Data Min Knowl Disc (2014) 28:1314–1335
DOI 10.1007/s10618-014-0352-3

Preserving worker privacy in crowdsourcing

Hiroshi Kajino · Hiromi Arai · Hisashi Kashima

Received: 11 October 2013 / Accepted: 2 May 2014 / Published online: 29 May 2014

Abstract This paper proposes a crowdsourcing quality control method with worker-privacy preservation. Crowdsourcing allows us to outsource tasks to a number of workers. The results of tasks obtained in crowdsourcing are often low-quality due to the difference in the degree of skill. Therefore, we need quality control methods to estimate reliable results from low-quality results. In this paper, we point out privacy problems of workers in crowdsourcing. Personal information of workers can be inferred from the results provided by each worker. To formulate and to address the privacy problems, we define a worker-private quality control problem, a variation of the quality control problem that preserves privacy of workers. We propose a worker-private latent class protocol where a requester can estimate the true results with worker privacy preserved. The key ideas are decentralization of computation and introduction of secure computation. We theoretically guarantee the security of the proposed protocol and experimentally examine the computational efficiency and accuracy.

Responsible editors: Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo.

H. Kajino (✉)
Department of Mathematical Informatics, Graduate School of Information Science and Technology, The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
e-mail: hiroshi_kajino@mist.i.u-tokyo.ac.jp

H. Arai
Advanced Center for Computing and Communication, RIKEN, Hirosawa 2-1, Wako, Saitama 351-0198, Japan
e-mail: hiromi.arai@riken.jp

H. Kashima
Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
e-mail: kashima@i.kyoto-u.ac.jp

Keywords Crowdsourcing · Quality control · Privacy-preserving data mining · EM algorithm

1 Introduction

Crowdsourcing is a technique to outsource tasks to a large number of unspecified workers. Because it provides us an easy way to access abundant human resources at very low cost, it is a popular solution for executing large amounts of tasks that require human intelligence. One example of crowdsourcing is *Galaxy Zoo*[1], where workers voluntarily classify large numbers of galaxies by their own eyes to contribute to the development of science. There are also crowdsourcing services designed for general purpose such as *Amazon Mechanical Turk*[2] and *CrowdFlower*[3], where workers get monetary rewards. These examples show that there exist a variety of crowdsourcing services that have different purposes and reward designs.

Although crowdsourcing has achieved great success, it has a problem that the quality of results varies significantly. The quality is not guaranteed because workers may not be skilled at the task or the task design is poor. Figure 1 depicts a small example of such results, where three workers were engaged in the same image labeling task. Worker 1 gave the correct labels, whereas worker 2 sometimes failed, and worker 3 seemed to return non-informative labels because all the labels given by worker 3 were the same. To make full use of crowdsourcing, we have to estimate the unobservable ground truths from the noisy results. The problem is known as the *quality control problem* (Lease 2011).

One of the most popular approaches to the quality control problem is a *repeated labeling technique* (Sheng et al. 2008), where redundant multiple labels are collected from different workers as shown in Fig. 1, and the true labels are estimated by aggregating them. Sheng et al. (2008) used majority voting (MV) to estimate the true labels. Based on this approach, many researchers have proposed more sophisticated aggregation methods. 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. Dawid and Skene (1979) assumed that the quality of labels depended on the ability of workers, i.e., they assumed that a worker had ability parameters and gave the same label as the true label with probability proportional to the ability parameters. The true labels are estimated in a form of weighted MV based on each worker's ability. By using the LC method, not only the true labels but also the ability of workers can be estimated, which will help requesters eliminate inferior workers, for example.

As stated above, several approaches have been proposed to address the quality control problem. However, they yield another problem: the invasion of *worker privacy*. The use of quality control methods invades the privacy of workers because requesters collect the labels from workers and estimate their abilities. Let us discuss the invasion

[1] http://www.galaxyzoo.org/.

[2] https://www.mturk.com/mturk/welcome.

[3] http://crowdflower.com/.

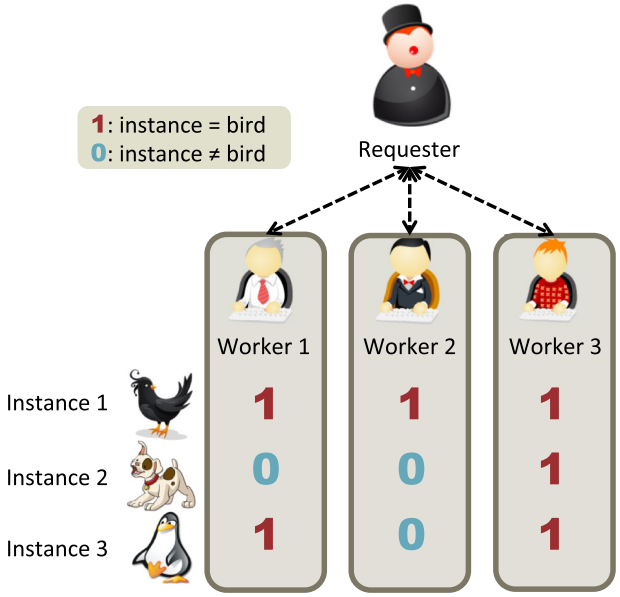

Fig. 1 Illustrative model of crowdsourcing and our protocol. We assume an image labeling task. *Each column* in the table shows labels given by each worker to the images, which do not necessarily agree with unobservable ground truths. The *dotted lines* indicate feasible communication in crowdsourcing. We develop a protocol where a requester can estimate the true labels by communicating with each worker who secretly holds his own labels

of the privacy of workers by giving three examples. The first example is a location-based task where workers are asked to submit location information of specific objects. For example, AED4[4] asks workers to submit the locations of automated external defibrillators (AEDs) in order to create a location map of AEDs. For such a task, quality control methods will be applied to obtain more accurate locations. However, such location data possibly reveal the movement history of each worker, and thus 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. Some try to *approximate* the crowd's majority opinion from a small subset of the crowd (Ertekin et al. 2012). The quality control methods can be used as an approximation method of the majority opinion by interpreting the ability of a worker as the importance of a worker. However, 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). 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. For such a task, the quality control methods will be used to increase the reliability of the results, but some workers may feel uncomfortable to be evaluated since they contribute the service without pay.

[4] http://www.aed4.eu/.

These examples clearly show that there exist privacy problems in the quality control methods and that they are mainly caused by leaking raw labels. Moreover, preserving worker's privacy can encourage more workers to participate in crowdsourcing who could not participate because of these privacy problems. Therefore, it is important to address the privacy problem. Note that in paid crowdsourcing services, preserving the privacy of workers (or keeping workers anonymous) may sometimes harm the quality of results because workers have little motivation to give high-quality results, but in non-paid crowdsourcing services, it would not harm the quality because only well-motivated workers will participate in the non-paid one. This problem is also discussed later.

