. The goal of the quality control problem is that the requester infers the true labels $\mathcal{Y}^\star$ from the crowd labels $\mathcal{Y}$ correctly.*

Note that this problem setting can be extended to deal with multi-class or real-valued labels. For simplicity, we focus on the binary labeling task, and the extensions are described in Appendix B.1.

5.2.2 Latent Class Method

We review the LC method (Dawid and Skene, 1979), which is a standard method for Problem 5.1. The LC method employs an unsupervised algorithm to aggregate multiple labels given to each instance by taking the ability of a worker into account, and outputs the aggregated labels as the estimates of the true labels.

Latent Class Model

We first introduce a latent class model, which the LC method assumes in order to aggregate crowd labels. The LC model assumes that the unreliability of crowd labels comes from on the ability of the worker who produces the labels. The crowd labels $\mathcal{Y}$ are assume to be generated by the following model. Each instance $i \in \mathcal{I}$ has the single unobservable true label y_i. Each

worker $j \in \mathcal{J}$ independently gives label $y_{i,j}$ to instance i according to

$$\alpha_j = p(y_{i,j} = 1 \mid y_i = 1; \theta_j),\ \beta_j = p(y_{i,j} = 0 \mid y_i = 0; \theta_j),$$

where let $\theta_j = \{\alpha_j, \beta_j\}$ be the ability parameters of worker j. It is also assumed that for each instance $i \in \mathcal{I}$, y_i is generated according to

$$p(y_i; p) = p^{y_i} \cdot (1-p)^{1-y_i}.$$

Let $\Theta = \{p\} \cup \{\theta_j \mid j \in \mathcal{J}\}$ be the set of all the model parameters.

EM Algorithm for Inference

The LC method employs an EM algorithm to estimate the model parameters Θ as well as the latent variables $\{y_i \mid i \in \mathcal{I}\}$. It repeats an expectation-step (E-step) and a maximization-step (M-step) alternately until convergence. Intuitively, the E-step updates the true labels using majority voting weighted by estimated ability parameters of the workers, and the M-step updates the ability parameters of the workers using the true labels. Repeating these two steps is proven to increase the likelihood function (Dempster et al., 1977), which validates the use of the EM algorithm. Each step is described in the following.

E-step: for each $i \in \mathcal{I}$, update $\mu_i = p(y_i = 1 | \mathcal{Y}; \Theta)$ as

$$\mu_i \leftarrow \frac{pa_i}{pa_i + (1-p)b_i},$$
$$\text{where } a_i = \prod_{j: y_{i,j} \neq \perp} \alpha_j^{y_{i,j}} (1-\alpha_j)^{1-y_{i,j}} \text{ and } b_i = \prod_{j: y_{i,j} \neq \perp} \beta_j^{1-y_{i,j}} (1-\beta_j)^{y_{i,j}}.$$

M-step: update p as

$$p \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mu_i,$$

and for each $j \in \mathcal{J}$, update the ability parameters α_j and β_j as

$$\alpha_j \leftarrow \frac{\sum_{i: y_{i,j} \neq \perp} \mu_i y_{i,j}}{\sum_{i: y_{i,j} \neq \perp} \mu_i} \text{ and } \beta_j \leftarrow \frac{\sum_{i: y_{i,j} \neq \perp} (1-\mu_i)(1-y_{i,j})}{\sum_{i: y_{i,j} \neq \perp} (1-\mu_i)}.$$

These update rules are derived so that the parameters maximize the Q-function shown below:

$$Q(\Theta) = \sum_{i \in \mathcal{I}} \left[\mu_i \log pa_i + (1-\mu_i) \log (1-p) b_i \right]. \tag{5.1}$$

We consider that the algorithm converges if

$$|Q(\Theta^{(t+1)}) - Q(\Theta^{(t)})|/|Q(\Theta^{(t+1)})| < 10^{-8}$$

holds, where $\Theta^{(t)}$ is the set of the parameters obtained in the t-th iteration. The time complexity of both the E-step and the M-step is $O(IJ)$.

5.3 Worker-Private Quality Control Problem

We present our problem setting called a worker-private quality control problem in Section 5.3.1 by introducing privacy requirements into the quality control problem. Then, we present our solution to the worker-private quality control problem called a worker-private latent class (WPLC) protocol in Section 5.3.2, which implements the LC method using cryptographic operations.

5.3.1 Problem Setting

We specify the privacy requirement that is necessary to preserve workers' privacy. Our observation is that the issues of worker privacy do not happen if each worker stores his/her labels secretly and none of the parties can obtain labels of other parties. Based on this observation, we define the privacy requirement called worker privacy and the worker-private quality control problem as Definition 5.1 and Problem 5.2, respectively. In Problem 5.2, values associated with a worker include the crowd labels given by the worker and the ability parameters if one uses the LC method.

Definition 5.1 (Worker-private)**.** *Assume that worker j secretly stores value v_j. If the requester and the workers except for worker j cannot determine v_j uniquely, then value v_j is worker-private.*

Problem 5.2 (Worker-private quality control problem)**.** *Assume that each worker $j \in \mathcal{J}$ has his/her labels $\{y_{i,j} \mid i \in \mathcal{I}\}$ secretly. The worker-private quality control problem is that the requester infers the true labels $\mathcal{Y}^\star$ from crowd labels $\mathcal{Y}$ while keeping all the values associated with the workers worker-private.*

The definition of worker-privacy is closely related to the privacy definition employed in query auditing (Nabar et al., 2008). Informally, query auditing attempts to prevent disclosures of private data from the public statistics of the data such as the mean and a maximum. The definition of disclosure called full disclosure describes the situation where a private value is determined uniquely from the statistics. This definition is related to our privacy definition, because a label is worker-private if and only if the label is not fully disclosed.

5.3.2 Worker-Private Latent Class Protocol

We present a worker-private latent class protocol (WPLC protocol) as a solution to the worker-private quality control problem (Problem 5.2). It executes the EM algorithm for the LC model in a privacy-preserving way; thus workers do not have to disclose their private values to the others. We first introduce the WPLC protocol, the main protocol in this research, in Section 5.3.2, and then we introduce a secure sum protocol, which is used as a sub-protocol in the main protocol, in Section 5.3.2.

Main Protocol

The WPLC protocol (Protocol 5.1) executes the EM algorithm of the LC method in a privacy-preserving way and finally outputs the posterior probability distribution of the unobservable true labels. The parties of it are the workers, the platform, and the requester.

The update rule of the WPLC protocol is basically the same as the original inference algorithm of the LC method except for the use of the secure sum protocol. The E-step (lines 8–9) is interpreted as the sum of the crowd labels weighted by the ability parameters of the workers. Since each summand is a secret value of each worker, the parties rely on the secure sum protocol to compute the summation without disclosing the secret values. The M-step (lines 15–16) is to estimate the ability parameters of each worker using the current estimates of the true labels. Since the true labels are public information in the protocol, each worker independently update his/her ability parameters.

The main idea for privacy preservation is twofold: introduction of the secure sum protocol in the E-step and participation of workers in the both steps. Both ideas are necessary to preserve worker privacy. Participation of workers is necessary because the requester cannot access crowd labels without it. The secure sum protocol is also necessary because the platform can get crowd labels $\mathcal{Y}$ in the second line of Protocol 5.1 without the secure sum protocol. Moreover, the platform can also acquire the parameters $\{\theta_j \mid j \in \mathcal{J}\}$ in the eighth line by using the crowd labels obtained in the second line. Therefore, we conclude that both ideas are essential to preserve worker-privacy. In Section 5.4, we prove that the crowd labels and the ability parameters are kept worker private after execution of the WPLC protocol.

The WPLC protocol can be extended to deal with a multi-class label and a real-valued label. We describe the update rules in Appendix B.1.

Protocol 5.1 Worker-Private Latent Class Protocol.
Parties: platform, requester, and workers $\mathcal{J}$.
Public input: public key pk
Public output: $\{\mu_i^{(t)} \mid i \in \mathcal{I}\}$ and $p^{(t)}$ for all $t \in \mathbb{N}$.
Private inputs:

- **Platform:** list of workers $\mathcal{J}$.
- **Requester:** secret key sk.
- **Worker** j**:** his/her crowd labels $\{y_{i,j} \mid i \in \mathcal{I}\}$.

Private outputs:

- **Requester:** $\{(a_i^{(t)}, b_i^{(t)})\}_{i \in \mathcal{I}}$ for all $t \in \mathbb{N}$.
- **Worker** j**:** $(\alpha_j^{(t)}, \beta_j^{(t)})$ for all $t \in \mathbb{N}$.

Requester updates $t \leftarrow 0$ and broadcasts it.
For each $i \in \mathcal{I}$, parties calculate the followings using Protocol 5.2:

$$\mu_i^{(d)} = \sum_{j \in \mathcal{J}} \mathbb{I}[y_{i,j} \neq \perp] \text{ and } \mu_i^{(n)} = \sum_{j \in \mathcal{J}} \mathbb{I}[y_{i,j} \neq \perp] y_{i,j}$$

Requester calculates $\mu_i^{(t)} = \frac{\mu_i^{(n)}}{\mu_i^{(d)}}$ and broadcasts $\{\mu_i^{(t)} \mid i \in \mathcal{I}\}$.
repeat
    Requester updates $p^{(t)} \leftarrow \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mu_i^{(t)}$.
    Each worker $j \in \mathcal{J}$ updates

$$\alpha_j^{(t)} \leftarrow \frac{\sum_{i:y_{i,j} \neq \perp} \mu_i^{(t)} y_{i,j}}{\sum_{i:y_{i,j} \neq \perp} \mu_i^{(t)}}, \ \beta_j^{(t)} \leftarrow \frac{\sum_{i:y_{i,j} \neq \perp} (1 - \mu_i^{(t)})(1 - y_{i,j})}{\sum_{i:y_{i,j} \neq \perp} (1 - \mu_i^{(t)})}.$$

    Requester broadcasts $t \leftarrow t + 1$.
    For each $i \in \mathcal{I}$, parties calculate the followings using Protocol 5.2:

$$\log a_i^{(t)} \leftarrow \sum_{j \in \mathcal{J}} \mathbb{I}[y_{i,j} \neq \perp] \left(y_{i,j} \log \alpha_j^{(t-1)} + (1 - y_{i,j}) \log(1 - \alpha_j^{(t-1)}) \right),$$

$$\log b_i^{(t)} \leftarrow \sum_{j \in \mathcal{J}} \mathbb{I}[y_{i,j} \neq \perp] \left((1 - y_{i,j}) \log \beta_j^{(t-1)} + y_{i,j} \log(1 - \beta_j^{(t-1)}) \right),$$

    Requester broadcasts $\mu_i^{(t)} \leftarrow \frac{p^{(t-1)} a_i^{(t)}}{p^{(t-1)} a_i^{(t)} + (1 - p^{(t-1)}) b_i^{(t)}}$ for each $i \in \mathcal{I}$.
    Requester calculates the Q-function $Q^{(t)}$ using Equation (5.1).
until $|Q^{(t)} - Q^{(t-1)}| / |Q^{(t)}| < \epsilon$

Protocol 5.2 Secure Sum Protocol.

Parties: platform, requester, and workers $\mathcal{J}$.
Public input: Large integer $L \in \mathbb{Z}_N$, rounding function $r : \mathbb{R} \to \mathbb{Z}_+$, and public key pk.
Private inputs:

- **Platform:** list of workers $\mathcal{J}$.
- **Requester:** secret key sk.
- **Worker j:** value $v_j \in \mathbb{R}$.

Private outputs:

- **Requester:** approximated sum of the values $\sum_{j \in \mathcal{J}} v_j$.

1: Each worker $j \in \mathcal{J}$ encrypts $c_j \leftarrow \mathsf{Enc}(r(v_j L))$.
2: Each worker $j \in \mathcal{J}$ sends c_j to the platform.
3: The platform calculates

$$\chi \leftarrow \prod_{j \in \mathcal{J}} c_j \quad \left(= \mathsf{Enc}\left(\sum_{j \in \mathcal{J}} v_j L \right) \right).$$

4: The platform sends χ to the requester.
5: The requester decrypts χ to obtain approximation of $\sum_{j \in \mathcal{J}} v_j$.

Secure Sum Protocol

We present a secure sum protocol (Protocol 5.2) that allows the requester to obtain the sum of values workers have secretly. The parties of it consist of the workers, the platform, and the requester. At the beginning, each worker j secretly hold his/her value v_j. Each worker j rounds it into an integer as $r(v_j L)$, where $r : \mathbb{R} \to \mathbb{Z}_+$ is a rounding function, and L is a large integer called an approximation parameter. Then, s/he encrypts and sends it to the platform. The platform serves as an intermediate node between the requester and the workers and performs the summation of the encrypted values received from the workers. The requester has a pair of public and secret keys $(\mathsf{pk}, \mathsf{sk})$ and performs decryption of the encrypted sum received from the platform. Finally, only the requester obtains the summation of all the values of the workers.

5.3.3 Discussion

Drawbacks

The WPLC protocol has two drawbacks compared to the LC method: numerical errors and computation time. Numerical errors are inevitable because of the rounding function in the secure sum protocol. In Section 5.5.1, we evaluate how the rounding has an effect on the performance of the WPLC protocol and how we can control it by adjusting the approximation parameter. The WPLC protocol requires more computation time than the LC method because of communication between parties, key generation, and encryption and decryption of messages. We examine the cryptographic overhead in Section 5.5.2 to show that the computation finishes in practical computation time. Communication cost is not examined because it heavily depends on the communication environment.

Exclusion of Spam Workers

One may have a concern that preserving worker privacy makes it impossible to exclude spam workers. We believe that a software to execute the protocol provided by a crowdsourcing platform can help us exclude spam workers with minimum worker privacy invasion. It is possible that the worker-side software by itself reports to the crowdsourcing platform that the worker is a spam worker if the worker's ability is lower than a certain threshold. This allows the crowdsourcing platform to ban the worker's account, which helps to increase the reliability. This solution invades little worker privacy; the requester does not obtain any information about the labels and the ability parameters of the workers, and the crowdsourcing platform only learns that the ability of the worker is lower than a certain threshold.

5.4 Security Proofs of the Protocols

We prove the security of the WPLC protocol in Theorem 5.1 and the security of the secure sum protocol in Lemma 5.1. The theorem states that crowd labels are kept worker-private after the execution of the WPLC protocol.

5.4.1 Statement of the Theorem

We borrow the privacy assumptions made in Section 2.4. Then, Theorem 5.1 is described as follows.

Theorem 5.1 (Security of the WPLC protocol)**.** *For any $i \in \mathcal{I}$, we assume that the set of labels given to instance i consists of at least two null labels and at least two valid labels. Under the privacy assumptions in Section 2.4, after the execution of the WPLC protocol, any party cannot determine any private label $y_{i,j}$ uniquely, and therefore, the crowd labels are worker-private.*

The first assumption in Theorem 5.1 is not strong. In most cases, a requester determines the number of redundant labels to one instance beforehand (typically, three labels per instance), and a number of workers are engaged in the task. As a result, the set of labels given to one instance consists of $|\mathcal{J}| - 3$ null labels and three valid labels with large $|\mathcal{J}|$.

