Privacy-Enhanced and Practical Truth Discovery in Two-Server Mobile Crowdsensing

Haiqin Wu, Liangmin Wang, *Member, IEEE,* Ke Cheng, *Student Member, IEEE,* Dejun Yang, *Senior Member, IEEE,* Jian Tang, *Fellow, IEEE,* and Guoliang Xue, *Fellow, IEEE*

Abstract—In mobile crowdsensing, truth discovery (TD) enables a crowdsensing server to extract truthful information from possibly conflicting crowdsensing data. TD provides a more accurate truth estimation than traditional truth inference methods like majority voting and averaging. However, there still exist crucial data privacy (including sensory data, inferred truths, and intermediates) and practicability (*e.g.*, efficiency, utility, and non-interaction) concerns in real-world crowdsensing applications. Existing researches either fail to provide adequate data privacy protection throughout the entire TD procedure or suffer from low practicability. In this paper, we propose two schemes: a basic privacy-aware TD scheme (BPTD) and a privacy-enhanced TD scheme (PETD) with two servers for mobile crowdsensing, comprehensively considering both privacy and practicability. BPTD is straightforwardly conducted on shared data with few user-side interactions, while achieving high efficiency. To further liberate mobile users and prevent disclosure of the intermediates, PETD incorporates a novel partial decryption-based Paillier Cryptosystem to work with secret sharing, offering enhanced privacy protection without relying on any user-side involvement. Additionally, we improve the efficiency of PETD via data packing. Security analysis shows the desired privacy goals. Compared to prior studies with the best security guarantees, our extensive experiments demonstrate a comparable and superior performance regarding different metrics.

Index Terms—Mobile crowdsensing, truth discovery, data aggregation, secret sharing, Paillier Cryptosystem.

1 Introduction

The past few years have witnessed the emergence and boom of mobile crowdsensing [39], [40] due to the proliferation of mobile devices equipped with powerful embedded sensors. With no geographic constraint and deployment cost, mobile crowdsensing has fostered a broad spectrum of applications in environmental monitoring, assistive healthcare, and intelligent transportation ([20], [26]). However, for specific applications, the quality of sensory data observed by different users may vary significantly due to differences in sensor quality, user expertise, and environmental noise. For example, a temperature sensor with high accuracy generally provides a more reliable measurement than another with low accuracy. A user with a malfunctioning sensor may even submit incorrect data. Hence, the sensory data provided by mobile users is not all reliable and even conflicting. As data receivers, data requesters want to obtain truthful data from user-contributed information. Accordingly, one research problem is how to extract the truthful data by aggregating unreliable information from different users. Conventional data aggregation methods like majority voting and averaging assume that users are in equal reliability, which fail to capture users' differences in reliabilities and cannot derive accurate truths ([22], [23], [30], [38]). For this issue, truth discovery (TD) [21] has recently received considerable attention in the context of mobile crowdsensing. TD aims to discover the truthful information of each sensing object from user-submitted conflicting data based on the users' weights (*i.e.*, reliabilities). The common principle in most TD algorithms is that a user is assigned a higher weight if his submitted data is closer to the aggregated results, and a user's data is counted more in truth estimation if the user has a higher weight. Compared with majority voting and averaging, TD provides more accurate truth information by iteratively estimating users' weights and updating the estimated truths based on weighted aggregation [13].

Despite these benefits, we observe several critical privacy concerns that may impede the full potential of TD application to real-world mobile crowdsensing [32], [33]. From the users' perspective, the sensory data may contain users' sensitive information (*e.g.*, location and health information [41]). For example, to evaluate new drugs' effects on patients, a research institution needs to aggregate data from the participating patients to derive the truthful value. However, the patients may not want to reveal their health data. Additionally, the weights derived in the TD