To address the privacy problem, we first notice that requesters do not necessarily have to know the raw labels; it is sufficient for them to know the estimates of the true labels. Based on this idea, we define a *worker-private quality control problem* where a requester estimates the true labels preserving the privacy of workers, and we propose a *worker-private latent class protocol* (WPLC protocol) based on the LC method (Dawid and Skene 1979) to address the problem above. 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 estimate the true labels while workers can hide their labels. To validate the WPLC protocol, we provide a theoretical guarantee of its security and experimentally evaluate its computational efficiency and accuracy. Only a few have ever identified privacy problems in crowdsourcing in spite of its great importance (Bernstein et al. 2011; Varshney 2012). To the best of our knowledge, this paper deals with the privacy problem of *workers* for the first time.

To summarize, this paper makes four contributions. (i) We propose a worker-private quality control problem in crowdsourcing. (ii) We propose a protocol to execute the EM algorithm of the LC method in a privacy-preserving manner. (iii) We provide a theoretical guarantee on the security of the proposed protocol. (iv) We evaluate the performance of the proposed protocol in terms of time complexity and accuracy.

2 Quality control problem

We introduce a model of crowdsourcing, define a quality control problem, and review an existing method called the latent class method.

2.1 Problem formulation

Assume that participants in crowdsourcing consist of a single *requester* who posts tasks and J *workers* who process the tasks. Following the convention used in computer security, we call a participant a *party*. We also assume that each worker does not know the other workers who participate in the same task. Therefore, workers can communicate only with the requester as shown in Fig. 1. This assumption is reasonable in various real crowdsourcing platforms including Amazon Mechanical Turk.

As for tasks, we assume a binary labeling task for simplicity, where a task is to give a binary label to an instance. However, the proposed method can be extended to deal with multi-class or real-valued labels with slight modifications, which ensures that the proposed method is widely applicable. Section 3.3.1 and Appendix 1 describe the detailed extensions.

We then introduce notation we use in this paper. The requester has a set of task instances $\mathcal{I} := \{1, \ldots, I\}$ and wants 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. Worker $j \in \{1, \ldots, J\}(=: \mathcal{J})$ gives an unreliable label $y_{i,j} \in \{0, 1\}$ to instance i, which we call a *crowd label*. Because crowd labels are unreliable, $y_{i,j} = y_i$ does not always hold. Let $\mathcal{I}_j \subseteq \mathcal{I}$ $(\mathcal{I}_j \neq \emptyset)$ be the index set of instances to which worker j gave labels, let $\mathcal{J}_i \subseteq \mathcal{J}$ $(\mathcal{J}_i \neq \emptyset)$ be the index set of workers who gave labels to instance i, let $\mathcal{Y}_{:,j} = \{y_{i,j} \mid i \in \mathcal{I}_j\}$ be the set of labels given by worker j, let $\mathcal{Y}_{i,:} = \{y_{i,j} \mid j \in \mathcal{J}_i\}$ be the set of labels given to instance i, and let $\mathcal{Y} = \{\mathcal{Y}_{:,j} \mid j \in \mathcal{J}\}$ be the set of all the crowd labels.

The quality control problem is defined as Problem 1, where the requester aims at estimating the true labels from the crowd labels $\mathcal{Y}$ the requester collected from workers.

Problem 1 (*Quality control problem*) Assume that the requester has crowd labels $\mathcal{Y}$. The quality control problem is that the requester estimates the true labels $\mathcal{Y}^\star$ from the crowd labels $\mathcal{Y}$.

2.2 Latent class method

We review the LC method (Dawid and Skene 1979), which is a standard method for Problem 1. The true labels can be estimated by inferring an LC model, where the labeling processes of workers are modeled by introducing ability parameters of workers.

2.2.1 Latent class model

The LC model assumes that the unreliability of crowd labels depends on the ability of workers. Each instance $i \in \mathcal{I}$ is assumed to have the single unobservable true label y_i, and each worker $j \in \mathcal{J}$ independently gives a label $y_{i,j}$ to instance i according to

$$\alpha_j = \Pr[y_{i,j} = 1 \mid y_i = 1, \theta_j] \text{ and } \beta_j = \Pr[y_{i,j} = 0 \mid y_i = 0, \theta_j],$$

where let $\theta_j = \{\alpha_j, \beta_j\}$ be the ability parameters of worker j. It is also assumed that y_i $(i \in \mathcal{I})$ is generated from $\Pr[y_i \mid p] = p^{y_i} \cdot (1-p)^{1-y_i}$. Let $\Theta = \{p\} \cup \{\theta_j \mid j \in \mathcal{J}\}$ be the set of all the model parameters. We assume that the crowd labels $\mathcal{Y}$ are generated from this model.

2.2.2 EM algorithm for inference

The LC method employs the maximum likelihood estimator to estimate the model parameters Θ. Due to the discrete latent variables $\mathcal{Y}^\star$ in the model, it is intractable to obtain the maximum likelihood estimator. Therefore, an EM algorithm is used to obtain an approximation of the maximum likelihood estimator, where an expectation-step (E-step) and a maximization-step (M-step) are repeated alternately until convergence. Intuitively, the E-step is the estimation of the true labels using MV weighted by estimated abilities of workers, and the M-step is the evaluation of abilities of workers using the estimated true labels. Repeating these two steps 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 $\mu_i = \Pr[y_i = 1|\mathcal{Y}, \Theta]$ as

$$\mu_i \leftarrow \frac{pa_i}{pa_i + (1-p)b_i},$$
$$\text{where } a_i = \prod_{j \in \mathcal{J}_i} \alpha_j^{y_{i,j}} (1-\alpha_j)^{1-y_{i,j}} \quad \text{and} \quad b_i = \prod_{j \in \mathcal{J}_i} \beta_j^{1-y_{i,j}} (1-\beta_j)^{y_{i,j}}.$$

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 $\alpha_j \leftarrow \frac{\sum_{i \in \mathcal{I}_j} \mu_i y_{i,j}}{\sum_{i \in \mathcal{I}_j} \mu_i}$ and $\beta_j \leftarrow \frac{\sum_{i \in \mathcal{I}_j} (1-\mu_i)(1-y_{i,j})}{\sum_{i \in \mathcal{I}_j} (1-\mu_i)}$ so that these parameters maximize the Q-function shown below:

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

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 $O(IJ)$.

3 Worker-private quality control problem

In this section, we propose a *worker-private quality control problem* by extending the quality control problem. Then, we propose a *worker-private latent class method* (WPLC method) by extending the latent class method to address the worker-private quality control problem.

3.1 Problem formulation

First, we define the privacy requirements and formulate the worker-private quality control problem. The privacy problems do not happen if each worker stores his labels secretly and if none of the parties can obtain the labels of other parties. Based on this idea, we define worker privacy and the worker-private quality control problem as

Definition 1 and Problem 2, respectively. In Definition 1, the value can be a label $y_{i,j}$ or the ability parameters $\{\alpha_j, \beta_j\}$.