5.4.2 Proof

We prove Theorem 5.1 using the security of the secure sum protocol, stated in Lemma 5.1. We first give the statement of the lemma along with its proof, and then, we prove the theorem.

Lemma 5.1 (Security of the secure sum protocol)**.** *Let $|\mathcal{J}| \geq 3$. Under the privacy assumptions in Section 2.4, after the execution of the secure sum protocol, the requester learns nothing but the sum, and the platform and the workers learn nothing.*

Proof of Lemma 5.1. We prove the lemma by examining the information each entity obtains.

(i) Worker j ($\in \mathcal{J}$)
Since worker j does not receive any additional information but sends a ciphertext during the protocol, worker j learns nothing after the execution of the protocol.

(ii) Platform
The additional information the platform receives during the protocol is ciphertexts $\{c_j \mid j \in \mathcal{J}\}$ from all the workers. Since these ciphertexts are encrypted using the Paillier cryptosystem, which is IND-CPA, the platform learns nothing from them.

(iii) Requester
The additional information the requester receives during the protocol is the encrypted sum χ and its decryption. Therefore, the requester learns nothing but the encrypted sum.

These discussions complete the proof of Lemma 5.1.

□

The lemma helps us to specify the additional information each entity obtains during the main protocol. The idea of the proof is that even if an entity has an estimate of private information, the entity can come up with another estimate, which implies that the entity cannot uniquely determine the private information. In other words, we prove that there does not exist a function from the information each entity has to a private label, which we call a privacy invading function in this proof.

Proof of Theorem 5.1. For each entity, we first examine the information s/he has after the execution of the protocol, which corresponds to his/her private input and output and the public output. Then, we show that there does not exist a privacy invading function that uniquely determines private information.

Let us first specify the public output of the protocol. Let us denote the number of iterations to converge by T. The public output is

$$\left\{\mu_i^{(t)} \mid i \in \mathcal{I}, t \in \{0, \dots, T\}\right\}. \tag{5.2}$$

Since $\{p_i^{(t)} \mid i \in \mathcal{I}, t \in \{0, \dots, T\}\}$ can be calculated given the public output, we ignore them in the following.

(i) Requester

The private input of the requester is the secret key sk, and the private output is

$$\mu_i^{(d)} = \sum_{j \in \mathcal{J}} \mathbb{I}[y_{i,j} \neq \perp],\ \mu_i^{(n)} = \sum_{j \in \mathcal{J}} \mathbb{I}[y_{i,j} \neq \perp] y_{i,j}$$

for each $i \in \mathcal{I}$ and

$$\log a_i^{(t)} = \sum_{j \in \mathcal{J}} \mathbb{I}[y_{i,j} \neq \perp] \left(y_{i,j} \log \alpha_j^{(t-1)} + (1 - y_{i,j}) \log(1 - \alpha_j^{(t-1)})\right),$$
$$\log b_i^{(t)} = \sum_{j \in \mathcal{J}} \mathbb{I}[y_{i,j} \neq \perp] \left((1 - y_{i,j}) \log \beta_j^{(t-1)} + y_{i,j} \log(1 - \beta_j^{(t-1)})\right)$$

for all $t \in \{1, \dots, T\}$ and $i \in \mathcal{I}$. We ignore the public output because it can be computed using the private output of the requester.

Assume that the requester had a privacy invading function from these values to the private label $y_{i',j'}$ for arbitrary $i' \in \mathcal{I}$ and $j' \in \mathcal{J}$. From the assumption of Theorem 5.1, there exists $j'' \in \mathcal{J} \backslash \{j'\}$ such that $y_{i',j'} \neq y_{i',j''}$ holds. Let us construct another set of crowd labels $\tilde{\mathcal{Y}} =$

$\{\tilde{y}_{i,j} \mid i \in \mathcal{I}, j \in \mathcal{J}\}$ by permuting j' and j'' in the original set of crowd labels as follows:

$$\tilde{y}_{i,j} = \begin{cases} y_{i,j} & (i \neq i' \text{ or } j \neq j', j''), \\ y_{i,j'} & (i = i' \text{ and } j = j''), \\ y_{i,j''} & (i = i' \text{ and } j = j'). \end{cases}$$

Notice that the values of the information the requester has are the same between $\mathcal{Y}$ and $\tilde{\mathcal{Y}}$. Therefore, the output of the privacy invading function on the permuted set should be the same as that on the original set, *i.e.*, $y_{i',j'}$. However, from the definition of the privacy invading function, it should output the private label to instance i' given by worker j', *i.e.*, $\tilde{y}_{i',j'}$ $(\neq y_{i\prime,j'})$, which is a contradiction. Therefore, the requester cannot determine any private label uniquely.

(ii) Worker $j^\star$ $(\in \mathcal{J})$

The information worker $j^\star$ has is the public output (Equation (5.2)) and the private input of worker $j^\star$, *i.e.*, the sets of his/her crowd labels and ability parameters:

$$\{y_{i,j^\star} \mid i \in \mathcal{I}\},$$
$$\left\{\alpha_{j^\star}^{(t)}, \beta_{j^\star}^{(t)} \mid t \in \{0, \ldots, T\}\right\}.$$

Assume that worker $j^\star$ had a privacy invading function from these values to the private label $y_{i',j'}$ for arbitrary $i' \in \mathcal{I}$ and $j' \in \mathcal{J}\backslash\{j^\star\}$. From the assumption of Theorem 5.1, there exists $j'' \in \mathcal{J}\backslash\{j', j^\star\}$ such that $y_{i',j'} \neq y_{i',j''}$ holds. Then, in the same way as the case with the requester, we are able to leads to a contradiction, and therefore, any worker cannot determine any private label uniquely.

(iii) Platform

The private input of the platform is a set of workers participating in the protocol $\mathcal{J}$. In this case, we can prove that a privacy invading function does not exist in the same way as the requester case by constructing another set of crowd labels. Since the permuted set does not change the information the platform has, there comes the contradiction. Therefore, the platform cannot determine any private label uniquely.

These discussions prove that any private label cannot be determined. □

5.5 Experiments

The performance of the WPLC protocol can be different from that of the LC method in computation time and accuracy as discussed in Section 5.3.3. We have conducted two experiments, each of which examines the accuracy degradation (Section 5.5.1) and the computation overhead (Section 5.5.2) compared to the LC method, to show that the WPLC protocol is practical in a real environment.

5.5.1 Experiments on Approximation Accuracy

The WPLC protocol executes the LC method approximately, because it involves a rounding operation in the secure sum protocol. We investigate the approximation accuracy using both synthetic and real datasets to show that it does not ruin the aggregated results.

Datasets

Synthetic Dataset. We use the LC model to generate synthetic datasets in the following procedure. First, for all $i \in \mathcal{I}$, we generate the true labels y_i from the Bernoulli distribution with its parameter $p = 0.5$. Then, for all $i \in \mathcal{I}$ and $j \in \mathcal{J}$, we generate crowd labels given by worker j from two Bernoulli distributions with their parameters α_j and β_j. The parameters of the synthetic dataset are the number of instances I, the number of workers J, and the ability parameters $\{\alpha_j \mid j \in \mathcal{J}\}$ and $\{\beta_j \mid j \in \mathcal{J}\}$.

Real Dataset. We employ the Duchenne Smiles Dataset (Whitehill et al., 2009) as a real dataset. The task was to give each face image a label whether the smile on the image was a duchenne one (enjoyment smile) or a non-duchenne one. The images (or instances in our paper) have the ground truth labels that were given by experts. The number of images I is 159, and 58 out of the 159 images contain Duchenne smiles according to the ground truths. In total, 20 workers gave labels, and $3,513$ crowd labels were collected, where $1,804$ out of the $3,513$ crowd labels were duchenne labels.

Experimental Settings

To understand the approximation accuracy, we examined (i) the accuracy of the aggregated labels compared to the ground truth labels and (ii) the relative errors of model parameters obtained by using the WPLC protocol and the LC method.

Table 5.1: Performance comparison on the real dataset. The performance of the WPLC protocol was not affected by L.

MV method	LC method	**WPLC protocol**
0.752	**0.761**	**0.761**

(i) Accuracy of the Aggregated Labels. We examine the accuracy of the aggregated labels against the ground truth labels, varying the parameters of datasets and the approximation parameter L, which controls the approximation accuracy of the WPLC protocol. The effect of the approximation parameter L is examined on both synthetic and real datasets, and the effect of the parameters of datasets is examined on synthetic datasets, because the parameters cannot be altered in the real dataset. The detailed parameters of the synthetic datasets are described in the captions of Figure 5.2, where we use the notation $\alpha_{k:l} = \alpha^\star$ to denote $\alpha_k = \alpha_{k+1} = \cdots = \alpha_{l-1} = \alpha^\star$.

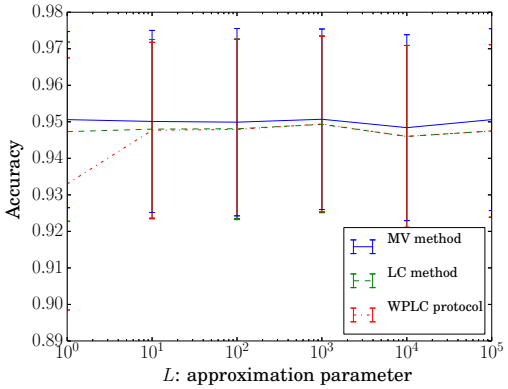
The performance is measured by the accuracy, *i.e.*, the percentage of the aggregated labels that agree with the ground truth labels. For both the LC method and the WPLC protocol, we estimate the aggregated label of instance i as $y_i = 1$ if $\mu_i > 0.5$, and $y_i = 0$ otherwise. For comparison, we also test the majority voting (MV) method, which is also easily made secure by our secure sum protocol. For the experiments using synthetic datasets, we repeat experiments 100 times to obtain the mean and the standard deviation of the accuracy.

(ii) Relative Errors of Parameters. We compare the relative errors of the model parameters of the WPLC protocol and the LC method on the real dataset, varying the approximation parameter L as $10^0, 10^1, \ldots, 10^{14}$. Let us denote a parameter of the LC method by vector $\mathbf{x}$ and the corresponding parameter of the WPLC protocol by vector $\tilde{\mathbf{x}}$. Then, we define the relative error of the parameter as $\|\log \mathbf{x} - \log \tilde{\mathbf{x}}\| / \|\log \mathbf{x}\|$. We examine the relative errors of $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_J]$, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_J]$, $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_I]$, and p.

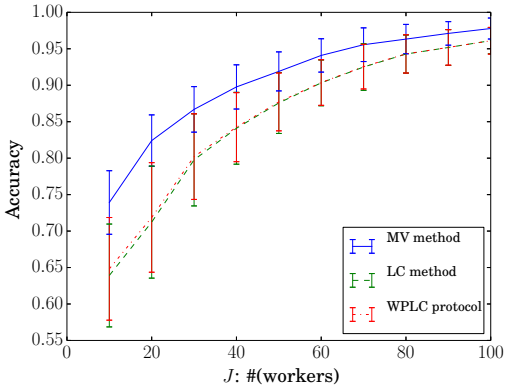
Results

(i) Accuracy of the Aggregated Labels. The results on the synthetic datasets are shown in Figures 5.2a–5.2f, and those on the real dataset are shown in Table 5.1. We have three findings from the results.

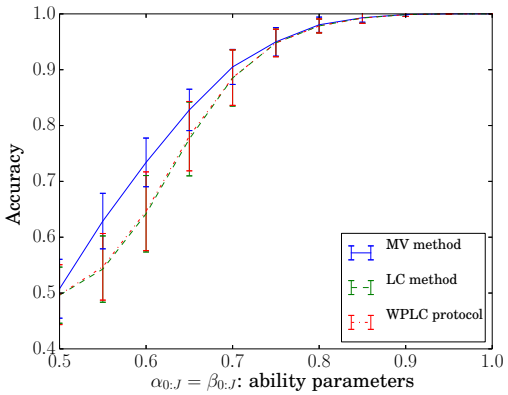
First, Figure 5.2a and Table 5.1 show that the performance of the WPLC protocol was almost the same as that of the LC method if the approximation parameter L was larger than $L = 10$. In specific, Table 5.1 shows that



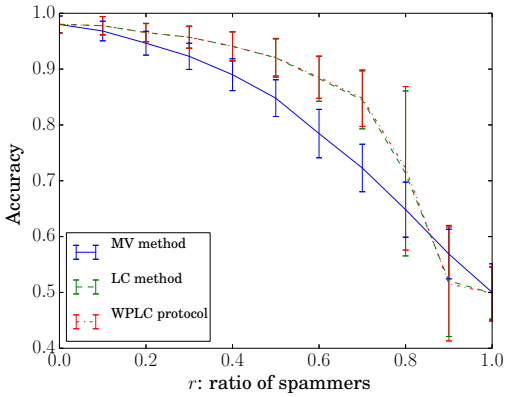
(a) Fixing $I = 100, J = 10, \alpha = \beta = 0.75$, and varying $L = 10^0, 10^1, \ldots, 10^5$.



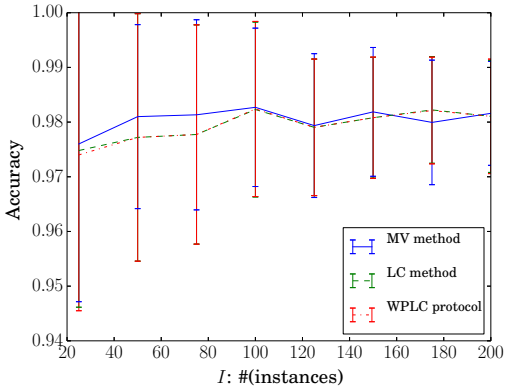
(b) Fixing $I = 100, \alpha = \beta = 0.6, L = 100$ and varying $J = 10, 20, \ldots, 100$.



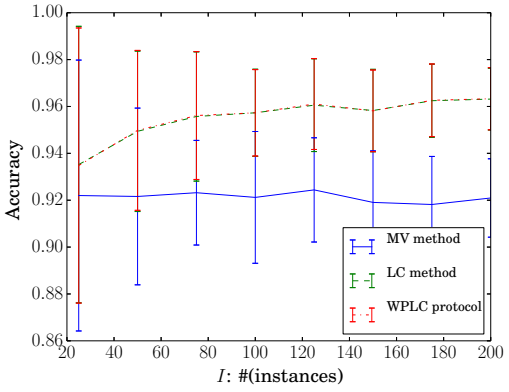
(c) Fixing $I = 100, J = 10, L = 100$ and varying $\alpha_{0:J} = \beta_{0:J} = 0.5, 0.55, \ldots, 1.0$.