- *H. Wu was with the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, 212013, China, and is with the Department of Computer Science, University of Copenhagen, 2100 Copenhagen, Denmark (e-mail: hw@di.ku.dk).*
- *L. Wang is with the School of Cyber Science and Engineering, Southeast University, Nanjing, 211100, China (e-mail: liangmin@seu.edu.cn).*
- *K. Cheng is with the School of Computer Science and Technology, Xidian University, Xi'an, 710071, China, (e-mail: kechengstu@gmail.com).*
- *D. Yang is with the Department of Computer Science, Colorado School of Mines, Golden, CO 80401 USA (e-mail:djyang@mines.edu).*
- *J. Tang is with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244 USA (e-mail: jtang02@syr.edu).*
- *G. Xue is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: xue@asu.edu).*

This work was supported in part by NSF Grants 1717197 and 1717315, in part by the National Natural Science Foundation of China under Grants U1736216, 61472001, and 61702233, the National Key Research and Development Program Grant 2017YFB1400703. The information reported here does not reflect the policy of the funding agencies. The majority of this work was done while Wu was visiting Arizona State University under the support of the Chinese Scholarship Council (No. 201708320241).

TABLE 1
Comparison with Existing Schemes

Schemes	S. data	U. weight	E. truth	I. results	Non-I.	Secure C.	No P-U T.	Non-collusion assumption
[22], [23], [35], [36], [37]	✔	✔	✗	✗	✗	✗	✔	All parties
[24]	✔	✗	✗	✗	✗	✗	✔	All parties
[48]-I	✔	✔	✗	✗	✗	✔	✔	$\geq$Two users
[48]-II	✔	✔	✗	✗	✗	✔	✔	$\geq$ Two users, two servers
[43]-I, [44]	✔	○	✗	✗	✗	✗	✔	Two servers
[27]	✔	○	○	✗	✔	✗	✔	Two servers
[43]-II	✔	✔	✗	✗	✔	✗	✔	Two servers
[45]-I	✔	○	✗	✗	✗	✗	✔	All parties
[45]-II	✔	✗	✗	✗	✗	✗	✔	All parties
[42]-I	✔	○	✗	✗	✗	✗	✔	$\geq$ Three users, two servers
[42]-II	✔	✔	✗	✗	✔	✗	✔	$\geq$ Three users, two servers
[28]	✔	✔	✗	✔	✔	✗	✔	Most users, two servers
[15], [16], [38]	✔	✗	✗	✗	✔	✗	✗	—
[30]	✔	✔	✗	✗	✗	✗	✗	User and server, cloud and edge
[19]	✔	○	✗	✗	✗	✗	✔	Most users
[2], [3], [47]	✔	✔	✔	✔	✔	✗	✔	Two servers
BPTD	✔	✔	✔	○	✗	✔	✔	User and server, two servers
PETD	✔	✔	✔	✔	✔	✔	✔	Two servers

S., U., E., I., Non-I., C., and No P-U T. are abbreviations for sensory, user, estimated, intermediate, non-interaction, convergence, and no privacy-utility tradeoff; *Comp*. and *Comm*. stand for computation cost and communication cost, respectively; -I and -II denote the first and second solution, respectively. ○ denotes partial data disclosure and — means that this item is not mentioned.

procedures reflect user's reliability degrees, indicating some personal information such as education level and wealth condition [37], and hence should be protected as well from anyone including users themselves[1]. From the requester's perspective, he regards the final inferred truths as his proprietary property since generally he pays the server and mobile users some money for truth acquisition. Inherently, the requester wants to keep them confidential against others except for those willing to pay for access. Besides these, the confidentiality of TD intermediate results is also of great significance, but is neglected in most previous researches. Although directly deriving the sensory data or the final results is not easy based on an individual intermediate data, it is much easier if big data or machine learning techniques ([25], [29]) are applied or multiple entities collude with each other. Therefore, for stringent privacy concerns, the above private/sensitive information should be protected.