Definition 1 (*Worker-private*) Assume that worker $j \in \mathcal{J}$ 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 2 (*Worker-private quality control problem*) Assume that each worker $j \in \mathcal{J}$ has his labels $\mathcal{Y}_{:,j}$ secretly. The worker-private quality control problem is that the requester estimates the true labels $\mathcal{Y}^\star$ from crowd labels $\mathcal{Y}$ keeping all the crowd labels and the ability parameters worker-private.

Note that 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.

3.2 Proposed protocols

We propose a worker-private latent class protocol (WPLC protocol) to address the worker-private quality control problem (Problem 2). The protocol is a privacy-preserving inference algorithm for the LC model. We first introduce the WPLC protocol, the main protocol in this paper, and then we introduce our secure sum protocol, which is used as a sub-protocol in the main protocol.

3.2.1 Worker-private latent class protocol

The WPLC protocol is shown in Protocol 1, which executes the inference of the LC method in a privacy-preserving way. It allows the requester to obtain the estimated true labels and workers to keep their labels and ability parameters private. Update rules are basically the same as the original inference algorithm of the LC method, and therefore the same results can be obtained. In fact, the E-step and the M-step correspond to the seventh and eighth lines and the fourth and fifth lines in Protocol 1, respectively. The main ideas for privacy preservation are twofold: decentralization of computation to all the parties and introduction of a secure sum protocol. The M-step is decentralized to workers, and the E-step is decentralized to all the parties with introduction of secure sum. The necessity of these modifications is discussed in Sect. 3.3, and we show that public outputs obtained during the execution of the WPLC protocol do not reveal any private information in Sect. 4.

3.2.2 Secure sum protocol

We propose a secure sum protocol (Protocol 2) which allows the requester to obtain the sum of values workers have secretly in a crowdsourcing setting. We first introduce

Protocol 1 Worker-private latent class protocol

Parties: the requester and workers $\mathcal{J}$.
Private input of worker $j \in \mathcal{J}$: $\mathcal{Y}_{:,j}$ and $\mathcal{I}_j$.
Private input of the requester: $\mathcal{J}$.

Public output: $\mu_i^{(t)}$ for all $i \in \mathcal{I}, t \in \mathbb{N}$.
Private output of worker j: $(\alpha_j^{(t)}, \beta_j^{(t)})$ for all $t \in \mathbb{N}$.
Private output of the requester: $(a_i^{(t)}, b_i^{(t)})$ for all $t \in \mathbb{N}$.

1: The requester updates $t \leftarrow 0$ and broadcasts it.
2: Parties calculate $\mu_i^{(t)} = \frac{1}{|\mathcal{J}_i|}\sum_{j\in\mathcal{J}_i} y_{i,j}$ for each $i \in \mathcal{I}$ using Protocol 2, and the requester obtains the results.
3: The requester broadcasts $\{\mu_i^{(t)} \mid i \in \mathcal{I}\}$.
4: **repeat**
5: The requester updates $p^{(t)} \leftarrow \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}} \mu_i^{(t)}$.
6: Each worker $j \in \mathcal{J}$ updates $\alpha_j^{(t)} \leftarrow \frac{\sum_{i\in\mathcal{I}_j} \mu_i^{(t)} y_{i,j}}{\sum_{i\in\mathcal{I}_j} \mu_i^{(t)}}$ and $\beta_j^{(t)} \leftarrow \frac{\sum_{i\in\mathcal{I}_j} (1-\mu_i^{(t)})(1-y_{i,j})}{\sum_{i\in\mathcal{I}_j} (1-\mu_i^{(t)})}$.
7: The requester updates and broadcasts $t \leftarrow t+1$.
8: Using Protocol 2, parties calculate

$$\log a_i^{(t)} \leftarrow \sum_{j\in\mathcal{J}_i} (y_{i,j} \log \alpha_j^{(t-1)} + (1-y_{i,j}) \log(1-\alpha_j^{(t-1)})),$$
$$\log b_i^{(t)} \leftarrow \sum_{j\in\mathcal{J}_i} ((1-y_{i,j}) \log \beta_j^{(t-1)} + y_{i,j} \log(1-\beta_j^{(t-1)})),$$

for each $i \in \mathcal{I}$, and the requester obtains the results.
9: The requester updates and 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 $i \in \mathcal{I}$.
10: The requester calculates the Q-function $Q^{(t)}$ and checks its convergence.
11: **until** $|Q^{(t)} - Q^{(t-1)}|/|Q^{(t)}| < \epsilon$

Protocol 2 Secure sum protocol

Parties: the requester and workers $\mathcal{J}$.
Private input of worker $j \in \mathcal{J}$: $v_j \in \mathbb{R}$.
Public input: $L \in \mathbb{Z}_N$ s.t. $v_j L \in \mathbb{Z}_N$ for all $j \in \mathcal{J}$.
Private output of the requester: $\sum_{j\in\mathcal{J}} v_j$.
Setup: parties use the $(J+1, \theta)$-threshold additive homomorphic cryptosystem that satisfies $\theta \leq J$.
Secure sum:
1: Each worker $j \in \mathcal{J}$ encrypts $c_j \leftarrow \mathrm{Enc}(v_j L)$.
2: Each worker $j \in \mathcal{J}$ sends c_j to the requester.
3: The requester calculates $\chi \leftarrow \prod_{j\in\mathcal{J}} c_j$ $(= \mathrm{Enc}(\sum_{j\in\mathcal{J}} v_j L))$.
Decryption:
1: The requester randomly chooses a subset $\tilde{\mathcal{J}} \subseteq \mathcal{J}$ that satisfies $|\tilde{\mathcal{J}}| = \theta - 1$. Then the requester sends χ to $\tilde{\mathcal{J}}$ and asks $\tilde{\mathcal{J}}$ for sending decryption shares.
2: Each worker $\tilde{j} \in \tilde{\mathcal{J}}$ sends his decryption share $\mathrm{Dec}_{\tilde{j}}(\chi)$ to the requester, and the requester calculates his decryption share $\mathrm{Dec}_0(\chi)$.
3: The requester calculates $\sum_{j\in\mathcal{J}} v_j L$ using decryption shares $\{\mathrm{Dec}_{\tilde{j}}(\chi)\}_{\tilde{j}\in\{0\}\cup\tilde{\mathcal{J}}}$.
4: The requester calculates $\sum_{j\in\mathcal{J}} v_j \leftarrow \frac{1}{L}\sum_{j\in\mathcal{J}} v_j L$.

the properties of public key cryptosystems that are used in the proposed protocol, and then we introduce our secure sum protocol.