(d) Fixing $J = 10, L = 100, \alpha_{0:rJ} = \beta_{0:rJ} = 0.5, \alpha_{rJ:J} = \beta_{rJ:J} = 0.8$ and varying $r = 0, 0.1, \ldots, 1.0$.



(e) Fixing $J = 10, L = 100, \alpha_{0:J} = \beta_{0:J} = 0.8$ and varying $I = 25, 50, \ldots, 200$.



(f) Fixing $J = 10, L = 100, \alpha_{0:3} = \beta_{0:3} = 0.5, \alpha_{3:10} = \beta_{3:10} = 0.8$ and varying $I = 25, 50, \ldots, 200$.

Figure 5.2: Performance comparison on the synthetic dataset. The parameter setting for each figure is given in the caption of it. The error bars show standard deviations.

rounding in secure sum had little effect on the performance in practice. From these results, we conclude that the accuracy of the aggregated labels are not affected by the approximation parameter if appropriately chosen.

Second, Figures 5.2b–5.2f show that the performance of the LC method and that of the WPLC protocol are not influenced by the parameters of the dataset if $L = 100$. Therefore, together with the first finding, we conclude that the performance of the WPLC protocol with L (> 100) is the same as that of the LC method.

Finally, Figures 5.2b–5.2f and Table 5.1 highlight the performance characteristics of the MV method and the LC-type methods, *i.e.*, the LC method and the WPLC protocol. The LC-type methods perform better than the MV method in Table 5.1 and Figures 5.2d and 5.2f, when the ability parameters of the workers are heterogeneous. The performance is the same in Figure 5.2e, when the workers are highly skilled. The MV method is better than the LC-type methods in Figures 5.2b and 5.2c, when the ability parameters of the workers are relatively low and homogeneous. From these results, we conclude that the LC-type methods are preferable to the MV method when the workers have diverse expertise. Since workers in a real crowdsourcing platform actually have diverse expertise, the LC-type methods perform better than the MV method as shown in Table 5.1, which supports the extension of the LC method, not the MV method.

From these studies, we conclude that the rounding operation in the protocol has little effect on the performance if we set the approximation parameter L reasonably large and that the extension of the LC method is preferred to the MV method considering the performance on real crowdsourcing.

(ii) Relative Errors of Parameters. The results on the real dataset are shown in Figure 5.3. The relative errors decrease as the approximation parameter L increases, which implies that we can decrease the errors as small as necessary. Therefore, we conclude that the relative errors of the model parameters are not problematic if we set the approximation parameter L sufficiently large.

One may notice that the relative error of $\log \mu$ do not decrease drastically if the approximation parameter L is smaller than 10^7. This is partly due to the difference in the number of iterations. The number of iterations for the algorithms to converge is shown in Figure 5.4. We can observe that the number of iterations of the WPLC protocol is different from that of the LC method when the approximation parameter L is smaller than 10^7, which is expected to have influence on the relative error. When the approximation parameter L is too small, small changes in the ability parameters are not

reflected to the posterior probabilities $\{\mu_i \mid i \in \mathcal{I}\}$ in the E-step due to rounding, which will cause the small number of iterations of the WPLC protocol.

5.5.2 Experiment on Computational Efficiency

We examine the computational efficiency of the WPLC protocol. As stated in Section 5.3.3, the WPLC protocol requires more computation time than the LC method because the WPLC protocol requires communication between parties, key generation, and encryption and decryption of messages. We experimentally evaluate cryptographic computation time to investigate whether the WPLC protocol is practical in real crowdsourcing.

Experimental Setting

We evaluate the computation time of key generation and the overhead caused by cryptographic operations in the WPLC protocol based on the computation time of basic operations shown in Table 2.1. The computation overhead of one iteration of the WPLC protocol depends on the number of instances I and the number of workers J. One iteration of the WPLC protocol consists of $2I$ secure sum operations, and one iteration of the secure sum protocol consists of J encryptions, $J-1$ secure addition operations, and one decryption. Therefore, the computation overhead of one iteration of the WPLC protocol is estimated as

$$2I(JT_{\text{enc}} + T_{\text{dec}} + (J-1)T_{\text{add}}),$$

where T_{enc}, T_{add}, and T_{dec} correspond to the computation times of encryption, secure addition, and decryption.

Results

The computation time of key generation was 45.2 ms, and the computation overhead of one update of the WPLC protocol was estimated using the results in Table 2.1 as

$$2I(10.14J + 9.63 + 0.01(J-1)),$$

For example, assuming that the numbers of instances and workers are $I = J = 100$, the overhead will be about 205 seconds. Considering that the parties only have to put their computers on and do not have to be involved in the protocol, this computation in exchange for worker privacy will be tolerable.

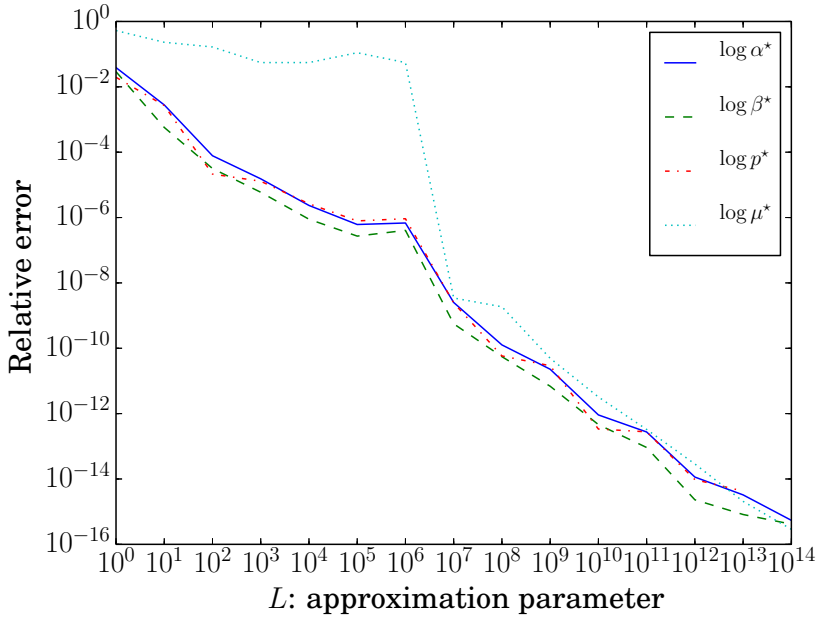


Figure 5.3: Relative errors of the model parameters of the LC method and those of the WPLC protocol (L versus the relative errors).

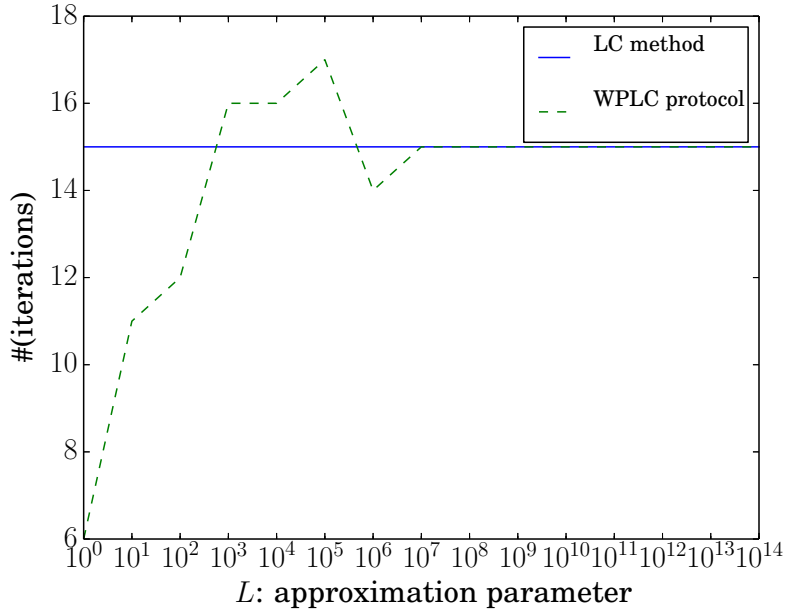


Figure 5.4: Number of iterations required by the LC method and the WPLC protocol to converge (L versus the number of iterations).

Furthermore, considering that each worker performs encryption in a parallel manner in the secure sum protocol, the actual computation time will be significantly lower than the above estimate. In summary, we conclude that the computation overhead incurred by the Paillier cryptosystem is practically acceptable in consideration of the privacy guarantee for workers.

5.6 Summary and Future Work

This chapter has discussed the issue of worker privacy in crowdsourcing and has introduced a basic idea to preserve worker privacy. Observing that the label aggregation by a quality control method can detach personal identifiers from the results, our solution called the worker-private latent class protocol is to secretly aggregate results based on the update rules of the latent class method and hand the aggregated results to the requester. We theoretically prove the security of our protocol, guaranteeing that the re-identification of a worker is impossible. In addition, we have empirically evaluated the performance of our protocol compared to the original latent class method. The experimental results show that our protocol achieves the same accuracy as the latent class method if we employ a sufficiently large approximation parameter and that the computational overhead caused by the cryptographic operations is tolerable in exchange for the privacy guarantee.

An interesting research direction is to learn a classifier directly from crowd labels with preserving worker privacy. Raykar et al. (2010) experimentally show that it is better to learn a classifier directly from crowd labels than to infer the aggregated labels first and to learn a classifier from the aggregated labels. Therefore, if the requester is willing to acquire a classifier using the crowd labels, such a protocol will be desirable to the requester.

Chapter 6

Related Work

This chapter discusses work related to this dissertation. We first discuss the work related to privacy preservation in crowdsourcing and clarify the contributions of this dissertation in Section 6.1. We then widen the scope of related work to review the existing work of two distinct research areas, crowdsourcing and privacy preservation, in Sections 6.2 and 6.3, respectively, and we state the relationships between this dissertation and these two different research areas. The latter two sections will help readers to grasp the contributions of this dissertation to these two popular research areas.

6.1 Privacy Preservation in Crowdsourcing

We review the privacy preservation research for crowdsourcing in this section. Because our dissertation consists of three main topics, we provide the related work and clarify our contributions for each topic.

6.1.1 Privacy Preservation in Task Assignment

This section reviews the work related to privacy preservation in task assignment (Chapter 3). Table 6.1 briefly compares our protocol with the existing protocols.

Privacy issues in task assignment have been mostly discussed in the context of spatial crowdsourcing, which deals with tasks that cannot be completed without traveling to specific locations. Examples of such spatial tasks include tasks to collect pictures of specific locations, tasks to collect as many points of interest as possible, tasks to report traffic information, and errand tasks. Task assignment for spatial tasks requires workers and requesters to disclose their locations and the locations of points of interest, respectively,

Table 6.1: Comparison of PTA and the existing studies. Most of the existing studies focus on preserving the locations of workers, while our PTA preserves the features of both workers and tasks, including the locations.

	Target	**Approach**
Kazemi and Shahabi (2011)	Locations of workers	Perturbation
To et al. (2014)	Locations of workers	Perturbation
PTA (Chapter 3)	Features of workers and tasks	Cryptography

to a crowdsourcing platform, which can invade their privacy. Therefore, it is necessary to develop a privacy-preserving task assignment system for spatial crowdsourcing.

One of the pioneering studies in this literature was conducted by Kazemi and Shahabi (2011), who specifically focus on participatory sensing. Participatory sensing is a variant of spatial crowdsourcing that leverages mobile devices equipped with sensors for large-scale sensing. Their main concern is that a worker has to disclose his/her location information to the assignment server when s/he queries a task. They hence proposed PiRi, which addresses the privacy issue by relying on a P2P technique to compute spatially k-anonymized queries posed by workers to the assignment server. The subsequent work (Kazemi and Shahabi, 2012b) deals with the trust issue in addition to the privacy issue. Because workers in crowdsourcing are untrustworthy, the data provided by workers are not always correct. They therefore extended PiRi to cope with the trust issue by assigning a single task to multiple workers to obtain redundant data.

To et al. (2014) employed differential privacy (Dwork, 2006) as a privacy guarantee instead of the k-anonymity-based privacy guarantee. Their solution is to utilize a cellular service provider that already has trust relationships with workers. In their framework, the cellular service provider sanitizes and releases the location data of workers according to the differential privacy criterion, and spatial crowdsourcing platforms assign tasks based on the sanitized location data. To et al. (2015) followed up this research by developing an interactive visualization and tuning toolbox for their framework. Their toolbox helps users to tune several parameters of their framework such as those for differential privacy and a selection of geocasting strategies.

In summary, the existing studies preserve workers' privacy by adding an appropriate amount of noise to these workers' locations, *e.g.*, according to the differential privacy criterion (Dwork, 2006) or a spatial k-anonymity cri-

Table 6.2: Comparison of our study (the IC protocol and UPTA) and the existing studies. Our study, resigning the perfect privacy preservation, is able to preserve instance privacy on more general instances than medical charts, and thus, UPTA is necessary that quantifies both utility and privacy. UPTA guarantees the privacy not by the accuracy but by an information-theoretic measure that is based on the uninformative principle.

	Target	**Privacy guarantee**
Little and Sun (2011)	Medical chart	Perfect
Lasecki et al. (2015a)	Video	Accuracy
Lasecki et al. (2015b)	Array	Accuracy
IC protocol (Chapter 4)	Array	UPTA

terion. There are three main differences between their studies and ours. First, they do not take the privacy of requesters into account, whereas our problem setting takes the privacy of the requesters into account in addition to that of workers. As some requesters, *e.g.*, those who using crowdsourcing for business purposes, are willing to conceal the details of their tasks, it is strongly desirable to preserve their privacy. Second, they resort to perturbation approaches to preserving privacy at the cost of accuracy, whereas our method follows a cryptographic approach that preserves privacy without sacrificing accuracy. Third, their methods specialize in spatial tasks, whereas our method considers features of a more general nature pertaining to tasks and workers, including skills, locations, wages, and working hours. These differences clearly emphasize the contributions of our work in the research area of privacy-preserving task assignment in crowdsourcing.

6.1.2 Instance Privacy

This section reviews the work related to instance privacy (Chapter 4). Table 6.2 briefly compares our study with other existing studies.

Instance privacy concerns the risk of information extraction from an instance that is handed to workers along with a job instruction. For example, in an audio transcription task, an audio recording corresponds to an instance, and it may contain sensitive information if it is a recording of a business meeting. Because crowdsourcing involves untrustworthy workers by its nature, how to preserve instance privacy in crowdsourcing is an essential research challenge.

Research to preserve instance privacy in crowdsourcing has been con-

ducted in the human-computer-interaction and human computation communities. The existing studies can be grouped into three types: (i) practical studies developing a privacy-preserving protocol for a specific problem setting, (ii) theoretical studies analyzing the trade-off that emerges when instance privacy is preserved, and (iii) studies pointing out and warning about the risk of instance privacy. Our study resides in the first and second groups.