With these privacy concerns, Miao *et al*. [22] first formulated the privacy-preserving truth discovery (PPTD) problem in mobile crowdsensing. After that, a series of researches ([2], [15], [19], [23], [24], [27], [28], [35], [36], [37], [38], [42], [43], [45], [47], [48]) continued this line of work, targeting different privacy requirements or efficiency improvements based on the single-server or the two-server models. Despite their effectiveness, as shown in Table 1, these solutions still suffer from limitations. First, limited privacy protection is provided. Schemes like [22], [23], [35], [36], [37], [45] ignore the truth and intermediates privacy. Most solutions ([19], [24], [35], [36], [37], [42], [45], [48]) are vulnerable to collusion attacks between users or between users and the server. Differential privacy-based PPTD [15], [16], [30], [38] can greatly improve the TD efficiency but inevitably sacrifices the truth accuracy due to the added noise, *i.e.*, there is a tradeoff between privacy and utility. Noteworthily, for convergence determination, many PPTD schemes simply predefine a maximum number of iterations and focus on ensuring privacy in weight and truth estimation. How to securely conduct computation-involved convergence determination and preserve the privacy throughout the entire TD procedure is hardly investigated except in [48]. Second, past researches ([19], [22], [23], [24], [35], [36], [37], [45], [48]) require users' online involvement to help compute intermediate values, which is difficult for practical realization due to failure of network connections [12] or battery issues of mobile devices. Additionally, extra computation and communication overhead is incurred at the users. Recently, schemes in [2], [3], [47] achieve non-interactive PPTD with more privacy considered by combining Garbled Circuit (GC) with homomorphic encryption or additive secret sharing. However, generating the related GC is expensive and achieving secure convergence determination still remains neglected. For practicability concerns, a non-interactive PPTD scheme with no privacy-utility tradeoff and heavy cost is highly advocated. However, when the above privacy and practicability drawbacks are both taken into account, *designing a lightweight and practical PPTD scheme with a stronger security guarantee throughout the entire TD phase* is a great challenge.

In this paper, inspired by the idea of additive secret sharing [4], we initiate our first attempt and propose a **B**asic **P**rivacy-aware **T**ruth **D**iscovery scheme, named BPTD, based on the two non-colluding server model. BPTD protects not only the sensory data and user weights, but also the final inferred truths and some intermediate results. It yields high efficiency with little user privacy sacrifice and a few user-server interactions in each secure weight estimation. To mitigate these, we further craft a **P**rivacy-**E**nhanced and practical **T**ruth **D**iscovery scheme, called PETD. Secure TD protocols are proposed by making the best of the additive secret sharing and the state-of-the-art Pallier Cryptosystem

1. Revealing lower weights to users may discourage them from participation in the long term.

with partial decryption (PCPD) techniques. PETD indeed releases the mobile users after data submission, and only requires the online participation of the two servers. The main contributions of this paper are summarized as follows.

- We propose a basic scheme BPTD, which enables efficient TD on shared data while satisfying the privacy demands of users and the data requester under the non-collusion assumption. We present a delicate design of three secure sub-protocols on shared data with the fewest interactions with the mobile users.
- We further present a privacy-enhanced and practical scheme PETD, which provides stronger privacy guarantees (resistant to user-server collusion attacks) with no user-side interactions. Similarly, three secure PPTD sub-protocols are proposed on hybrid data. Moreover, the server-side cost is further reduced by seamlessly incorporating a data packing technique.
- We provide formal security and complexity analysis to demonstrate the desired security and practicality. Extensive experiments are conducted to validate the superiority of our schemes to prior solutions.

The rest of this paper is organized as follows. Section 2 reviews previous work. We formulate the main problem in Section 3. Section 4 introduces some preliminaries. Section 5 elaborates on our scheme details. Security analysis and performance evaluations are presented in Sections 6 and 7, respectively. We conclude the paper in Section 8.