Public key cryptosystems We denote a plaintext by $m \in \mathbb{Z}_N$ $(= \{0, 1, \ldots, N-1\}$, and the corresponding ciphertext by $c = \mathrm{Enc}_{\mathsf{pk}}(m; r) \in \mathbb{Z}^*_{N^2} (= \mathbb{Z}_{N^2} \backslash \{0\})$, where pk is a public key, and r is a uniformly random value, which are used to encrypt the plaintext. In a public key cryptosystem, if r is selected uniformly at random from $\mathbb{Z}^*_N$, the ciphertext c is also uniformly distributed over $\mathbb{Z}^*_{N^2}$. Consequently, anyone cannot infer the original plaintext without knowing the secret key sk. We use the generalized Paillier cryptosystem (Damgård and Jurik 2001a) in our secure sum protocol because it has the properties of both an additive homomorphic cryptosystem and an (n, θ)-threshold cryptosystem. They allow us to execute summation of distributed data securely. We briefly explain both properties in the following.

An additive homomorphic cryptosystem allows us to obtain the encrypted sum of private values without decrypting the encrypted private values. The product of two ciphertexts is decrypted to the sum of their corresponding plaintexts, i.e., $\mathrm{Enc}_{\mathsf{pk}}(m_1 + m_2; r) = \mathrm{Enc}_{\mathsf{pk}}(m_1; r_1) \cdot \mathrm{Enc}_{\mathsf{pk}}(m_2; r_2)$ holds where r is uniformly random if either r_1 or r_2 is uniformly random.

An (n, θ)-threshold cryptosystem allows us to distribute secret keys to n parties and prevents any single party from decrypting ciphertexts. Efficient decryption requires cooperation among at least θ parties. In decryption, each party i calculates his decryption share $\mathrm{Dec}_{\mathsf{sk}^i}(c)$ (partial information of the original plaintexts of all the workers) using his secret key sk^i, and the ciphertext is decrypted by combining at least θ decryption shares.

Our secure sum protocol Protocol 2 describes our secure sum protocol. The approximation parameter L is introduced in order to convert values $\{v_j \in \mathbb{R} \mid j \in \mathcal{J}\}$ into integers by multiplying L, which is necessary to apply a public key cryptosystem. The proposed one is made suitable for the crowdsourcing setting, which only allows the communication between the requester and each worker. It is impossible to directly apply other secure sum protocols used to execute EM algorithms in a privacy-preserving manner (Lin et al. 2005; Yang et al. 2012) because they require message passing among workers, which is unrealistic in crowdsourcing.

3.3 Discussions

3.3.1 Extensions to multi-class and real-valued labels

We briefly introduce how to extend the WPLC protocol to deal with multi-class and real-valued labels. In the quality control problem, these types of labels can be handled by replacing the model of workers from the Bernoulli distribution to the multinomial distribution (Dawid and Skene 1979) and the normal distribution (Raykar et al. 2010), respectively. To preserve worker privacy, we only have to apply the same extensions to the modified LC methods, i.e., to execute the E-step using Protocol 2 and to decentralize the M-step to workers. Detailed update rules of the modified LC methods and extensions of them are described in Appendix 1.

3.3.2 Necessity of two modifications

We briefly argue that both decentralization and secure sum are necessary to preserve worker privacy. Decentralization is necessary because the requester cannot access crowd labels without it. The secure sum protocol is also necessary because the requester could get crowd labels $\mathcal{Y}$ in the second line of Protocol 1 without the secure sum protocol. Moreover, the requester could also obtain the parameters $\{\theta_j \mid j \in \mathcal{J}\}$ in the seventh line by using the crowd labels obtained in the second line.

3.3.3 Drawbacks

The WPLC protocol has two drawbacks compared to the LC method: numerical errors and computation time. Numerical errors are inevitable because it is difficult to find the approximation parameter L such that $v_j L$ becomes an integer for all $j \in \mathcal{J}$. Practically, the requester defines and broadcasts the approximation parameter L, and each worker j rounds $v_j L$ to an integer to execute the protocol, which causes numerical errors. However, they are shown to be tolerable in Sect. 5.2. In addition, because of numerical errors, the convergence of the WPLC protocol is not guaranteed. Therefore, in practice, it is necessary to set an upper bound on the number of iterations. The relationship between the approximation parameter L and the number of iterations is studied in the experiments.

The WPLC protocol requires more computation time than the LC method because of communication between parties, key generation, and encryption and decryption of messages. The time complexity of our secure sum is $O(J)$, and therefore, the time complexity of one iteration of the WPLC protocol is $O(IJ)$, which is the same as that of the LC method. However, the excluded coefficients in the big O notation are different. The coefficient of the dominating term of the time complexity of the WPLC protocol is bigger than that of the LC method because of the encryption and decryption steps of our secure sum protocol. Therefore, we examined the cryptographic overheads in Sect. 5.3. Communication cost is not examined because it heavily depends on the communication environment.

3.3.4 Practical implementation

The use of the WPLC protocol in a real world cannot be realized without the help of crowdsourcing platforms. Most of the present crowdsourcing platforms only support direct communication between a requester and a worker via mail services in the platforms, which prevents us from applying the WPLC protocol in a real setting. Therefore, a crowdsourcing platform will have to provide a software to execute the protocol.

3.3.5 Exclusion of spam workers

One may have a concern that preserving worker privacy makes it impossible to exclude spam workers. We believe that the protocol execution software 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 abilities of workers, and the crowdsourcing platform only learns that the ability of the worker is lower than a certain threshold.

4 Security proofs of the protocols

We prove the security of the WPLC protocol in Theorem 1. It states that crowd labels are kept worker-private after the execution of the WPLC protocol. We first make assumptions and state the security of the proposed protocols, and then we provide security proofs.

4.1 Assumptions and statements

We make two assumptions that are common in computer security: the semi-honest model and the non-collusion assumption. The semi-honest model is defined as parties who follow a protocol, which does not refer to actions outside the protocol. Therefore, in the semi-honest model, each of the parties may attempt to learn private information from the information he knows or parties may communicate with each other outside the protocol. Assuming this model ensures that the intermediate and the final outputs of a protocol can be obtained without fail. The non-collusion assumption is that parties do not collude, i.e., they do not communicate with each other outside the protocol. These two assumptions ensure that each party only knows his private information and all the intermediate values he obtained throughout the protocol execution. The validity of these assumptions is discussed in Sect. 4.3.

For convenience, we introduce a label $\perp$. $y_{i,j} = \perp$ means that worker j does not give a label to instance i. Therefore, a label takes a value in $\{0, 1, \perp\}$ in this section. Note that label $\perp$ is not imaginary; if worker j gave label $y_{i,j} = \perp$ to instance i, then the worker just encrypts value 0 in Protocol 1, which is also made secure by the encryption. Then, Theorem 1 is described as follows.