(i) Practical Studies

A pioneering practical study was conducted by Little and Sun (2011), who developed a privacy-preserving human OCR system for medical charts. They make use of a blank template of the medical chart to decompose the chart into items, which prevents workers from associating a personal identifier (*e.g.*, a name) with attributes (*e.g.*, a disease). Although it is a popular idea to decompose documents into fragments for task requests (von Ahn et al., 2008; Chen et al., 2012), this work applied the idea to privacy preservation for the first time.

Another practical study was conducted by Lasecki et al. (2015a,b) after the publication of our work (Kajino et al., 2014b). Lasecki et al. (2015a) developed a privacy-preserving behavioral video coding method and analyzed the trade-off between utility and privacy. Behavioral video coding is a video annotation task that identifies specific events and their time of occurrence. In this task, a video clip corresponds to an instance, which cannot be put on a public website without any protection. Their idea is to blur the video so that a worker cannot identify the person in the video.

Lasecki et al. (2015b) developed a method to filter sensitive information contained in array-type instances given a natural language description of what should be filtered. Their idea of privacy preservation is similar to the IC protocol in that a small portion of an instance does not invade privacy much and thus can be handed to workers. In fact, their method improves the IC protocol by utilizing a pyramid workflow, which helps to overcome the problem of the fixed-size clipping window. The pyramid workflow first presents workers with small sub-instances and asks whether they contain sensitive information or not. It then masks the original instances using the responses from workers and repeats the same procedure with a larger clipping window.

(ii) Theoretical Studies

A few studies have pursed the theoretical aspects of instance privacy. In principle, privacy preservation comes at the cost of the quality of a result.

Therefore, it is a main subject of theoretical research to unravel the trade-off between privacy and quality.

Varshney (2012) approached this problem by building a mathematical model of a privacy preservation method using random perturbation. His privacy concern is that, even if we employ the random perturbation approach, colluding workers can recover the original instance from multiple independently and randomly perturbed instances. To this end, he theoretically investigated the trade-off between the quality of a result, privacy guarantee under collusion, and cost.

Lasecki et al. (2015a) also investigated the trade-off issue using experiments in a similar way to ours. They asked workers whether they were able to identify the person in a blurred video from a set of candidates. Because the identification accuracy drops quickly as the blur level increases while the quality does not drop steeply, they concluded that their framework can balance the trade-off. Our approach to the trade-off analysis is different from these two studies in that (1) we provide utility and privacy measures that can be computed by experiments, unlike the work by Varshney (2012), and (2) our privacy measure is more pessimistic than the measure proposed by Lasecki et al. (2015a). Our privacy measure penalizes the false identification case, in which the responses of workers are not correct but concentrate on one choice. For example, even if all the responses of workers concentrate on Alice in a privacy test for which the correct answer is Bob, we consider instance privacy to be invaded because Alice could suffer from the false identification.

(iii) Risk Analysis

Finally, several studies have pointed out the risk of instance privacy. Harris (2011) discussed several unethical uses of crowdsourcing, including information extraction by workers. Lasecki et al. (2014) raised the problem of information extraction by workers and experimentally showed that a non-negligible number of workers are willing to be engaged in unethical tasks. These studies present the risk that some workers could take part in unethical tasks such as information extraction tasks, which could lead to an invasion of instance privacy.

6.1.3 Privacy Preservation for Workers

This section reviews the work related to worker privacy (Chapter 5). Table 6.3 briefly compares our study and the existing studies.

The issue of worker privacy is that the sensitive information of a worker can be inferred from the results the worker produces. For example, the ability

Table 6.3: Comparison of WPLC protocol and the existing studies. Our protocol can preserve the utility of a requester by simulating a quality control method, while the existing studies sacrifice it.

	Target	**Approach**
Cornelius et al. (2008) Hu and Shahabi (2010) Puttaswamy et al. (2010)	Location	Anonymization
Huang et al. (2009)	Location	Perturbation
WPLC protocol (Chapter 5)	Any label	Quality control with cryptography

of a worker can be inferred by using the latent class method (Dawid and Skene, 1979), and the trajectory of a worker can be inferred from the results of a spatial crowdsourcing task. Although the importance of respecting worker privacy has been mentioned (Bernstein et al., 2011), the issue of worker privacy in a general task had been unsolved until our research was published.

In the context of participatory sensing, several studies deal with worker privacy. A worker in participatory sensing serves as a sensor node that reports their sensing results along with its location information. Even though workers are anonymized on the platform, we can identify them using the location information and resultant trajectory inferred from a collection of the location information. There are two major approaches to preserving the privacy of workers in participatory sensing. One approach is to keep the anonymity of workers by detaching the identifier of a worker from their results through anonymous routing (Cornelius et al., 2008; Hu and Shahabi, 2010; Puttaswamy et al., 2010). Another approach is to perturb the location information by adding noise or sanitization in order to satisfy some privacy criterion such as k-anonymity (Huang et al., 2009).

Our study differs from these studies mainly in the following two points. First, we develop a privacy-preserving aggregation method in the context of quality control, and therefore, the aggregation improves the quality of results the requester obtains while preserving worker privacy. On the contrary, these studies mainly focus on privacy preservation without taking into account the reliability of different workers. Second, our protocol can deal with binary, multi-class, and real-valued results, while their methods focus on real-valued location data only. These two differences highlight the contribution of our study not only to crowdsourcing research but also to participatory sensing.

6.2 Crowdsourcing

We review the existing studies on crowdsourcing in the machine learning and data mining communities. There are two major research topics: quality control and task assignment. Because these two topics are related to our research, we briefly summarize these two research topics to show the current research directions.

6.2.1 Quality Control

Since the launch of Amazon Mechanical Turk, a general-purposed crowdsourcing marketplace, a number of research groups have been pursuing the possibility of crowdsourcing for data annotation (Sheng et al., 2008; Snow et al., 2008; Deng et al., 2009). Using crowdsourcing entails both advantages and disadvantages. The advantage is that we can significantly reduce the monetary and time costs of constructing a training dataset for supervised learning. For example, in order to learn a classifier of texts, it is necessary to annotate a number of texts for training the classifier. If we use crowdsourcing for annotation, we can employ a number of workers to annotate the texts in an extremely parallel manner at low cost. One successful example is the ImageNet dataset (Deng et al., 2009), which is the state-of-the-art benchmark dataset for image classification. The disadvantage is that the quality of annotation is not guaranteed. As pointed out at the very beginning of crowdsourcing (Howe, 2006a), the quality of a result depends on the ability and motivation of a worker, which is unknown to the requester. Therefore, a number of studies have been conducted to address the quality control problem (Lease, 2011) to improve the quality of annotation in a statistical way.

One of the earliest studies was conducted on data mining (Sheng et al., 2008) and natural language processing (Snow et al., 2008). Sheng et al. (2008) proposed a repeated labeling strategy for the quality control problem. Their basic strategy was to assign the same labeling task to multiple workers to obtain redundant labels and aggregate them by majority voting to improve the quality. Snow et al. (2008) used crowdsourcing for human linguistic annotation and applied the latent class method (Dawid and Skene, 1979) to improve the annotation quality. This approach aggregates multiple labels considering the ability of each worker. Results produced by a low-ability worker have less influence on aggregation than those produced by a high-ability worker.

The subsequent research branches mainly into two directions: **(i)** the development of a more sophisticated aggregation method for the simple labeling task and **(ii)** the extension of the applicability of a quality control method to

various tasks. In this thesis, Chapters 4 and 5 are related to the research on quality control. The research in Chapter 4 does not investigate the quality control problem but applies the latent class method to improve the quality of results while preserving instance privacy. The research in Chapter 5 follows the first direction. The protocol presented there is a privacy-preserving variant of the latent class method. In the following, we review these two research directions respectively.

(i) Quality Control for Multiple-Choice Tasks

This research direction has two sub-fields. Some researchers use more sophisticated aggregation methods to improve the quality of aggregated results, while others analyze the performance of quality control methods.

Improvement of quality. The basic framework of more sophisticated methods is the same as that by Dawid and Skene (1979); the annotation process of a worker is modeled as a noisy label generation process from the latent true label, and multiple labels are aggregated by inferring the latent true label in an unsupervised manner. There are three approaches to improving the quality control method.

The first approach is to make the model more precise. Whitehill et al. (2009) propose GLAD, which takes the difficulty of instances into consideration in addition to the ability of workers. Welinder et al. (2010) introduced a latent feature space into their model to better capture the behavior of workers. In their model, each instance is mapped in the latent feature space, and each worker is modeled as a linear classifier in the same space who generates labels according to the classifier.

The second approach is to devise the objective function and optimization algorithm. Liu et al. (2012) focused on the inference algorithm part of the latent class method. They showed that we can improve the performance of the latent class method by applying existing inference methods for graphical models, including belief propagation and mean field approximation. In addition, they investigated the connections between their algorithms and the existing inference algorithms, including the expectation-maximization (EM) algorithm, majority voting algorithm, and message-passing-style algorithm for quality control (Karger et al., 2011a). Zhou et al. (2012) proposed the minimax entropy principle, which enables us to learn a more complex worker model than the model of the latent class method.

The third approach is to incorporate auxiliary information. Mo et al. (2013) incorporated the idea of transfer learning into the quality control problem. They proposed a hierarchical Bayesian model, which considers the

characteristics of a worker that govern his/her performance in different tasks. Ma et al. (2015) considered a question-and-answer task, where a worker is given a question and is asked to choose the answer from multiple choices. They assume that the reliability of a worker depends on the topic of the question. They estimate the topic-wise reliability from both questions and answers in order to improve the performance.

Analysis on performance. Another research line is to guarantee the performance of the quality control methods. Because most quality control methods utilize inference algorithms on probabilistic models with latent variables, it is relatively difficult to guarantee their performance. Karger et al. (2011a) simplify the model of the latent class method and develop an iterative inference algorithm for the model inspired by belief propagation. They provide asymptotic error bounds for the performance of the algorithm, which is the first theoretical guarantee on the performance of a quality control method, as far as we know. Li et al. (2013) proved finite-sample error bounds for the performance of weighted majority voting under the latent class model. Their error bounds are valid for any set of weights. Berend and Kontorovich (2014) investigated the consistency of the Nitzan-Paroush weighted majority voting rule, where a weight for each worker is defined as the log-odd of its accuracy. Note that they assume that the accuracy parameters of workers, or at least, their estimates are known. Zhang et al. (2014) proposed a provably optimal two-stage algorithm for the latent class method (Dawid and Skene, 1979). The first step is to use a spectral method to estimate the parameters roughly, and the second step is to refine the estimation using the EM algorithm. The performance guarantee of their algorithm can be derived by virtue of the global optimality of the spectral method.

(ii) Quality Control for Complex Tasks

Another research direction is to develop quality control methods to deal with various types of task results beyond multiple choices.

Gomes et al. (2011) proposed a quality control method for clustering in crowdsourcing. In their framework, workers are shown multiple instances chosen from a large dataset and are asked to perform clustering on them. The clustering results by multiple workers are then aggregated to obtain a clustering result over the entire dataset. Lin et al. (2012) handled an open-question task whose result is defined on an countably infinite set. They model the answering process using the Chinese restaurant process to represent the infinite variations of worker responses. Baba and Kashima (2013) proposed a quality control method for general tasks whose results do not have any fixed

format. They ask multiple workers to grade the redundantly acquired results and aggregate the scores to determine the best one. Chen et al. (2013), Yi et al. (2013), and Matsui et al. (2014) tackled the quality control problem on a ranking task, where a result corresponds to a ranking of objects. The basic approach is to employ a probabilistic model of ranking given the true ranking and modify the latent class method using the model. The goal of Chen et al. (2013) and Matsui et al. (2014) was to infer one true ranking over a set of objects, while the goal of Yi et al. (2013) was to infer a respective ranking over a set of objects personalized for each user. Kajimura et al. (2015) developed a quality control method for a task to collect points of interest in a map. The challenge is that it is necessary to infer whether two close points given by different workers indicate the same location or different point of interests. Their approach is to perform clustering on all the points to group points that indicate the same location, and then, to estimate the reliability of each cluster to eliminate unreliable ones.

There are also machine learning specific problem settings. A typical problem setting is the quality control of a classifier learned from a crowd-generated training dataset (Raykar et al., 2010; Dekel and Shamir, 2009; Yan et al., 2010; Zheng et al., 2010; Yan et al., 2011; Wauthier and Jordan, 2011; Kajino et al., 2012a,b; Liu et al., 2013; Lin et al., 2014). Directly learning a classifier is expected to achieve better classification accuracy than learning a classifier from a training dataset that is quality-controlled independently from the classifier. This was experimentally validated by Raykar et al. (2010). Another interesting machine learning specific problem setting is to learn a kernel from crowds (Tamuz et al., 2011). In their framework, workers are shown objects a, b, and c and are asked, *"Is object a more similar to b or to c?"* They proposed an adaptive method to learn a kernel over all the objects by sequentially selecting the triple (a, b, c) based on the current estimate of the kernel.

6.2.2 Task Assignment

Another major concern is how to design a task assignment algorithm in crowdsourcing. The standard task assignment method is the open call strategy, in which tasks are listed on web pages and workers choose and process tasks they like. This strategy can lead to a sub-optimal assignment in terms of accuracy and throughput. Accuracy can degrade because the open call policy allows spam workers to process a number of tasks, and throughput can degrade because a highly skilled worker may process tasks that do not require any special skill, while highly specialized tasks remain unprocessed. In addition, a survey on workers revealed that they were not satisfied with

Table 6.4: Existing approaches to homogeneous task assignment. Most deal with an online setting, in which an algorithm sequentially assigns a task to an appropriate worker.

	Assignment approach
Donmez et al. (2009)	Elimination of low-quality workers
Welinder and Perona (2010)	Elimination of low-quality workers
Yan et al. (2011)	Pick a worker and an instance (active learning formulation)
Tran-Thanh et al. (2014)	Pick a worker (bandit formulation)
Karger et al. (2011a,b,c)	Offline random assignment

the current open call system because they had to spend too much effort finding appropriate tasks (Kittur et al., 2013). Therefore, it is imperative to assign tasks to appropriate workers from a global perspective.

Homogeneous tasks. The earliest studies on task assignment focus on a labeling task, whose goal is to construct a dataset, the quality of whose labels is maximized within a fixed budget. The challenge of this problem setting is that the ability of workers is unknown to the requester. These studies often assume that the task is homogeneous, *i.e.*, the quality depends on a single ability. Table 6.4 summarizes the existing approaches to the homogeneous task assignment.

One approach to task assignment is to eliminate low-quality workers from a pool of workers. Donmez et al. (2009) proposed an active learning method called IEThresh, which aims to learn a classifier by adaptively selecting both an instance to be labeled and a set of workers whose labels are used to estimate the true label for the instance. Welinder and Perona (2010) proposed an online quality control method based on the latent class method, which aims to infer the true labels at low cost by adaptively selecting both instances and workers.

Another approach is to pick an appropriate worker based on a certain criterion. Yan et al. (2011) proposed an active learning method that chooses both an instance and a worker in an active learning manner. In detail, they repeatedly pick the most uncertain instance and the most confident worker for that instance based on the model. Tran-Thanh et al. (2014) formulated the task assignment problem as a bounded multi-armed bandit problem in which the workers correspond to the arms, charging different prices, and the number of calls per worker is limited. They provided a regret analysis of their algorithm.