2 Related work

As the first attempt, Miao *et al*. [22] designed a cloud-enabled framework for PPTD in mobile crowdsensing. The core of this framework is a secure sum protocol SSP which protects the data/weight privacy but needs user-side involvement for decryption. For a better performance, an efficient PPTD scheme was proposed in [35], [37], which however has a severe security compromise as a secret key and random numbers must be shared among users (vulnerable to user-side or user-server collusion attacks). Moreover, frequent user-side interactions are required. As an improvement, [36] employed $t-\mathcal{D}$ secret sharing to support users dropping out to some extent. Instead of employing a single server, Miao *et al*. [24] proposed two lightweight PPTD frameworks via collaboration of two non-colluding servers. Despite its low user-side cost, these two solutions are also not secure enough, as user's weight, the intermediate results, and the estimated truths are all revealed to the servers. The same disclosure problem also exists in [43], [45], [48]. In [42], [45], different solutions were proposed for user interactive and non-interactive scenarios, respectively. Similarly, [43], [44] further crafted two reliable fog-based PPTD schemes with consideration of external attacks. However, as in [19], besides direct disclosure of truths and intermediates, the weight information is also revealed to each user. Adopting DP-based methods, mechanisms [15], [16], [30], [38] were proposed to enable lightweight PPTD, which bear a trade-off between privacy and utility. Besides, the scheme (called PrivSTD) proposed in [30] integrated edge servers with the cloud server for privacy-preserving streaming TD, which also needs user-side interactions with the edge servers.

Recently, Tang *et al*. [28] made their endeavors in designing the non-interactive PPTD, which leverages Yao's GC and provides data privacy except the final estimated truths. Zheng *et al*. [47] proposed an encrypted confidence-aware TD approach by bridging additive homomorphic encryption and GC. The adoption of GC also exists in [2], [3] with an integration of additive secret sharing. It however is time-consuming to generate GC and needs approximation for logarithm computation with high accuracy. Instead of concealing the sensory data, an anonymization protocol named AnonymTD [27] was designed to delink workers from their data and it enabled PPTD without worker-side interactions. However, AnonymTD incurs a high communication overhead as the number of data submitted by each worker is the same as the number of workers. Although the deficiency is mitigated by another protocol named PerturbTD with perturbation, both AnonymTD and PerturbTD ignore secure convergence determination and the privacy of intermediates. User weights and estimated truths are also revealed to some entities.

Besides achieving PPTD, enabling privacy-aware worker recruitment [34], [46] and task allocation [31], [32] have also been active research issues in the mobile crowdsensing community. Since we aim to protect users' privacy in different mobile crowdsensing phases, we omit the detailed literature review of these two lines of work.

3 System Model & Problem Formulation

In this section, we present our system model, threat model, and the problem statement for TD in mobile crowdsensing.

3.1 System Model

Our system consists of four kinds of parties: *data requester*, *cloud platform*, *mobile users*, and *key generation center*.

1) Data requester: An individual/organization who has data sensing tasks on certain objects. Due to the limited resources, the data requester publishes his tasks on a cloud platform and relies on the platform to collect data from mobile users and conduct the TD procedure.

2) Cloud platform: An organization (or multiple organizations) who is/are responsible for distributing tasks, collecting data, and exploring the truths for the data requester. The cloud-based sensing model fosters a notable concept called *sensing as a service* [26]. In this paper, we consider that the cloud platform comprises two cloud servers managed by two non-colluding and independent organizations. Such a two-server model is also adopted by [24], [28], [43], [47].

3) Mobile users: Participants who perform tasks and collect sensory data with their mobile devices. The sensory data is sent to the cloud platform for further TD analysis.

4) Key generation center (KGC): A trusted party who generates and distributes keys to the system participants. The KGC only works in PETD for system initialization and will stay offline in the PPTD procedures. A similar entity was also introduced/mentioned for key distribution in prior cryptography-based researches [22], [23], [42], [43], [44], [45].