Theorem 1 (Security of the WPLC protocol) *Let* $|\mathcal{J}| \geq 3$. *For any* $i \in \mathcal{I}$, *let there exist at least one pair* $j_1, j_2 \in \mathcal{J}$ *such that* $j_1 \neq j_2$ *and* $y_{i,j_1} \neq y_{i,j_2}$. *Assume that parties execute the WPLC protocol in the semi-honest model and that parties do not collude.*

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 second assumption in Theorem 1 is not strong because usually not all the workers give labels to one instance, which implies that some of the labels must take $\perp$. For example, assuming that $|\mathcal{J}| = 3$, $\mathcal{Y}_{i,:} = \{1, 1, \perp\}$ is legal where two of three workers gave labels to instance i. This assumption is unavoidable because if all the labels given to one instance are the same, the requester can infer the workers' labels.

This theorem is built on the security of our secure sum protocol shown in Lemma 1, which states that the ciphertexts exchanged in Protocol 2 cannot be decrypted by any polynomial time attack.

Lemma 1 (Security of our secure sum protocol) *Let* $|\mathcal{J}| \geq 3$. *Assume that parties execute our secure sum protocol (Protocol 2) in the semi-honest model and that parties do not collude. After the execution of our secure sum protocol, the requester learns nothing but the sum, and the workers learn nothing.*

4.2 Proofs

In this section, we give a sketch of the proof of Lemma 1 and a proof of Theorem 1 using the result of Lemma 1.

Proof (Lemma 1) We prove Lemma 1 using the properties of our secure sum protocol introduced in Sect. 3.2.2.

All of the messages exchanged among parties in the protocol are the ciphertexts $\{\mathrm{Enc}(v_j L) \mid j \in \mathcal{J}\}$, the encrypted sum $\chi = \mathrm{Enc}(\sum_{j\in\mathcal{J}} v_j L)$, and the decryption shares $\{\mathrm{Dec}_j(\chi) \mid j \in \tilde{\mathcal{J}}\}$. Because we assume the parties execute the protocol in the semi-honest model and the parties do not collude, we have only to prove that (i) each worker cannot infer the other workers' private values from the encrypted sum χ and his private information, and (ii) a requester cannot infer each worker's private value from all of the messages.

(i) Worker
An (n, θ)-threshold cryptosystem ensures that the encrypted sum cannot be decrypted without the collusion of at least θ parties. Because of the non-collusion assumption, a worker cannot decrypt the encrypted sum. Therefore, we conclude that a worker learns nothing after the execution of our secure sum protocol.

(ii) Requester
First, the ciphertexts $\{\mathrm{Enc}(v_j L) \mid j \in \mathcal{J}\}$ cannot be decrypted by the requester because of the property of the (n, θ)-threshold cryptosystem and the non-collusion assumption. In addition, from the properties of decryption shares, the requester learns nothing but the sum from the decryption shares $\{\mathrm{Dec}_j(\chi) \mid j \in \tilde{\mathcal{J}}\}$. Therefore, we conclude that the requester learns nothing except for the sum $\sum_{j\in\mathcal{J}} v_j L$.

These two discussions complete the proof of Lemma 1. □

Proof (Theorem 1) An attacker tries to determine a private label by reverse-engineering his private information and the public outputs $\{\mu_i^{(t)} \mid i \in \mathcal{I},\ t \in \{0, 1, \ldots, T\}\}$, where let T denote the number of iterations to converge, because Lemma 1 proves parties cannot learn other variables. We prove the theorem by showing that there does not exist a function from the information an attacker has to a private label, which we call a *privacy invading function* in this proof. We give a proof by contradiction; we assume that there were a privacy invading function.

(i) Requester

We first consider the case when the requester is an attacker. The information the requester has is the public outputs and his private outputs,

$$\log a_i^{(t)} = \sum_{j \in \mathcal{J}_i} (y_{i,j} \log \alpha_j^{(t-1)} + (1 - y_{i,j}) \log(1 - \alpha_j^{(t-1)})),$$
$$\log b_i^{(t)} = \sum_{j \in \mathcal{J}_i} ((1 - y_{i,j}) \log \beta_j^{(t-1)} + y_{i,j} \log(1 - \beta_j^{(t-1)})),$$

for all $t \in \{1, \ldots, T\}$ and $i \in \mathcal{I}$. Assume that the attacker had a privacy invading function from these values to the private label to instance i' given by worker j' ($i' \in \mathcal{I}$ and $j' \in \mathcal{J}$ are arbitrary). From the assumption of Theorem 1, there exists $j'' \in \mathcal{J} \backslash \{j'\}$ such that $y_{i',j'} \neq y_{i',j''}$ holds. Let us create another set of crowd labels by permuting j' and j'' in the original set of crowd labels, and let us denote a label in the permuted set as $\tilde{y}_{i,j}$, which is related to the label in the original set as

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

The output of the privacy invading function on the permuted set should be $\tilde{y}_{i',j'}$ from the definition of the function. However, the actual output is $y_{i',j'}$ ($\neq \tilde{y}_{i',j'}$) because the input values are the same as those of the original set, which contradicts the assumption that the privacy invading function outputs a private label of instance i' given by worker j'.

(ii) Worker

Then, we consider the case when worker $j^\star \in \mathcal{J}$ is an attacker. The information worker $j^\star \in \mathcal{J}$ has is the public outputs and his private information $\{y_{i,j^\star} \mid i \in \mathcal{I}\}$, $\alpha_{j^\star}$, and $\beta_{j^\star}$. Assume that worker $j^\star$ had a privacy invading function from the information he had to the private label $y_{i',j'}$ ($i' \in \mathcal{I}$ and $j' \in \mathcal{J} \backslash \{j^\star\}$ are arbitrary). The list of workers $\mathcal{J}$ who participate in the protocol is private information of the requester, and therefore, worker $j^\star$ does not know who participates in the protocol. Let us create another set of crowd labels by conceiving of another worker $j'' \notin \mathcal{J}$ such that $y_{i',j''} \neq y_{i',j'}$ holds and permuting j' and j'' in the original set of crowd labels, and let us denote a label in the permuted set as $\tilde{y}_{i,j}$, which is related to the label in the original set as

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

The output of the privacy invading function on the permuted set should be $\tilde{y}_{i',j'}$ from the definition of the function. However, the actual output is $y_{i',j'}$ ($\neq \tilde{y}_{i',j'}$) because the input values are the same as those of the original set, which contradicts the assumption that the privacy invading function outputs a private label of instance i' given by worker j'.

In both cases, it is proven that any private label cannot be determined. □

4.3 Discussions

We discuss the validity of the assumptions we made. Both assumptions will hold except for malicious parties who mainly attempt to collect private information. As for the semi-honest assumption, the requester is expected to follow the protocol because his main purpose is to estimate the true labels, and violating the protocol will not give the proper results. Workers will be semi-honest because he will be rewarded without any cheating. The non-collusion assumption will also hold because workers do not know each other, and even if some workers should know each other, colluding will not benefit workers (collusion requires parties to share private information, which invades the worker's privacy). If workers do not collude, then the requester cannot collude because he does not have any other party to collude. Therefore, these assumptions are expected to hold in a real setting.