A series of studies by Karger et al. (Karger et al., 2011a,b,c) aims to minimize the cost whilst achieving a certain reliability of labels. Because their model assumes that the ability of a worker is uniform across different tasks, a random task assignment strategy is proven to be asymptotically order-optimal, in terms of the budget necessary to achieve a given error rate. To the best of our knowledge, this is the first theoretical work.

Heterogeneous tasks. Subsequent studies generalize the existing problem settings to deal with general tasks that require workers to have special skills to complete.

One of the pioneering studies in this research line was conducted by Shahaf and Horvitz (2010). They considered an offline task assignment problem that allows workers to form teams to process tasks. They assume that processing a task requires a team of workers to have a certain set of skills.

Most of the studies have been developed based on the idea of the multi-armed bandit problem. Ho and Vaughan (2012) formulated the task assignment problem as a bandit-like problem in which a worker exhibits different abilities, which are unknown to the requester, on different tasks. They performed theoretical competitive analysis of their online algorithm against the optimal offline algorithm. Ho et al. (2013) considered the task assignment problem in the context of the quality control problem with the goal of high-quality label aggregation. This problem setting has two main difficulties. Contrary to the work by Karger et al. (2011a), they assume that each worker has heterogeneous skills depending on the tasks, which allows an adaptive task assignment strategy to be more advantageous than a random task assignment. Furthermore, contrary to their previous work (Ho and Vaughan, 2012), the quality of a label cannot be evaluated immediately in this model. They showed that their algorithm is near-optimal compared to an optimal offline algorithm. Chen et al. (2013) rely on the finite-horizon Markov decision process to formulate the task assignment problem. Their model is simpler than those employed by Ho and Vaughan (2012) and Ho et al. (2013) in that the correlation between a worker and task is not taken into account; the quality of a label given by a worker depends individually on the ability of the worker and the difficulty of the task. Abraham et al. (2013) formalized the combination of the task assignment and quality control problems as a bandit survey problem. The bandit survey problem focuses on inferring the true result of a single task. Each worker is modeled as a probability distribution over the possible multiple-choice results, and each worker advertises a different cost. The goal of the problem is to minimize the total cost to achieve the predefined error rate of the aggregated result of the task.

The most recent work in this literature was conducted by Bragg et al. (2014) and discusses the issue of the existing sequential task routing, focusing on parallel task routing rather than a sequential one. They point out that the sequential task routing strategy is inappropriate in that keeping workers waiting for task assignments leads to a bottleneck in the platform and makes workers frustrated and demotivated, which should be avoided, especially in a volunteer-based crowdsourcing platform. They developed algorithms for both offline and online settings, exploiting the submodularity of the objective function. They assume that the difficulty of tasks and the ability of workers are known a priori.

Our study in Chapter 3 deals with the task assignment problem for general tasks that requires workers to gain special skills in the same way as the recent work. Our problem setting is different from the existing problem settings in three ways. First, our goal is to investigate whether privacy can be preserved in a task assignment problem while retaining the optimality of the assignment; the existing ones aim to develop assignment strategies for different crowdsourcing models. This difference makes our study unique in this area. Second, we assume that the quality of a task result is deterministic, while other studies assume that the quality is drawn from a probability distribution. Because our objective is to investigate whether it is possible to preserve privacy in task assignment, we employ a simpler crowdsourcing model to avoid the difficulty of handling the uncertainty. Third, our algorithm is a batch algorithm, while other studies propose an online algorithm. We consider that it is impossible to preserve privacy in an online task assignment algorithm, because adaptively selecting workers requires estimating their skills, which is private information.

6.3 Privacy Preservation

Finally, we briefly discuss the relationship of this thesis with the research on privacy preservation. Because the research in this thesis applies the ideas of privacy preservation for graph algorithms and data mining algorithms, we review these research areas and discuss the novelty of our research.

6.3.1 Privacy-Preserving Graph Protocols

We review protocols that perform graph algorithms in a privacy-preserving manner, which are related to privacy preservation in task assignment (Chapter 3). To the best of our knowledge, only one study that investigated the maximum flow problem has been reported (Aly et al., 2013). These re-

searchers developed privacy-preserving Edmonds-Karp and push-relabel protocols based on a cryptosystem that offers secure addition, multiplication, and comparison. As instances of such a cryptosystem, they suggest to use secret-sharing-based software such as SEPIA (Burkhart et al., 2010). Their work differs from ours in three main respects. First, they assume that they have an encrypted network as an input on which they compute a maximum flow, whereas our protocol constructs it in a privacy-preserving manner from scratch. Second, our maximum flow protocol results in speeding up the computation time eight times by specializing in assignment networks, whereas their protocol, applicable to general networks, is not optimized for assignment networks. Third, they only provide an abstract protocol that does not specify the cryptographic roles and the communication between them. Our protocol is carefully tuned for practical applicability in the crowdsourcing environment. In particular, we invented a generally applicable solution to assign cryptographic roles using crowdsourcing. This addresses a typical issue found in implementing a cryptographic protocol that there does not exist an appropriate entity to perform the cryptographic role without invading the privacy assumptions.

Furthermore, there are two main reasons why we developed the Paillier-cryptosystem-based protocol rather than relying on the existing secret-sharing-based implementation. The first reason is that the Paillier implementation naturally induces proper load balancing among three parties. In our implementation, the platform, playing the role of the decryptor, mainly takes on the computational burden of decryption, and the crowdsourced parties, who may not have sufficient computing power, perform less heavy computation. In contrast, a secret-sharing-based implementation usually requires all the parties, including those recruited in crowdsourcing, to have the same computing power, which is less realistic in our crowdsourcing setting. The second reason is the security of the protocol in case of collusion. If PTA were to be naively implemented using a secret-sharing scheme managed by the three parties, all the sensitive information, including the feature vectors and the requirement vectors, would leak in the case of collusion. In contrast, PTA, which is implemented by the Paillier cryptosystem, would not fully leak the sensitive information; even in the most pessimistic case, the colluding three parties could only obtain the assignment network, from which the feature vectors and the requirement vectors could not be fully recovered. For example, it is difficult to extract the location information of each worker encoded in the feature vectors from the assignment network, because the vectors whose elements are permuted using the same permutation result in the same assignment network. These two reasons motivated us to implement PTA using the Paillier cryptosystem rather than secret-qsharing schemes. Other graph

algorithms have been made private by several research groups: the stable marriage algorithms (Golle, 2006; Teruya and Sakuma, 2013), shortest path algorithms (Brickell and Shmatikov, 2005; Aly et al., 2013), and PageRank and HITS algorithms (Sakuma and Kobayashi, 2009). We believe that it is possible to develop a more efficient private task assignment protocol by applying the ideas developed in previous research to accelerate the privacy-preserving algorithms.

6.3.2 Privacy-Preserving Data Mining

The concept of privacy-preserving data mining was presented independently by two groups (Agrawal and Srikant, 2000; Lindell and Pinkas, 2000) around the same time. Its basic concept is to address privacy issues caused by data mining algorithms. This research area is related to our research on worker privacy (Chapter 5), because our solution is a privacy-preserving variant of a data mining algorithm.

Among a number of studies in this area, our work is most closely related to research that uses secure protocols to execute EM algorithms (Lin et al., 2005; Yang et al., 2012). None of the existing techniques can be applied to the crowdsourcing setting because they require cyclic communication among parties, which is not practical in crowdsourcing. Moreover, none of the existing studies give formal theoretical guarantees of the security of the protocols. Some of them only prove that the information obtained in one iteration of the protocol does not invade privacy, while we prove that the information obtained in all the iterations does not invade privacy. These two points indicate the novelty of our work.

Our work is also related to secure multiparty aggregation, aiming to aggregate private data to obtain some statistics. Burkhart et al. (2010) developed SEPIA to realize the aggregation of private data of multiple parties based on Shamir's secret sharing (Shamir, 1979). This method is not suitable for crowdsourcing because it requires communication among workers. If a worker would like to communicate with another worker via the platform, the platform could reconstruct all the secret information because of the properties of secret sharing. Many other secure multiparty aggregation methods are also not suitable for crowdsourcing for the same reasons.

Chapter 7

Conclusion and Future Directions

7.1 Conclusion

The main concern of this thesis is that the use of crowdsourcing gives rise to privacy risks, which, except for several pioneering studies, have almost been dismissed in the literature of crowdsourcing research. To shed light on the privacy risks and establish the research field of PPCS, we formulated the following two research questions to take the first step toward a systematic treatment of privacy risks in crowdsourcing:

- *What types of privacy risks are present in crowdsourcing?*
- *How can we measure and control the privacy risks in crowdsourcing?*

To answer the first research question, we summarized the privacy risks present in crowdsourcing and discussed the relationship between PPCS and the existing privacy preservation research. Because a privacy risk basically occurs along with data processing, we studied each of the data processing procedures in crowdsourcing and its potential privacy risks. Further, to highlight the novelty of PPCS over the existing privacy preservation research, we explored whether the existing privacy preservation approaches are applicable to PPCS. Noting that the applicability depends on the properties of a data processing procedure, *i.e.*, its processor (*machine* or *human*) and whether its output is *specified* or *not*, we investigated the properties of each data processing procedure in crowdsourcing. As a result, we found that four types of data can cause privacy issues and that some of them cannot be handled by the existing privacy preservation methods, which emphasizes the novelty of PPCS. The findings are summarized as follows.

- Task assignment using the features of entities
  If a platform employs a push-type assignment strategy rather than the

typical pull-type one, both workers and requesters must report their features to the platform, with which the platform assigns tasks to appropriate workers based on feature matching. These features include sensitive information such as a skill set, the location, reward, and working hours. This data processing procedure is performed by a machine with the aim of computing a task assignment, and therefore, it is possible to apply the cryptographic approach.

- Task processing using a job instruction and an instance
  When a requester submits a task, s/he has to send two types of data to workers, a job instruction and an instance. Because a job instruction compiles what the requester intends, workers can infer, for example, the future business direction of the requester and his/her identity from it. An instance corresponds to an image, document, video, or audio clip, for example, and it often contains sensitive information regarding the requester and third parties. Both job instructions and instances are processed by a human to produce a task result. These data processing procedures cannot be handled by the existing approaches because the processor is a human.

- Delivery of task results
  After a worker finishes performing tasks, s/he sends results of the tasks to the requesters. In a location-based task, a result itself is sensitive, and even in other tasks, we can infer sensitive attributes of workers from the data, including the ability. In general, the further task result process is not fixed, and therefore, it is impossible to apply the existing approaches. However, with careful observation, we notice that in many cases, task results are processed by a machine to control their quality. In such a case, a cryptographic approach is appropriate.

Given the privacy risk analysis above, we conducted three studies, addressing the privacy issues associated with the four types of data.

- PTA protocol (Chapter 3)
  At the initial state of the PTA protocol, each entity keeps his/her features secret. The PTA protocol leverages the Paillier cryptosystem to compute an optimal task assignment whilst keeping the features secret. Finally, only the platform knows the assignment for task routing.

  With this protocol, the features of entities are kept as secret as possible. Although the assignment leaks some information about the features, it cannot be helped if tasks are to be routed to appropriate workers. In addition to keeping features secret, the protocol limits the information

leakage from job instructions and instances compared to the leakage incurred by the popular pull-type assignment strategy. The pull-type strategy allows all the workers to browse job instructions and instances, while our task assignment opens it only to those who are assigned to the task.

- UPTA (Chapter 4)
  Because a general IPP protocol sacrifices utility to preserve instance-privacy, it is indispensable to evaluate the trade-off between them. We developed UPTA, which enables us to evaluate the trade-off quantitatively. Our idea is to model the task execution and privacy invasion as samplings from the respective probability distributions. Because the models can be estimated using a real crowdsourcing platform, it is possible to compute divergence-based utility and privacy measures using the estimated models.

  As a case study, we examined the properties of an IC protocol. The IC protocol, instead of submitting an original instance, clips the instance and asks workers to perform the task on the clipped instances. From the experimental results, we conclude that the IC protocol can preserve instance privacy without notably degrading the quality for a pair consisting of a local task and global privacy definition. We also found that UPTA is consistent with standard performance measures, which validates its use.

- WPLC protocol (Chapter 5)
  At the initial state of the WPLC protocol, each worker keeps his/her results secret. The WPLC protocol leverages the Paillier cryptosystem to aggregate the results based on the update rule of the latent class method with the results kept secret. Finally, the requester only knows the aggregated results, from which the results of the workers cannot be recovered.

  By aggregating the results of multiple workers, a requester obtains one result per instance that is not associated with any worker. Although the aggregated results may still contain some sensitive information, this cannot be helped because some information is indispensable for delivering the results to the requester.

In summary, the conclusion of this thesis is the following. Privacy risks in crowdsourcing are caused by four types of data that are associated with heterogeneous data processing procedures. Some of the procedures have the

unique property that their data is processed by a human, which clearly distinguishes the PPCS research from the existing privacy preservation research. By presenting three privacy-preserving methods, we have established a research methodology for privacy-preserving crowdsourcing: to tailor a privacy preservation method for each data, taking its characteristics into consideration. Intensive research on this topic will surely end up achieving the end-to-end privacy-preserving crowdsourcing, in which any entity can regulate his/her sensitive information considering the trade-off between utility and privacy.

7.2 Future Directions

Because the purpose of this thesis is to establish the research basis of privacy-preserving crowdsourcing, a number of research opportunities are left for further research. We close the thesis with discussion on a few of the future directions from a global perspective.

First of all, as stated in the last section of each chapter, it is obviously crucial to refine each of the protocols to decrease the computational burden and privacy risk and increase their usability. In specific, for cryptography-based protocols, algorithm speed-up is crucial for deploying the protocols in real crowdsourcing services. It is also necessary to devise instance-privacy preserving protocols tailored for each of a number of tasks and privacy definitions.

Another research direction is to develop a privacy-preservation method for job instructions. Although the privacy concern associated with it has been eased by the privacy-preserving task assignment protocol, a fundamental solution is still required. The technical difficulty is that it is more difficult to generalize job instructions than task instances. On one hand, most of the task instances can be represented by arrays, which enables us to develop the IC protocol; on the other hand, job instructions do not have any standard form, which prevents us from inventing a generally-applicable solution.

It is also interesting to deliberate on jointly preserving multiple types of data. For example, it is possible to preserve privacy on a job instruction, task instance, and task result at the same time by appropriately converting and decomposing a task. Jointly preserving privacy is expected to decrease the burdens imposed by privacy preservation techniques.

Finally, it would be interesting to investigate the influence of privacy guarantees on the users of crowdsourcing. For example, it is possible to quantify the effectiveness of introducing privacy-preserving functions by counting the number of requesters and workers who are willing to participate in crowd-

sourcing if privacy is guaranteed. It is also appealing to examine its effect on the motivation of workers. We believe that such investigations will open up new research directions.