3.2 Threat Model

In our attack model, in line with [22], [23], [42], [43], [44], [45], the KGC is considered fully trusted. Similar to [47], we assume that the data requester, the cloud servers, and mobile users are honest-but-curious, *i.e.,* they will strictly follow the designated PPTD protocol, but try to infer private information of others. Specifically, the cloud servers may want to know the sensory data and weights of mobile users, or the estimated ground truths of the data requester, due to either curiosity or business-driven reasons (financial purposes [22], [23]). For example, the cloud servers may sell the sensory data or estimated truths to third parties for making a profit. Similarly, it is possible that the requester is not satisfied with only getting the estimated truths, but also be curious about the private information of mobile users. On the other hand, each mobile user may also want to learn others' sensory data, weight information including his own weight, and the data requester's final estimated truths.

Additionally, in realistic crowdsensing applications, some parties may collude with each other to infer private information, which is easy to launch but is more challenging to defend than single-adversary attacks. For example, in [35], [37], a user's sensory data can be easily deduced if some users and the server collude with each other, as the private key is shared between a set of users and the server. The servers may collude with each other to reveal the private data of the mobile users and the data requester. To mitigate this, economic means [6] can be employed, where the main idea is to leverage game theory and smart contracts to stimulate tension between the servers, so that rational servers will not collude and cheat because it is unprofitable and too risky. This line of work is out of the scope of this paper and we assume such an economic approach serving our schemes. Hence, as in the prior two-server models ([24], [28], [43], [47]), we follow the common non-collusion assumption about the servers.

Moreover, regarding the Dolev-Yao threat model [5] in which "active" eavesdroppers may obtain any message passing through the network, impersonate another user, or alter the message being transmitted, we assume that each data share is transmitted via secure and authenticated channels. Such an assumption can be also found in [2], [3]. Alternatively, the sender of a data share can also encode his name together with the data share in the encrypted text and sends it to the receiver. This solution is proved secure against arbitrary behavior of the active eavesdroppers in [5]. For the partially decrypted ciphertexts transmitted between the servers and the final truth ciphertexts sent to the requester, they can be transmitted via an insecure channel. However, the "active" eavesdroppers cannot discover the corresponding plaintexts by impersonating the servers as each server can only partially decrypt the ciphertexts. An "active" eavesdropper may encrypt arbitrary data with the system public key and obtain the partially decrypted ciphertext from a cloud server. The eavesdropper may then impersonate the cloud server and send the partially decrypted ciphertext to another server for further decryption, which leads to an incorrect computation result. Similarly, an "active" eavesdropper may encrypt an arbitrary data with the requester's public key and impersonate a server to return an incorrect truth ciphertext to the requester. The above attacks can be resolved by letting the sender attach a digital signature to each message, so that message authenticity and integrity is ensured.

In this paper, we define two kinds of privacy disclosure: direct privacy disclosure and indirect privacy disclosure.

Definition 1. *(Direct Privacy Disclosure). The sensory data of users and the estimated truths of the requester are known by others without relying on additional information. Moreover, the weights of users are directly revealed to the system entities including the users themselves.*

Solutions requiring the users/server to compute weights [24], [27], [43], [45] and revealing the truths to the server/users [22], [23], [27], [45], [48] all suffer from direct privacy disclosure.

Definition 2. *(Indirect Privacy Disclosure). The sensory data, weights, or estimated truths are inferred from additional information. The additional information may be obtained based on entity collusions or intermediate data disclosure.*

Collusion-vulnerable solutions [35], [36], [37] inevitably suffer from indirect privacy disclosure. As pointed out in [28], a continuous collection of the intermediates potentially reveal the users' sensory data and weights. Therefore, those with no protection of intermediate data may be susceptible to indirect privacy disclosure.

Note that the malicious attack model, in which malicious users submit invalid (*e.g.,* out-of-range) data or malicious cloud server(s) returns incorrect truths to the data requester, is not the focus of this paper. As potential solutions, reputation evaluation [33], zero-knowledge proof [7], incentive mechanism [8], [39], and verifiable computation [10] can be adaptively integrated to address these issues.