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 Sect. 3.3. By comparing the WPLC protocol with the LC method experimentally, we show that the WPLC protocol is practical in a real setting.

5.1 Datasets

5.1.1 Synthetic dataset

We used the LC model to generate synthetic datasets. First, for all $i \in \mathcal{I}$, we generated 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 generated 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 $\{\beta_j \mid j \in \mathcal{J}\}$.

5.1.2 Real dataset

We employed the "Duchenne Smiles Dataset" (Whitehill et al. 2009). 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.

5.2 Experiments on approximation accuracy

We investigated the approximation accuracy using both synthetic and real datasets. Approximation is caused by rounding that is inevitable to execute the protocol in a real setting.

5.2.1 Experimental settings

To understand the approximation accuracy, we examined the accuracy of the estimated true labels and the relative errors of model parameters obtained by using the WPLC protocol and the LC method.

Accuracy of the estimated true label We examined the accuracy of the estimated true 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 was examined on both synthetic and real datasets, and the effect of the parameters of datasets was examined on synthetic datasets. The performance was measured by the accuracy, i.e., the percentage of the estimated true labels that agree with the ground truth labels. For the LC method and the WPLC protocol, we estimated the true label as $y_i = 1$ if $\mu_i > 0.5$, and $y_i = 0$ otherwise. For comparison, we also tested the MV method, which is easily made secure by our secure sum protocol. The detailed parameters of the synthetic datasets were described in the captions of Fig. 2, where we used the notation $\alpha_{k:l} = \alpha^\star$ to denote $\alpha_k = \alpha_{k+1} = \cdots = \alpha_{l-1} = \alpha^\star$. For the experiments using synthetic datasets, we repeated experiments 100 times to obtain the mean and the standard deviation of the accuracy.

Relative errors of parameters We compared the relative errors of parameters obtained by using the WPLC protocol and the LC method on the real dataset, varying the approximation parameter L as $10^0, 10^1, \ldots, 10^{14}$. We define the relative error of the parameters as follows. Let us denote the parameters of the LC method by $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_J]$, $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_J]$, and $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_I]$ and those of the WPLC protocol by $\tilde{\boldsymbol{\alpha}}$, $\tilde{\boldsymbol{\beta}}$, and $\tilde{\boldsymbol{\mu}}$. For each pair of parameters $\mathbf{x} \in \{\boldsymbol{\alpha}, \boldsymbol{\beta}, p, \boldsymbol{\mu}\}$ and $\tilde{\mathbf{x}} \in \{\tilde{\boldsymbol{\alpha}}, \tilde{\boldsymbol{\beta}}, \tilde{p}, \tilde{\boldsymbol{\mu}}\}$, we define the relative error as $\| \log \mathbf{x} - \log \tilde{\mathbf{x}} \| / \| \log \mathbf{x} \|$.

5.2.2 Results

Accuracy of the estimated true label The results on the synthetic datasets are shown in Fig. 2a–f, and those on the real dataset in Table 1. We have three studies from the results. First, Fig. 2a and Table 1 show that the performance of the WPLC protocol was almost the same as that of the LC method unless the approximation parameter L was too small. Especially, Table 1 shows that rounding in secure sum had little effect on the performance in practice. Second, Fig. 2b–f show that the effect of rounding was not influenced by the parameters of datasets. Finally, Fig. 2d–f and Table 1 highlight the performance characteristics of the MV method and the LC-type methods, i.e., the LC method and the WPLC protocol. Figure 2d shows that when the abilities of workers were heterogeneous (i.e., in the middle of Fig. 2d), the LC-type methods performed better than the MV method. Figure 2e, f indicate that the LC-type methods became

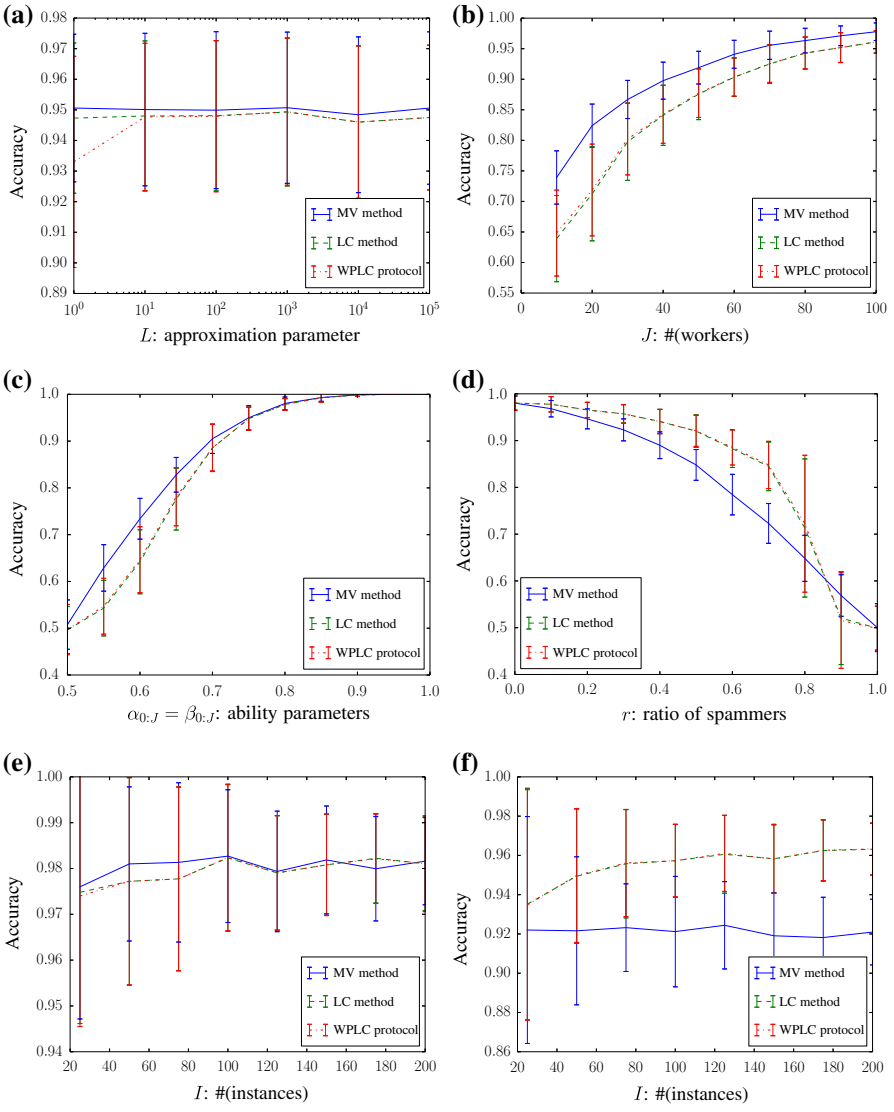

Fig. 2 Performance comparison on synthetic datasets. The parameter setting for each figure is given in the caption of it. The *error bars* show standard deviations

much superior to the MV method as we increased the number of instances in case of heterogeneous abilities of workers. They suggest that the heterogeneous abilities are necessary for the LC-type methods to beat the MV method. This is because the MV method is a special case of the LC-type methods with the assumption of homogeneous abilities of workers. It also supports this conclusion of the necessity of heterogeneity that just increasing the number of instances did not take sides with the LC-type methods in case of homogeneous abilities (Fig. 2e). In reality, crowd-generated datasets often contain workers of different abilities, which favors the LC-type methods over the MV

Table 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

method as shown in Table 1. The third study justifies the extension of the LC method, not the MV method.