Bibliography

Ittai Abraham, Omar Alonso, Vasilis Kandylas, and Aleksandrs Slivkins. Adaptive crowdsourcing algorithms for the bandit survey problem. In *Proceedings of the 26th Annual Conference on Learning Theory*, pages 882–910, 2013.

Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 439–450, 2000.

Abdelrahaman Aly, Edouard Cuvelier, Sophie Mawet, Olivier Pereira, and Mathieu Van Vyve. Securely solving simple combinatorial graph problems. In *Proceedings of the 17th International Conference on Financial Cryptography and Data Security*, pages 239–257, 2013.

Yukino Baba and Hisashi Kashima. Statistical quality estimation for general crowdsourcing tasks. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 554–562, 2013.

Yukino Baba, Hisashi Kashima, Kei Kinoshita, Goushi Yamaguchi, and Yosuke Akiyoshi. Leveraging non-expert crowdsourcing workers for improper task detection in crowdsourcing marketplaces. *Expert Systems with Applications*, 41(6):2678–2687, 2014.

Daniel Berend and Aryeh Kontorovich. Consistency of weighted majority votes. In *Advances in Neural Information Processing Systems 27*, pages 3446–3454, 2014.

Michael Bernstein, Ed H. Chi, Lydia Chilton, Björn Hartmann, Aniket Kittur, and Robert C. Miller. Crowdsourcing and human computation: systems, studies and platforms. In *Proceedings of CHI 2011 Workshop on Crowdsourcing and Human Computation*, pages 53–56, 2011.

Big Brother Watch. New privacy concerns about Internet Eyes, 2011a. URL http://www.bigbrotherwatch.org.uk/2011/03/new-privacy-concerns-about-internet-eyes/.

Big Brother Watch. Internet Eyes falls at the first hurdle, 2011b. URL http://www.bigbrotherwatch.org.uk/2011/03/internet-eyes-falls-at-the-first-hurdle/.

Jonathan Bragg, Andrey Kolobov, Mausam, and Daniel S. Weld. Parallel task routing for crowdsourcing. In *Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing*, pages 11–21, 2014.

Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Advances in Cryptology – ASIACRYPT 2005*, pages 236–252, 2005.

Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. SEPIA: privacy-preserving aggregation of multi-domain network events and statistics. In *Proceedings of the 19th USENIX Conference on Security*, pages 223–240, 2010.

Kuang Chen, Akshay Kannan, Yoriyasu Yano, Joseph M. Hellerstein, and Tapan S. Parikh. Shreddr: pipelined paper digitization for low-resource organizations. In *Proceedings of the 2nd ACM Symposium on Computing for Development*, 2012.

Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 193–202, 2013.

Daniel Clery. Galaxy Zoo volunteers share pain and glory of research. *Science*, 333(6039):173–175, 2011.

Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, and Foldit Players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.

Cory Cornelius, Apu Kapadia, David Kotz, Dan Peebles, Minho Shin, and Nikos Triandopoulos. AnonySense: privacy-aware people-centric sensing. In *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pages 211–224, 2008.

A. P. Dawid and A. 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.

Ofer Dekel and Ohad Shamir. Vox populi: collecting high-quality labels from a crowd. In *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: a large-scale hierarchical image database. In *Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268, 2009.

Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, pages 1–12, 2006.

Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing. *ACM Computing Surveys*, 42(4):1–53, jun 2010.

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

Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940, oct 1988.

Oded Goldreich. *Foundations of cryptography: basic applications*. Cambridge University Press, 2004.

Philippe Golle. A private stable matching algorithm. In *Proceeding of Financial Cryptography and Data Security*, pages 65–80, 2006.

Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. In *Advances in Neural Information Processing Systems 24*, pages 558–566, 2011.

Christopher G. Harris. Dirty deeds done dirty cheap: a darker side to crowdsourcing. In *Proceedings of 2011 IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing*, pages 1314–1317, 2011.

Chien-ju Ho and Jennifer Wortman Vaughan. Online task assignment in crowdsourcing markets. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 45–51, 2012.

Chien-Ju Ho, Shahin Jabbari, and Jennifer Wortman Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning*, pages 534–542, 2013.

Jeff Howe. The rise of crowdsourcing. *Wired Magazine*, (14.06), 2006a.

Jeff Howe. Crowdsourcing: a definition, 2006b. URL `http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html`.

Ling Hu and Cyrus Shahabi. Privacy assurance in mobile sensing networks: go beyond trusted servers. In *Proceedings of 2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 613–619, 2010.

Kuan Lun Huang, Salil S. Kanhere, and Wen Hu. Towards privacy-sensitive participatory sensing. In *Proceedings of 2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–6, 2009.

Panagiotis G. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21, 2010a.

Panos Ipeirotis. Demographics of mechanical turk. 2010b.

Markus Jakobsson and Claus Peter Schnorr. Efficient oblivious proofs of correct exponentiation. In *Proceedings of Joint Working Conference on Communications and Multimedia Security*, pages 71–84, 1999.

M. Jordan Raddick, Georgia Bracey, Pamela L. Gay, Chris J. Lintott, Carie Cardamone, Phil Murray, Kevin Schawinski, Alexander S. Szalay, and Jan Vandenberg. Galaxy Zoo: motivations of citizen scientists. *Astronomy Education Review*, 12(1):1–41, 2013.

Shunsuke Kajimura, Yukino Baba, Hiroshi Kajino, and Hisashi Kashima. Quality control for crowdsourced POI collection. In *Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 255–267, 2015.

Hiroshi Kajino, Yuta Tsuboi, and Hisashi Kashima. A convex formulation for learning from crowds. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 73–79, 2012a.

Hiroshi Kajino, Yuta Tsuboi, Issei Sato, and Hisashi Kashima. Learning from crowds and experts. In *Proceedings of the 4th Human Computation Workshop*, pages 107–113, 2012b.

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

Hiroshi Kajino, Yukino Baba, and Hisashi Kashima. Instance-privacy preserving crowdsourcing. In *Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing*, pages 96–103, 2014b.

Hiroshi Kajino, Hiromi Arai, Jun Sakuma, and Hisashi Kashima. Privacy-preserving task assignment in crowdsourcing. Technical report, 2015.

David R. Karger, Sewoong Oh, and Devavrat Shah. Iterative learning for reliable crowdsourcing systems. In *Advances in Neural Information Processing Systems 24*, pages 1953–1961, 2011a.

David R. Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal crowdsourcing using low-rank matrix approximations. In *Proceedings of the 49th Annual Allerton Conference on Communication, Control and Computing*, pages 284–291, 2011b.

David R. Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Arxiv preprint arXiv:1110.3564*, pages 1–27, 2011c. URL `http://arxiv.org/abs/1110.3564`.

Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. 2007.

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

Leyla Kazemi and Cyrus Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 189–198, 2012a.

Leyla Kazemi and Cyrus Shahabi. TAPAS: trustworthy privacy-aware participatory sensing. *Knowledge and Information Systems*, 37(1):105–128, 2012b.

Sunyoung Kim, Christine Robson, Thomas Zimmerman, Jeff Pierce, and Eben M. Haber. Creek watch: pairing usefulness and usability for successful citizen science. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2125–2134, 2011.

Aniket Kittur, Jeffrey V. Nickerson, Michael S. Bernstein, Elizabeth M. Gerber, Aaron Shaw, John Zimmerman, Mattew Lease, and John J. Horton. The future of crowd work. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, pages 1301–1318, 2013.

C. L. Lai, K. Q. Xu, Raymond Y. K. Lau, Yuefeng Li, and Dawei Song. High-order concept associations mining and inferential language modeling for online review spam detection. In *Proceedings of 2010 IEEE International Conference on Data Mining Workshops*, pages 1120–1127, 2010.

Walter S. Lasecki, Jaime Teevan, and Ece Kamar. Information extraction and manipulation threats in crowd-powered systems. In *Proceedings of the 2014 ACM Conference on Computer Supported Cooperative Work*, pages 248–256, 2014.

Walter S. Lasecki, Mitchell Gordon, Winnie Leung, Ellen Lim, Jeffrey P. Bigham, and Steven P. Dow. Exploring privacy and accuracy trade-offs in crowdsourced behavioral video coding. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1945–1954, 2015a.

Walter S. Lasecki, Mitchell Gordon, Jaime Teevan, Ece Kamar, and Jeffrey P. Bigham. Preserving privacy in crowd-powered systems. In *Proceedings of AAMAS 2015 Workshop on Human-Agent Interaction Design and Models*, 2015b.

Matthew Lease. On quality control and machine learning in crowdsourcing. In *Proceedings of the Third Human Computation Workshop*, pages 97–102, 2011.

Hongwei Li, Bin Yu, and Dengyong Zhou. Error rate analysis of labeling by crowdsourcing. In *Proceedings of International Conference on Machine Learning 2013 Workshop: Machine Learning Meets Crowdsourcing*, 2013.

Christopher H. Lin, Mausam, and Daniel S. Weld. Crowdsourcing control: moving beyond multiple choice. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence*, pages 491–500, 2012.

Christopher H. Lin, Mausam, and Daniel S. Weld. To re(label), or not to re(label). In *Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

Xiaodong Lin, Chris Clifton, and Michael Zhu. Privacy-preserving clustering with distributed EM mixture modeling. *Knowledge and Information Systems*, 8(1):68–81, 2005.

Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Advances in Cryptology – CRYPTO 2000*, pages 36–54, 2000.

Chris J. Lintott, Kevin Schawinski, Anze Slosar, Kate Land, Steven Bamford, Daniel Thomas, M. Jordan Raddick, Robert C. Nichol, Alex Szalay, Dan Andreescu, Phil Murray, and Jan Vandenberg. Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 389(3): 1179–1189, 2008.

Helger Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Advances in Cryptology – ASIACRYPT 2003*, pages 416–433, 2003.

Greg Little and Yu-An Sun. Human OCR: insights from a complex human computation process. In *Proceedings of CHI 2011 Workshop on Crowdsourcing and Human Computation*, pages 8–11, 2011.

Qiang Liu, Jian Peng, and Alexander Ihler. Variational inference for crowdsourcing. In *Advances in Neural Information Processing Systems 25*, pages 701–709, 2012.

Zhiquan Liu, Luo Luo, and Wu-Jun Li. Robust crowdsourced learning. In *Proceedings of 2013 IEEE International Conference on Big Data*, 2013.

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*, pages 745–754, 2015.

Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1), mar 2007.

Toshiko Matsui, Yukino Baba, Toshihiro Kamishima, and Hisashi Kashima. Crowdordering. In *Proceedings of the 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 336–347, 2014.

Kaixiang Mo, Erheng Zhong, and Qiang Yang. Cross-task crowdsourcing. In *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 677–685, 2013.

Shubha U. Nabar, Krishnaram Kenthapadi, Nina Mishra, and Rajeev Motwani. A survey of query auditing techniques for data privacy. In *Privacy-Preserving Data Mining: Models and Algorithms*, pages 415–431. 2008.

Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology – EUROCRYPT 1999*, pages 223–238, 1999.

Krishna P. N. Puttaswamy, Ranjita Bhagwan, and Venkata N. Padmanabhan. Anonygator: privacy and integrity preserving data aggregation. In *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware*, pages 85–106, 2010.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010.

Jun Sakuma and Shigenobu Kobayashi. Link analysis for private weighted graphs. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 235–242, 2009.

François Schnitzler, Alexander Artikis, Matthias Weidlich, Ioannis Boutsis, Thomas Liebig, Nico Piatkowski, Christian Bockermann, Katharina Morik, Vana Kalogeraki, Jakub Marecek, Avigdor Gal, Shie Mannor, Dermot Kinane, and Dimitrios Gunopulos. Heterogeneous stream processing and crowdsourcing for urban traffic management: highlights. In *Proceedings of the 7th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, pages 520–523, 2014.

David Segal. A rave, a pan, or just a fake?, 2011. URL `http://www.nytimes.com/2011/05/22/your-money/22haggler.html`.

Sumit Shah, Fenye Bao, Chang-Tien Lu, and Ing-Ray Chen. CROWDSAFE: crowd sourcing of crime incidents and safe routing on mobile devices. In

Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pages 521–524, 2011.

Dafna Shahaf and Eric Horvitz. Generalized task markets for human and machine computation. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 986–993, 2010.

Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11): 612–613, 1979.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622, 2008.

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 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, 2008.

Latanya Sweeney. k-anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5): 557–570, 2002.

Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively learning the crowd kernel. In *Proceedings of the 28th International Conference on Machine Learning*, pages 673–680, 2011.

Tadanori Teruya and Jun Sakuma. Round-efficient private stable matching from additive homomorphic encryption. In *Proceedings of the 16th Information Security Conference*, 2013.

Hien To, Gabriel Ghinita, and Cyrus Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. In *Proceedings of the VLDB Endowment*, pages 919–930, 2014.

Hien To, Gabriel Ghinita, and Cyrus Shahabi. PrivGeoCrowd: a toolbox for studying private spatial crowdsourcing. In *Proceedings of 2015 IEEE 31st International Conference on Data Engineering*, pages 1404–1407, 2015.

Long Tran-Thanh, Sebastian Stein, Alex Rogers, and Nicholas R. Jennings. Efficient crowdsourcing of unknown experts using bounded multi-armed bandits. *Artificial Intelligence*, 214:89–111, 2014.

Daniel Trottier. Crowdsourcing CCTV surveillance on the Internet. *Information Communication & Society*, 17(5):609–626, 2014.

Lav R. Varshney. Privacy and reliability in crowdsourcing service delivery. In *Proceedings of the 2012 Annual SRII Global Conference*, pages 55–60, 2012.

Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 319–326, 2004.

Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: human-based character recognition via Web security measures. *Science*, 321(5895):1465–1468, 2008.

Yang Wang, Yun Huang, and Claudia Louis. Towards a framework for privacy-aware mobile crowdsourcing. In *Proceedings of 2013 International Conference on Social Computing*, pages 454–459, 2013.

Fabian L. Wauthier and Michael I. Jordan. Bayesian bias mitigation for crowdsourcing. In *Advances in Neural Information Processing Systems 24*, pages 1800–1808, 2011.

Peter Welinder and Pietro Perona. Online crowdsourcing: rating annotators and obtaining cost-effective labels. In *Proceedings of Workshop on Advancing Computer Vision with Humans in the Loop*, pages 25–32, 2010.

Peter Welinder, Steve Branson, Serge Belongie, and Pietro Perona. The multidimensional wisdom of crowds. In *Advances in Neural Information Processing Systems 23*, pages 2424–2432, 2010.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. Whose vote should count more: optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems 22*, pages 2035–2043, 2009.

Yan Yan, Rómer Rosales, Glenn Fung, and Jennifer Dy. Modeling multiple annotator expertise in the semi-supervised learning scenario. In *Proceedings of Conference on Uncertainty in Artificial Intelligence 2010*, pages 674–682, 2010.

Yan Yan, Rómer Rosales, Glenn Fung, and Jennifer G. Dy. Active learning from crowds. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1161–1168, 2011.