3.3 Problem Statement and Design Goals

The PPTD problem is formalized as follows: Suppose that there are M sensing tasks generated by the data requester, denoted as $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_M\}$, and there are K mobile users, denoted as $\mathcal{U} = \{u_1, u_2, \ldots, u_K\}$. Let $\mathbf{x}_k = (x_{k,1}, x_{k,2}, \ldots, x_{k,M})$ represent the data vector of user u_k, where $x_{k,m}$ denotes u_k's sensory data for task τ_m. Moreover, we use vector $\mathbf{w} = (w_1, w_2, \ldots, w_K)$ to represent the corresponding weights of users.

In our system, for each sensing task $\tau_m \in \mathcal{T}$, there is a ground truth unknown to all parties. Given the sensory data vector $\mathbf{x}_k$, $k \in [1, K]$, from a high level perspective, our objective is to enable the cloud platform to estimate the corresponding ground truths $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ of all tasks according to the users' weights $\mathbf{w} = (w_1, w_2, \ldots, w_K)$ in a privacy-aware and practical manner. For direct privacy protection, we aim to let mobile users only know their own sensory data and let the data requester only learn the inferred truths. The sensory data and estimated truths are not directly disclosed to others. A mobile user's weight cannot be directly revealed to anyone including the user himself. For indirect privacy protection, we aim to conceal the intermediate results derived in the PPTD procedure. No sensory data, weights, and truths can be inferred even if partial intermediates are exposed. For practicability, the

overhead incurred in PPTD should be as low as possible, especially on the user side. There is no privacy-utility trade-off. Moreover, we aim to minimize and ideally remove the user-server interactions during the iteration procedure.

4 PRELIMINARY

In this section, we first describe the general TD procedure in mobile crowdsensing, and then we review some cryptographic primitives as building blocks for our schemes.

4.1 Truth Discovery

The purpose of TD in mobile crowdsensing is to infer the truthful information of sensing tasks based on the massive sensory data collected by different users. The general procedure of TD algorithms is to iteratively estimate the weights of users (weight estimation) and the ground truths of tasks (truth estimation), until some predefined convergence criterion is satisfied (convergence determination).

Next, we illustrate the iterative steps by taking the representative TD algorithm CRH [13] as an example. CRH also serves as the underlying TD basis for our schemes. However, our proposed schemes are not restricted to CRH and also suits other iterative TD algorithms such as [14] with minor modifications.

1) Weight Estimation: Given all users' sensory data $\mathbf{x}_k, k \in [1, K]$ and the estimated ground truths $\mathbf{x} = (x_1, x_2, \ldots, x_M)$, the estimated weight of any user u_i, $i \in [1, K]$ can be computed as

$$\begin{aligned} w_i &= \log(\sum_{k=1}^{K} d(\mathbf{x}_k, \mathbf{x})) - \log(d(\mathbf{x}_i, \mathbf{x})) \\ &= \log(\frac{\sum_{k=1}^{K}(\mathbf{x}_k - \mathbf{x}) \cdot (\mathbf{x}_k - \mathbf{x})}{(\mathbf{x}_i - \mathbf{x}) \cdot (\mathbf{x}_i - \mathbf{x})}), \end{aligned} \tag{1}$$

where $d(\mathbf{x}_k, \mathbf{x}) = \sum_{m=1}^{M}(x_{k,m} - x_m)^2$ is the square of Euclidean distance between $\mathbf{x}_k$ and $\mathbf{x}$. In addition, $\cdot$ denotes the inner product of two vectors.

2) Truth Estimation: Given users' weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_K)$ and their sensory data, the estimated ground truth for each sensing task τ_m can be derived as

$$x_m = \frac{\sum_{k=1}^{K} w_k x_{k,m}}{\sum_{k=1}^{K} w_k} = \frac{\mathbf{w} \cdot \mathbf{x}_{*,m}}{\mathbf{w} \cdot \mathbf{s}}, \tag{2}$$

where $\mathbf{x}_{*,\mathbf{m}} = (x_{1,m}, x_{2,m}, \ldots, x_{K,m})$ and $\mathbf{s} = (1, 1, \ldots, 1)$.