From these studies, we conclude that rounding 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 justified.

Relative errors of parameters The results on the real dataset are shown in Fig. 3. The relative errors decreased as the approximation parameter L increased. Therefore, we can decrease the errors to as small as necessary by increasing the approximation parameter L, which indicates we do not have to care about the approximation accuracy if we set the approximation parameter L as large as possible. One may notice that the relative error of $\log \mu$ did not decrease drastically if the approximation parameter L was 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 Fig. 4. We can observe that the number of iterations of the WPLC protocol was different from that of the LC method when the approximation parameter L was smaller than 10^7, which would have effects on the relative error. When the approximation parameter L is too small, small changes in 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.

5.3 Experiments on computational efficiency

We examined the computational efficiency of the WPLC protocol. As stated in Sect. 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 evaluated cryptographic computation time.

5.3.1 Experimental settings

Computation time for key generation, encryption, and decryption depends on the number of parties $J + 1$ and the key length k. Usually the key length is fixed as $k = 1024$. Therefore, we examined the relationship between the computation time and the number of parties. Because the number of workers J is usually at most 100, we varied J as $3, 4, \ldots, 10, 20, \ldots, 100$ and measured the computation time of key generation and a set of encryption and decryption. We repeated the measurement in each setting of J ten times and calculated the average of the computation time. We employed the $(J + 1, \lceil \frac{2}{3}(J + 1) \rceil)$-threshold cryptosystem and set the approximation parameter $L = 10^{10}$, which does not affect the computation time. The secure sum

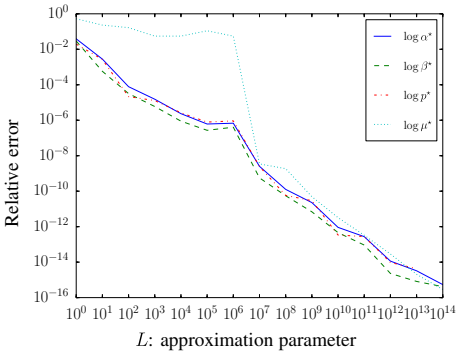

Fig. 3 Relative errors of the model parameters of the LC method and those of the WPLC protocol (L vs the relative errors)

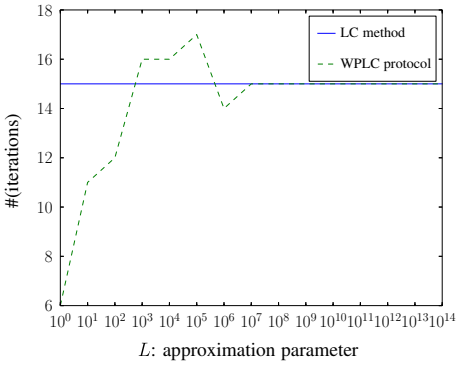

Fig. 4 Number of iterations required by the LC method and the WPLC protocol to converge (L vs the number of iterations)

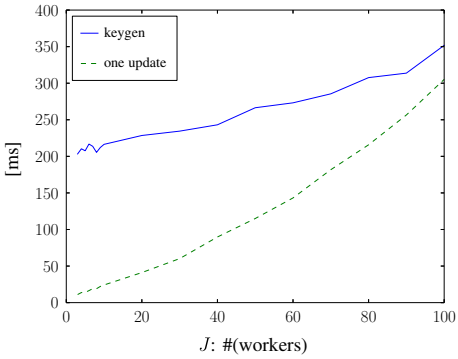

Fig. 5 Computation time for key generation (keygen) and that of encryption and decryption (one update) (J vs computation time)

protocol was implemented in Java 1.6.0 using UTD Paillier Threshold Encryption Library[5] and was run on a Linux machine with 3.20 GHz CPU and 4 GB RAM.

5.3.2 Results

The results are shown in Fig. 5, where the computation time of key generation is denoted by "keygen" and that of encryption and decryption by "one update". Both increased almost linearly as J increased. Considering that key generation is required only once and that the number of iterations was around 15 in the real dataset, we can conclude that these overheads can be tolerated. When $J = 20$ and the number of iterations was 15, which was the same condition with the execution for the real dataset, the total overhead (key generation and 15 iterations of encryption and decryption) was 789.3 ms. This also supports that the WPLC protocol is practical.

6 Related work

6.1 Crowdsourcing

As modern crowdsourcing systems have emerged, many researchers have been interested in them. The most well-studied problem is the quality control problem (Lease 2011). A basic strategy for the problem is the repeated labeling (Sheng et al. 2008) where multiple unreliable labels are given to one instance, and the true labels are estimated from the redundant unreliable labels by using statistical techniques. Although Sheng et al. (2008) used MV as a statistical technique, many researchers (Welinder et al. 2010; Whitehill et al. 2009) have sought more sophisticated methods based on the LC method (Dawid and Skene 1979). Our paper is different from these methods in that we pay attention to the privacy problem of workers whereas they do not.

There are only a few papers that deal with privacy problems in crowdsourcing. Bernstein et al. (2011) pointed out a problem of privacy in crowdsourcing for the first time. Varshney (2012) dealt with a privacy problem of task instances in crowdsourcing. Our work is different from the previous work in that we focus on privacy of *workers*.

6.2 Privacy-preserving data mining

The concept of privacy-preserving data mining was proposed by two groups (Agrawal and Srikant 2000; Lindell and Pinkas 2000) around the same time, and a number of studies have been conducted. Among them, our paper is related to research that uses secure protocols to execute EM algorithms (Lin et al. 2005; Yang et al. 2012). However, all of the existing techniques cannot be applied to the crowdsourcing setting because they require cyclic communication among parties, which is infeasible in crowdsourcing. Therefore, the WPLC protocol can be discriminated from the existing protocols.

[5] http://utdallas.edu/~mxk093120/paillier/.

Moreover, all of the existing papers did not give formal theoretical guarantees of the security of the protocols. Some of them just proved that information obtained in one iteration of the protocol does not invade privacy, while we proved that information obtained in all the iterations does not invade privacy. These two points indicate the novelty of our paper.