Bin Yang, Issei Sato, and Hiroshi Nakagawa. Privacy-preserving EM algorithm for clustering on social network. In *Proceedings of the 16th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 542–553, 2012.

Kan Yang, Kuan Zhang, Ju Ren, and Xuemin (Sherman) Shen. Security and privacy in mobile crowdsourcing networks: challenges and opportunities. *IEEE Communications Magazine*, 53(8):75–81, 2015.

Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *Proceedings of 2011 IEEE International Conference on Computer Vision*, pages 1331–1338, 2011.

Jinfeng Yi, Rong Jin, Shaili Jain, and Anil K. Jain. Inferring users' preferences from crowdsourced pairwise comparisons: a matrix completion approach. In *Proceedings of the First AAAI Conference on Human Computation and Crowdsourcing*, pages 207–215, 2013.

Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I. Jordan. Spectral methods meet EM: a provably optimal algorithm for crowdsourcing. In *Advances in Neural Information Processing Systems 27*, pages 1260–1268, 2014.

Yaling Zheng, Stephen Scott, and Kun Deng. Active learning from multiple noisy labelers with varied costs. In *Proceedings of 2010 IEEE International Conference on Data Mining*, pages 639–648, 2010.

Dengyong Zhou, John C Platt, Sumit Basu, and Yi Mao. Learning from the wisdom of crowds by minimax entropy. In *Advances in Neural Information Processing Systems 25*, pages 2204–2212, 2012.

Appendix A

Privacy-Preserving Task Assignment

A.1 Justification for the Maximum Flow Formulation

This section presents the proof of Proposition 3.1, which justifies the maximum flow formulation of the task assignment problem.

Proof of Proposition 3.1. We assume that there exists a maximum flow $F^\star$ on an assignment network N and the corresponding feasible assignment $\mathcal{M}^\star$ is obtained by transforming $F^\star$ following Algorithm 3.1. We further assume that there is a feasible assignment $\mathcal{M}^{\star\star}$ whose size is strictly larger than $\mathcal{M}^\star$, *i.e.*, $|\mathcal{M}^{\star\star}| > |\mathcal{M}^\star|$. We prove the proposition by contradiction; in the following, we construct a flow from $\mathcal{M}^{\star\star}$ whose value is strictly larger than the value of $F^\star$, which contradicts with the assumption that $F^\star$ is a maximum flow.

We construct a flow $F^{\star\star}$ from $\mathcal{M}^{\star\star} = \{(w_{j_k^{\star\star}}, t_{i_k^{\star\star}}) \mid k \in [|\mathcal{M}^{\star\star}|]\}$ as follows. At initialization, we set $f^{\star\star}_{u,v} = 0$ for all $(u, v) \in E$. For each $k \in [|\mathcal{M}^{\star\star}|]$, we update the flow as

$$\begin{aligned} f^{\star\star}_{s,w_{j_k^{\star\star}}} &\leftarrow f^{\star\star}_{s,w_{j_k^{\star\star}}} + 1, \\ f^{\star\star}_{w_{j_k^{\star\star}},t_{i_k^{\star\star}}} &\leftarrow f^{\star\star}_{w_{j_k^{\star\star}},t_{i_k^{\star\star}}} + 1, \\ f^{\star\star}_{t_{i_k^{\star\star}},t} &\leftarrow f^{\star\star}_{t_{i_k^{\star\star}},t} + 1. \end{aligned}$$

It is easy to see that the resultant flow satisfies the flow conditions (Definition 3.2). Then, the value of $F^{\star\star}$ is $|\mathcal{M}^{\star\star}|$, which is strictly larger than the value of $F^\star$, $|\mathcal{M}^\star|$. This contradicts the assumption that $F^\star$ is a maximum

flow. Therefore, the task assignment $\mathcal{M}^\star$ obtained from a maximum flow is proven to be a maximum assignment. □

A.2 Security Proofs

This section provides the security proofs of the protocols presented in this paper. We first specify the view of each party during execution of a protocol, *i.e.*, all of the messages received from other parties, and then, provide a simulator for the view using the input and output of the party. If a tuple of the view and private output is computationally indistinguishable from a tuple of the simulated view and private output, the party learns nothing after executing the protocol.[1]

In the following, let us denote the view of requester t_i, worker w_j, the decryptor, operator, and mixer during execution of a protocol Π by $\mathsf{view}^{\Pi}_{t_i}$, $\mathsf{view}^{\Pi}_{w_j}$, $\mathsf{view}^{\Pi}_{\mathrm{dec}}$, $\mathsf{view}^{\Pi}_{\mathrm{op}}$, and $\mathsf{view}^{\Pi}_{\mathrm{mix}}$, respectively, and the outputs by $\mathsf{output}^{\Pi}_{t_i}$, $\mathsf{output}^{\Pi}_{w_j}$, $\mathsf{output}^{\Pi}_{\mathrm{dec}}$, $\mathsf{output}^{\Pi}_{\mathrm{op}}$, and $\mathsf{output}^{\Pi}_{\mathrm{mix}}$. Let us denote a simulation of a variable A by $\tilde{A}$. Given a set A, sampling of a uniformly random variable a on A is denoted by $a \leftarrow A$.

A.2.1 Security Proof of the Conditional Test

This section provides the proof of Proposition 3.2, stating the security of the conditional test.

Proof of Proposition 3.2. We specify the view of each party and construct a simulator accordingly that outputs a computationally indistinguishable view.

(i) Decryptor

The view of the decryptor[2] is $\mathsf{view}^{\mathsf{COND}}_{\mathrm{dec}} = \{\mathsf{Enc}(T), T\}$, where

$$\begin{aligned} T = \{&s_{\sigma'(1)}(l_{\pi\circ\sigma(1)} - l_{\pi\circ\sigma(2)} - \sigma'(1)),\\ &s_{\sigma'(2)}(l_{\pi\circ\sigma(1)} - l_{\pi\circ\sigma(2)} - \sigma'(2)), \ldots,\\ &s_{\sigma'(L-1)}(l_{\pi\circ\sigma(1)} - l_{\pi\circ\sigma(2)} - \sigma'(L-1))\}, \end{aligned}$$

random permutations $\sigma' : [L-1] \to [L-1]$ and $\pi : [2] \to [2]$ and random variables $\{s_i\}_{i=1}^{L-1}$ are secretly chosen by the operator during

[1] Since the output of our protocol and the output of the corresponding functionality are identically distributed, we do not distinguish between them in the paper.

[2] Note that during the protocol the decryptor receives $\mathsf{Enc}(T)$ only. We added T to the view because the decryptor has the secret key to decrypt $\mathsf{Enc}(T)$.

INEQ, and random permutation $\sigma : [2] \to [2]$ is secretly chosen by the mixer. T is distributed according to

$$T = \begin{cases} \left\{t_i \leftarrow \mathbb{Z}_{N^2}^*\right\}_{i=1}^{L-1} & \text{with probability } \frac{1}{2}, \\ \left\{t_{\rho(i)} \leftarrow \mathbb{Z}_{N^2}^*\right\}_{i=1}^{L-2} \cup \left\{t_{\rho(L-1)} = 0\right\} & \text{with probability } \frac{1}{2}, \end{cases}$$

where $\rho : [L-1] \to [L-1]$ is a random permutation.

Then, let us construct a simulator for it. T can be perfectly simulated by generating it according to the above distribution. Considering that the decryptor outputs nothing, the decryptor learns nothing.

(ii) Operator

The view of the operator is $\mathsf{view}_{\mathrm{op}}^{\mathsf{COND}} = \{S', a\}$, where

$$\begin{aligned} S' = \Big\{ &\left[\mathsf{Enc}\left(m_{\sigma(1)}; r'_{\sigma(1),1}\right) \quad \mathsf{Enc}\left(l_{\sigma(1)}; r'_{\sigma(1),2}\right)\right], \\ &\left[\mathsf{Enc}\left(m_{\sigma(2)}; r'_{\sigma(2),1}\right) \quad \mathsf{Enc}\left(l_{\sigma(2)}; r'_{\sigma(2),2}\right)\right] \Big\}, \\ a = \mathsf{INEQ}\left(\mathsf{Enc}\left(l_{\sigma(1)}; r'_{\sigma(1),2}\right), \mathsf{Enc}\left(l_{\sigma(2)}; r'_{\sigma(2),2}\right)\right). \end{aligned}$$

Random variables $r'_{1,1}$, $r'_{1,2}$, $r'_{2,1}$, and $r'_{2,2}$ and random permutation $\sigma : [2] \to [2]$ are secretly chosen by the mixer.

Then, let us construct a simulator for it. Let us denote the private output of the operator by $c^\star$. First, we generate a simulation of a, *i.e.*, $\tilde{a}$ by sampling it from the Bernoulli distribution with parameter $1/2$. We then generate $\tilde{S}'$ as follows:

$$\tilde{S}' = \begin{cases} \left\{\begin{bmatrix} c^\star & c_{1,2} \end{bmatrix}, \begin{bmatrix} c_{2,1} & c_{2,2} \end{bmatrix}\right\} & \text{if } \tilde{a} = 1, \\ \left\{\begin{bmatrix} c_{1,1} & c_{1,2} \end{bmatrix}, \begin{bmatrix} c^\star & c_{2,2} \end{bmatrix}\right\} & \text{if } \tilde{a} = 0, \end{cases}$$

where $c_{1,1}$, $c_{1,2}$, $c_{2,1}$, and $c_{2,2}$ are randomly sampled from $\mathbb{Z}_{N^2}^*$. a and $\tilde{a}$ are identically distributed because σ, which determines the value of a, is a random permutation on $\{1, 2\}$ secretly chosen by the mixer. $(\{S', a\}, c^\star)$ and $(\{\tilde{S}', \tilde{a}\}, c^\star)$ are computationally indistinguishable because $c^\star$ in $\tilde{S}'$ and that in S' are identically distributed, and other elements in $\tilde{S}'$ are computationally indistinguishable from those in S' due to the re-encryption and shuffle performed by the mixer. Therefore, the operator learns nothing.

(iii) Mixer

The view of the mixer is $\mathsf{view}_{\mathrm{mix}}^{\mathsf{COND}} = \{S\}$, and the mixer outputs nothing. Since each element of S is a ciphertext, S is computationally

indistinguishable from uniformly random variables on $\mathbb{Z}^*_{N^2}$. Therefore, the mixer learns nothing.

Summarizing (i), (ii), and (iii), after execution of the conditional test, none of the parties learn anything.

We further show that the private output of the operator cannot be distinguished from a random variable on $\mathbb{Z}^*_{N^2}$. Since the elements in S' are encrypted and shuffled, they cannot be distinguished from random variables on $\mathbb{Z}^*_{N^2}$. The private output is one of the elements in S', and therefore, it is also computationally indistinguishable from a random variable on $\mathbb{Z}^*_{N^2}$. Therefore, the operator learns nothing from his/her private output.

□

A.2.2 Security Proof of PTA

This section presents the proofs of Lemmata 3.2, 3.3, and 3.4.

Proof of Lemma 3.2. The security of Lemma 3.2 is trivial given that no party communicates with each other. □

Proof of Lemma 3.3. Let us denote the private network construction protocol by PNC.

(i) Worker w_j

Since worker w_j does not receive any message during the protocol, worker w_j learns nothing.

(ii) Requester t_i

The view of requester t_i is $\mathsf{view}^{\mathsf{PNC}}_{t_i} = \{\mathsf{Enc}(\mathbf{s}_j)\}_{j\in[J]}$, and none of the requesters output anything. Since each element of the view is computationally indistinguishable from a random ciphertext, requester t_i learns nothing.

(iii) Operator

The view of the operator is

$$\mathsf{view}^{\mathsf{PNC}}_{\mathrm{op}} = \Big\{\{\mathsf{Enc}(L_i)\}_{i\in[I]}, \{\mathsf{Enc}(M_j)\}_{j\in[J]}, \big\{\mathsf{Enc}(\|\mathbf{r}_i\|_2^2 - \mathbf{s}_j\cdot\mathbf{r}_i)\big\}_{i\in[I],j\in[J]}\Big\} \cup \mathsf{view}^{\mathsf{COND}}_{\mathrm{op}} \cup \mathsf{output}^{\mathsf{COND}}_{\mathrm{op}},$$

and the private output of the operator is

$$\mathsf{output}^{\mathsf{PNC}}_{\mathrm{op}} = \{(\mathsf{Enc}(\mathbf{C}), \mathsf{Enc}(\mathbf{F})), (\mathsf{Enc}(\mathbf{h}), \mathsf{Enc}(\mathbf{e}))\}.$$

Then, let us construct a simulator for the view. Since $\{\mathsf{Enc}(L_i)\}_{i\in[I]}$ and $\{\mathsf{Enc}(M_j)\}_{j\in[J]}$ are contained in the output $\mathsf{Enc}(\mathbf{C})$, they can be easily simulated. $\{\mathsf{Enc}(\|\mathbf{r}_i\|_2^2 - \mathbf{s}_j \cdot \mathbf{r}_i)\}_{i\in[I],j\in[J]}$ is a set of ciphertexts generated by requesters; therefore, it is computationally indistinguishable from a set of random ciphertexts. $\mathsf{view}_{\mathrm{op}}^{\mathsf{COND}}$ and $\mathsf{output}_{\mathrm{op}}^{\mathsf{COND}}$ are computationally indistinguishable from random ciphertexts as shown in Proposition 3.2. Therefore, the operator learns nothing.

(iv) Decryptor
The view of the decryptor is $\mathsf{view}_{\mathrm{dec}}^{\mathsf{PNC}} = \mathsf{view}_{\mathrm{dec}}^{\mathsf{COND}}$, from which the decryptor learns nothing as shown in Proposition 3.2.

(v) Mixer
The view of the mixer is $\mathsf{view}_{\mathrm{mix}}^{\mathsf{PNC}} = \mathsf{view}_{\mathrm{mix}}^{\mathsf{COND}}$, from which the mixer learns nothing as shown in Proposition 3.2.

Therefore, after the execution of PNC, none of the parties learn anything. □

Proof of Lemma 3.4. Let us denote the private push-relabel protocol by PPR.

(i) Operator
The view of the operator consists of a fixed number of tuples of the view and output of the conditional test protocol, because during PPR the operator does not receive any message outside the conditional test. The operator outputs nothing. Consider a simulator generating the same number of random ciphertexts as the messages the operator receives during the protocol. Proposition 3.2 guarantees that the output of the simulator is computationally indistinguishable from the view, and therefore, the operator learns nothing.

(ii) Decryptor
The view of the decryptor consists of the views of the conditional test protocol and $\mathsf{Enc}(\mathbf{F})$, and the output is a maximum task assignment $\mathcal{A}^\star$. Considering that the decryptor learns nothing from the conditional test protocol and $\mathsf{Enc}(\mathbf{F})$ can be easily simulated from $\mathcal{A}^\star$, the decryptor learns nothing.