Also, our paper is related to secure multiparty aggregation that aims at aggregating private data to obtain some statistics. Burkhart et al. (2010) proposed SEPIA to realize aggregation of private data of multiple parties based on Shamir's secret sharing (Shamir 1979). However, this method is not suitable for crowdsourcing because it requires communication among workers. If a worker would communicate with another worker via the requester, the requester 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.

7 Conclusion

We introduced a privacy problem of workers in crowdsourcing and defined a worker-private quality control problem. To address the problem, we proposed a worker-private latent class protocol by extending the LC method. The security of the proposed protocol was theoretically evaluated, and its performance was evaluated by numerical experiments. These evaluations showed that the protocol was feasible practically.

Acknowledgments H. Kajino and H. Kashima were supported by the FIRST program.

Appendix 1: Extensions to multi-class and real-valued labels

We introduce the detailed update rules of modified LC methods to deal with multi-class and real-valued labels, and then we explain how to extend the inference algorithms to preserve worker privacy.

Appendix 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

$$\pi_{jkl} = \Pr[y_{i,j} = k \mid y_i = l, \Pi_j],$$

where $\sum_{k \in \mathcal{K}} \pi_{jkl} = 1$ holds for all $l \in \mathcal{K}$, and we denote $\Pi_j = \{\pi_{jkl} \mid k, l \in \mathcal{K}\}$. Also, for each $i \in \mathcal{I}$, the true label $y_i \in \mathcal{K}$ is generated by

$$p_l = \Pr[y_i = l],$$

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 $\mu_{il} = \Pr[y_i = l \mid \mathcal{Y}, \Pi]$ are estimated using the following EM algorithm.

E-step: for each $i \in \mathcal{I}$, update $\{\mu_{il} \mid l \in \mathcal{K}\}$ as

$$\mu_{il} = \frac{p_l \rho_{il}}{\sum_{l' \in \mathcal{K}} p_{l'} \rho_{il'}},$$
$$\text{where } \log \rho_{il} = \sum_{j \in \mathcal{J}_i} \sum_{k \in \mathcal{K}} \mathbf{I}(y_{i,j} = k) \log \pi_{jkl}.$$

M-step: for each $j \in \mathcal{J}$, update Π_j as

$$\pi_{jkl} = \frac{\sum_{i \in \mathcal{I}_j} \mu_{il} \mathbf{I}(y_{i,j} = k)}{\sum_{i \in \mathcal{I}_j} \mu_{il}},$$

and for each $l \in \mathcal{K}$, update p_l as

$$p_l = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \mu_{il}.$$

This algorithm can be extended to preserve worker privacy. In the E-step, the parties calculate $\{\log \rho_{il} \mid i \in \mathcal{I}, l \in \mathcal{K}\}$ using our secure sum protocol, and the requester calculates and broadcasts $\{\mu_{il} \mid i \in \mathcal{I}, l \in \mathcal{K}\}$. In the M-step, each worker j calculates $\{\pi_{jkl} \mid k, l \in \mathcal{K}\}$, and the requester calculates $\{p_l \mid l \in \mathcal{K}\}$.

Appendix 1.2: Real-valued labels

The LC method was modified to deal with real-valued labels by Raykar et al. (2010). Let us assume a task to give a real-valued label. 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 were 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

$$y_i = \frac{\sum_{j \in \mathcal{J}_i} \lambda_j y_{i,j}}{\sum_{j \in \mathcal{J}_i} \lambda_j}.$$

M-step: for each $j \in \mathcal{J}$, update λ_j by solving

$$\frac{1}{\lambda_j} = \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} (y_{i,j} - y_i)^2.$$

This algorithm can also be extended to preserve worker privacy. In the E-step, the parties calculate $\left\{ \sum_{j \in \mathcal{J}_i} \lambda_j y_{i,j}, \sum_{j \in \mathcal{J}_i} \lambda_j \mid i \in \mathcal{I} \right\}$ 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.

References

Agrawal R, Srikant R (2000) Privacy-preserving data mining. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp 439–450

Bernstein M, Chi EH, Chilton L, Hartmann B, Kittur A, Miller RC (2011) Crowdsourcing and human computation: systems, studies and platforms. In: Proceedings of CHI 2011 Workshop on Crowdsourcing and Human Computation, pp 53–56

Burkhart M, Strasser M, Many D, Dimitropoulos X (2010) SEPIA: privacy-preserving aggregation of multi-domain network events and statistics. In: Proceedings of the 19th USENIX Conference on Security, pp 223–240

Damgård I, Jurik M (2001) A Generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography, pp 119–136

Dawid AP, Skene AM (1979) Maximum likelihood estimation of observer error-rates using the EM algorithm. J R Stat Soc Ser C 28(1):20–28

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B 39(1):1–38

Ertekin S, Hirsh H, Rudin C (2012) Learning to predict the wisdom of crowds. In: Proceedings of Collective Intelligence 2012

Lease M (2011) On quality control and machine learning in crowdsourcing. In: Proceedings of the Third Human Computation Workshop, pp 97–102

Lin X, Clifton C, Zhu M (2005) Privacy-preserving clustering with distributed EM mixture modeling. Knowl Inf Syst 8(1):68–81

Lindell Y, Pinkas B (2000) Privacy preserving data mining. In: Advances in Cryptology-CRYPTO '00, pp 36–54

Nabar SU, Kenthapadi K, Mishra N, Motwani R (2008) A survey of query auditing techniques for data privacy. In: Privacy-Preserving Data Mining: Models and Algorithms, pp 415–431

Raykar VC, Yu S, Zhao LH, Florin C, Bogoni L, Moy L (2010) Learning from crowds. J Mach Learn Res 11:1297–1322

Shamir A (1979) How to share a secret. Commun ACM 22(11):612–613. doi:10.1145/359168.359176

Sheng VS, Provost F, Ipeirotis PG (2008) 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, pp 614–622

Sweeney L (2002) k-anonymity: a model for protecting privacy. Int J Uncertain Fuzziness Knowl Syst 10(5):557–570. doi:10.1142/S0218488502001648

Varshney LR (2012) Privacy and reliability in crowdsourcing service delivery. In: Proceedings of the 2012 Annual SRII Global Conference, pp 55–60

Welinder P, Branson S, Belongie S, Perona P (2010) The multidimensional wisdom of crowds. Adv Neural Inf Process Syst 23:2424–2432

Whitehill J, Ruvolo P, Wu T, Bergsma J, Movellan J (2009) Whose vote should count more: optimal integration of labels from labelers of unknown expertise. Adv Neural Inf Process Syst 22:2035–2043

Yang B, Sato I, Nakagawa H (2012) Privacy-preserving EM algorithm for clustering on social network. In: Advances in Knowledge Discovery and Data Mining 16th Pacific-Asia Conference, PAKDD 2012, pp 542–553