(iii) Mixer
The view of the mixer consists of a fixed number of the views of the conditional test protocol. Therefore, the mixer learns nothing.

Therefore, after the execution of PPR, no party learns anything.

□

A.3 Analysis of the Push-Relabel Algorithm on the Assignment Network

This section provides a new analysis of the push-relabel algorithm (Goldberg and Tarjan, 1988), applied to the assignment network. We derive tighter upper bounds on the numbers of push and relabel operations, leveraging the structure of the assignment network. Corollaries A.1 and A.2 respectively give upper bounds on the number of relabel and push operations.

First, we evaluate the upper bound on the number of relabel operations. Since the number of relabel operations is closely related to the upper bounds on heights, we analyze them in Lemma A.1.

Lemma A.1 (Upper bounds on heights)**.** *Let $N = (V, E, C)$ be an assignment network. After executing the push-relabel algorithm on N, the height of worker $w_j \in \mathcal{W}$ satisfies*

$$h_{w_j} \leq |V| + 1, \tag{A.1}$$

and the height of task $t_i \in \mathcal{T}$ satisfies

$$h_{t_i} \leq |V| + 2. \tag{A.2}$$

Proof. Since the heights are non-decreasing during the algorithm, we upper-bound the heights when the push-relabel algorithm terminates. We first analyze h_{w_j} for two cases:

(i) $w_j \in \mathcal{W}$ such that $0 \leq f_{s,w_j} < c_{s,w_j}$
Given that $f_{s,w_j} = c_{s,w_j}$ at the initial state, a push operation $\textsc{Push}(w_j, s)$ was performed during the algorithm, which implies that the height of w_j satisfied $h_{w_j} = |V| + 1$ at that time. Consider the last push operation from vertex w_j to source s. After the push operation, vertex w_j went inactive, and no relabel operation was performed on vertex w_j until the algorithm terminated. Therefore, in this case, $h_{w_j} = |V| + 1$ holds.

(ii) $w_j \in \mathcal{W}$ such that $f_{s,w_j} = c_{s,w_j}$
Assume that $h_{w_j} \geq |V| + 2$, and we prove Inequality (A.1) by contradiction. When the height of vertex w_j became $h_{w_j} \geq |V| + 2$ by a relabel operation, edge (w_j, s) must not be residual, which implies that the flows entering vertex w_j equal 0. Consequently, vertex w_j was inactive when the relabel operation was performed, which violates the applicability condition of the relabel operation. Therefore, in this case, $h_{w_j} \leq |V| + 1$ holds.

By combining these two cases, we obtain Inequality (A.1). We then analyze h_{t_i} in two cases to prove Inequality (A.2):

(i) $t_i \in \mathcal{T}$ such that $0 \leq f_{t_i,t} < c_{t_i,t}$
Assume that $h_{t_i} \geq 2$, and we prove $h_{t_i} \leq 1$ by contradiction. When the height of vertex t_i became $h_{t_i} \geq 2$ by a relabel operation, edge (t_i, t) must not be residual, which implies that $f_{t_i,t} = c_{t_i,t}$ holds. Since flow $f_{t_i,t}$ is non-decreasing while the algorithm is running, $f_{t_i,t} = c_{t_i,t}$ still holds when the algorithm terminates, which is a contradiction. Therefore, in this case, $h_{t_i} \leq 1$ holds.

(ii) $t_i \in \mathcal{T}$ such that $f_{t_i,t} = c_{t_i,t}$
Assume that $h_{t_i} \geq |V|+3$, and we prove $h_{t_i} \leq |V|+2$ by contradiction. When the height of vertex t_i became $h_{t_i} \geq |V|+3$ by a relabel operation, there must exist vertex w_j such that $h_{w_j} \geq |V|+2$, which contradicts Inequality (A.1). Therefore, in this case, $h_{t_i} \leq |V|+2$ holds.

By combining these two cases, we obtain Inequality (A.2). □

Lemma A.1 induces the upper bound on the number of relabel operations as shown in Corollary A.1.

Corollary A.1 (Upper bound on the number of relabel operations)**.** *Let $N = (V, E, C)$ be an assignment network. If the generic push-relabel algorithm initializes the heights as*

$$h_s = |V|, h_{w_j} = 2 \ (\forall w_j \in \mathcal{W}), h_{t_i} = 1 \ (\forall t_i \in \mathcal{T}), h_t = 0,$$

then, the number of relabel operations on N is upper-bounded by

$$|\mathcal{W}|(|V|-1) + |\mathcal{T}|(|V|+1).$$

Proof. Noticing that a relabel operation increases the height by at least 1, the number of relabel operations on each worker w_j is at most $|V|-1$, and that on each task t_i is at most $|V|+1$. Therefore, the total number of relabel operations is at most $|\mathcal{W}|(|V|-1) + |\mathcal{T}|(|V|+1)$. □

We then evaluate the number of push operations. Following the proof by Goldberg and Tarjan 1988, we evaluate the number of saturating push operations and the number of non-saturating push operations, respectively, in which $\textsc{Push}(v, w)$ is saturating if $c_{v,w} - f_{v,w} = 0$ after the push operation, and non-saturating otherwise. Lemmata A.2 and A.3 evaluate the number of saturating and non-saturating push operations, respectively.

Lemma A.2 (Upper bound on the number of saturating push operations)**.** *The number of saturating push operations on an assignment network is at most*

$$|\mathcal{W}||\mathcal{T}|\left(\left\lceil\frac{|V|-1}{2}\right\rceil+1\right)+|\mathcal{W}|+|\mathcal{T}|.$$

Proof. We prove the lemma for three cases.

(i) Edge (s, w_j) $(w_j \in \mathcal{W})$
At the initial state, the edge is saturated, *i.e.*, $f_{s,w_j} = c_{s,w_j}$. Since flow f_{s,w_j} is non-increasing, the number of saturating pushes along the edge is at most 1 (a push from vertex w_j to source s).

(ii) Edge (w_j, t_i) $(w_j \in \mathcal{W}, t_i \in \mathcal{T})$
At the initial state, the edge is empty, *i.e.*, $f_{w_j,t_i} = 0$, and the heights satisfy $h_{w_j} = 2$ and $h_{t_i} = 1$. We first observe that a push from w_j to t_i cannot be a saturating push because capacity c_{w_j,t_i} is set so as not to be saturated. Then, the first saturating push can occur after a push from w_j to t_i. After the push from w_j to t_i, the height of vertex t_i is relabeled to $h_{t_i} = 3$, thereby finally enabling the first saturating push to occur. The second saturating push cannot be performed until h_{w_j} increases to at least 4, there is a push from vertex w_j to vertex t_i, and h_{t_i} increases to at least 5. In other words, the height of vertex t_i must increase by at least 2 to achieve a saturating push from t_i to w_j. Therefore, the number of saturating pushes along the edge is at most $\left\lceil\frac{|V|-1}{2}\right\rceil+1$.

(iii) Edge (t_i, t) $(t_i \in \mathcal{T})$
At the initial state, the edge is empty, *i.e.*, $f_{t_i,t} = 0$. Since flow $f_{t_i,t}$ is non-decreasing, the number of saturating pushes along the edge is at most 1 (a push from vertex t_i to sink t).

By combining these three cases, we obtain the lemma. □

Lemma A.3 (Upper bound on the number of non-saturating push operations)**.** *The number of non-saturating push operations on an assignment network is at most*

$$(|V|+1)\left[|\mathcal{W}||\mathcal{T}|\left(\left\lceil\frac{|V|-1}{2}\right\rceil+1\right)+|\mathcal{W}|+|\mathcal{T}|\right].$$

Proof. Let $\Phi = \sum_{v:\text{active}} h_v$ be a potential function. At the beginning, $\Phi = 2|\mathcal{W}|$ holds, and at the end, $\Phi = 0$ holds. Since a non-saturating push decreases Φ by at least 1, we evaluate the extent to which relabel and saturating push operations increase the potential function.

First, we upper-bound the total increase in Φ caused by saturating pushes. A saturating push between edge (s, w_j) $(w_j \in \mathcal{W})$ does not increase Φ, because a saturating push from vertex w_j to source s, which does not necessarily occur, decreases Φ by $|V| + 1$. A saturating push between edge (w_j, t_i) $(w_j \in \mathcal{W}, t_i \in \mathcal{T})$ increases Φ by at most $|V| + 1$, because it activates the tail vertex w_j whose height is at most $|V| + 1$ (notice that only a push from vertex t_i to vertex w_j can be a saturating push). A saturating push between edge (t_i, t) $(t_i \in \mathcal{T})$ does not increase Φ, because sink t cannot be active from its definition. Therefore, the total increase in Φ due to saturating pushes is at most

$$|\mathcal{W}||\mathcal{T}|(|V| + 1)\left(\left\lceil \frac{|V| - 1}{2} \right\rceil + 1\right).$$

The total increase in Φ due to relabeling operations is at most

$$|\mathcal{W}|(|V| - 1) + |\mathcal{T}|(|V| + 1).$$

Considering that a non-saturating push decreases Φ by at least 1, and that $\Phi = 2|\mathcal{W}|$ at the beginning, and $\Phi = 0$ at the end, the number of non-saturating pushes is at most

$$\begin{aligned}
&|\mathcal{W}||\mathcal{T}|(|V| + 1)\left(\left\lceil \frac{|V| - 1}{2} \right\rceil + 1\right) + |\mathcal{T}|(|V| + 1) + |\mathcal{W}|(|V| - 1) + 2|\mathcal{W}| \\
=&(|V| + 1)\left[|\mathcal{W}||\mathcal{T}|\left(\left\lceil \frac{|V| - 1}{2} \right\rceil + 1\right) + |\mathcal{W}| + |\mathcal{T}|\right],
\end{aligned}$$

which proves the claim. □

By combining Lemmata A.2 and A.3, we obtain an upper bound on the number of push operations as follows.

Corollary A.2 (Upper bound on the number of push operations)**.** *Let $N = (V, E, C)$ be an assignment network. The upper bound of the number of push operations on N is*

$$(|V| + 2)\left[|\mathcal{W}||\mathcal{T}|\left(\left\lceil \frac{|V| - 1}{2} \right\rceil + 1\right) + |\mathcal{W}| + |\mathcal{T}|\right].$$

Appendix B

Worker-Privacy Preservation

B.1 Extensions to Multi-Class and Real-Valued Labels

We introduce the detailed update rules of the LC method for multi-class and real-valued labels, and then we explain how to extend the inference algorithms to preserve worker privacy.

B.1.1 Multi-Class Labels

The LC method was originally proposed for multi-class labels by Dawid and Skene (1979). Let us assume a task to give a K-class label ($K \geq 2$). For each $i \in \mathcal{I}$ and $j \in \mathcal{J}$, a crowd label $y_{i,j} \in \{0, \ldots, K-1\} (=: \mathcal{K})$ is generated by the multinomial distribution

$$\pi_{j,k,l} = p(y_{i,j} = k \mid y_i = l, \Pi_j),$$

where $\sum_{k \in \mathcal{K}} \pi_{j,k,l} = 1$ holds for all $l \in \mathcal{K}$, and we denote $\Pi_j = \{\pi_{j,k,l} \mid k, l \in \mathcal{K}\}$. For each $i \in \mathcal{I}$, the true label $y_i \in \mathcal{K}$ is generated by

$$p_l = p(y_i = l),$$

where $\sum_{l \in \mathcal{K}} p_l = 1$ holds. The model parameters $\Pi = \bigcup_{j \in \mathcal{J}} \Pi_j$ and $\{p_l \mid l \in \mathcal{K}\}$ and the posterior probabilities of the true labels $\mu_{i,l} = p(y_i = l \mid \mathcal{Y}, \Pi)$ are estimated using the following EM algorithm.

E-step: for each $i \in \mathcal{I}$, update $\{\mu_{i,l} \mid l \in \mathcal{K}\}$ as

$$\mu_{i,l} = \frac{p_l \rho_{i,l}}{\sum_{l' \in \mathcal{K}} p_{l'} \rho_{i,l'}},$$
$$\text{where } \log \rho_{i,l} = \sum_{j : y_{i,j} \neq \perp} \sum_{k \in \mathcal{K}} \mathbb{I}(y_{i,j} = k) \log \pi_{j,k,l}.$$

M-step: for each $j \in \mathcal{J}$, update Π_j as

$$\pi_{j,k,l} = \frac{\sum_{i:y_{i,j} \neq \perp} \mu_{i,l} \mathbb{I}(y_{i,j} = k)}{\sum_{i:y_{i,j} \neq \perp} \mu_{i,l}},$$

and for each $l \in \mathcal{K}$, update p_l as

$$p_l = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mu_{i,l}.$$

This algorithm can be extended to preserve worker privacy in the same way as the EM algorithm for binary labels. In the E-step, the parties calculate $\{\log \rho_{i,l} \mid i \in \mathcal{I}, l \in \mathcal{K}\}$ using our secure sum protocol, and the requester calculates and broadcasts $\{\mu_{i,l} \mid i \in \mathcal{I}, l \in \mathcal{K}\}$. In the M-step, each worker j calculates $\{\pi_{j,k,l} \mid k, l \in \mathcal{K}\}$, and the requester calculates $\{p_l \mid l \in \mathcal{K}\}$.

B.1.2 Real-Valued Labels

Raykar et al. (2010) extend the LC method to deal with real-valued labels. For each $i \in \mathcal{I}$ and $j \in \mathcal{J}$, a crowd label $y_{i,j} \in \mathbb{R}$ is generated by the normal distribution

$$p(y_{i,j} \mid y_i, \tau_j, \gamma) = \mathcal{N}(y_{i,j} \mid y_i, 1/\tau_j + 1/\gamma),$$

where $\tau_j (> 0)$ is the precision parameter of the normal distribution, which is interpreted as the ability of worker j, and γ works as regularization. Let us denote $1/\lambda_j := 1/\tau_j + 1/\gamma$. Assuming that the crowd labels are generated by this model, the true labels and the precision parameters are estimated by the following EM-like algorithm.

E-step: for each $i \in \mathcal{I}$, update the true label y_i as

$$y_i = \frac{\sum_{j:y_{i,j} \neq \perp} \lambda_j y_{i,j}}{\sum_{j:y_{i,j} \neq \perp} \lambda_j}.$$

M-step: for each $j \in \mathcal{J}$, update λ_j by solving

$$\frac{1}{\lambda_j} = \frac{1}{|\{i \in \mathcal{I} \mid y_{i,j} \neq \perp\}|} \sum_{i:y_{i,j} \neq \perp} (y_{i,j} - y_i)^2.$$

This algorithm can also be extended to preserve worker privacy. In the E-step, the parties calculate $\{\sum_{j:y_{i,j} \neq \perp} \lambda_j y_{i,j}, \sum_{j:y_{i,j} \neq \perp} \lambda_j \mid i \in \mathcal{I}\}$ using our secure sum protocol, and the requester calculates and broadcasts $\{y_i \mid i \in \mathcal{I}\}$. In the M-step, each worker j calculates λ_j.