3) Convergence Determination: Given a predefined threshold ϵ, $0 < \epsilon < 1$, we check the following inequality to determine whether convergence is satisfied.

$$d(\mathbf{x}^t, \mathbf{x}^{t-1}) < \epsilon, \tag{3}$$

where $\mathbf{x}^t$ and $\mathbf{x}^{t-1}$ denote estimated truth vectors in the t-th and $(t-1)$-th iterations, respectively.

4.2 Additive Secret sharing and Multiplication Triples

Additively Secret Sharing ($\mathsf{Shr}(x)$). The additive secret sharing scheme separates an l-bit value x into two shares $x^{(1)}$ and $x^{(2)}$, in which $x^{(1)}$ is a uniformly distributed random number in the ring $\mathbb{Z}_{2^l}$, and $x^{(2)} = x - x^{(1)} \ mod \ 2^l$. These two shares are held by two parties P_1 and P_2, respectively. For ease of presentation, we omit the modular operation (*i.e.*, $mod\ 2^l$) in the following descriptions.

Secret Reconstruction ($\mathsf{Rec}(x^{(1)}, x^{(2)})$). Given two shares $x^{(1)}$ and $x^{(2)}$, a party can reconstruct the secret x by computing $x = x^{(1)} + x^{(2)}$.

Secure Addition Protocol on Shared Data. Given two values x and y shared between P_1 and P_2, we assume that P_1 holds shares $x^{(1)}$ and $y^{(1)}$, and P_2 holds shares $x^{(2)}$ and $y^{(2)}$. P_i can compute the sum of his shares by calculating $z^{(i)} = x^{(i)} + y^{(i)}$, $i = \{1, 2\}$. Finally, the addition of x and y can be obtained by computing $z^{(1)} + z^{(2)}$. Similarly, we can also derive $x - y$ by having P_i compute $z^{(i)} = x^{(i)} - y^{(i)}$.

Secure Multiplication Protocol on Shared Data. To enable multiplication of two values x and y shared between P_1 and P_2, we adopt the multiplication triple technique [1], [11]. In this technique, P_1 and P_2 are assumed to share three values/triples a, b, c, where $c = a \cdot b$ and a, b are uniformly randomized in the ring $\mathbb{Z}_{2^l}$. P_i computes $e^{(i)} = x^{(i)} - a^{(i)}$ and $f^{(i)} = y^{(i)} - b^{(i)}$. Both parties then jointly reconstruct e and f. After that, P_i sets $z^{(i)} = (i-1) \cdot e \cdot f + f \cdot a^{(i)} + e \cdot b^{(i)} + c^{(i)}$. The above operations on shared values can also be generalized to shared vectors. In Section 5.1.2, we will show how to perform secure multiplication protocol on shared vectors.

4.3 Partial Decryption-based Paillier Cryptosystem

To further mitigate the leaking risk of the system private key which is held by a single server, [18] adapted the conventional Paillier Cryptosystem by separating the private key into two different shares between two parties. This new cryptosystem, named Paillier Cryptosystem with Partial Decryption (PCPD), provides a higher security guarantee as even if one server is compromised, the system private key is still hidden. PCPD works as follows:

- **KeyGen**(1^κ): Given a security parameter κ and two κ-bit large prime numbers p, q, this algorithm outputs a public-private key pair (pk, sk). Specifically, it computes $N = p \cdot q$ and $\lambda = (p-1)(q-1)/2$, based on which, we generate a generator g of order N and define a function $L(x) = (x-1)/N$. Finally, we set the public key $pk = N$ and the private key $sk = \lambda$.
- **KeyS**(λ): This algorithm splits the private key $sk = \lambda$ into two parts, *i.e.*, $sk^{(i)} = \lambda_i (i = \{1, 2\})$, where $\lambda_1 + \lambda_2 \equiv 0 \ mod \ \lambda$ and $\lambda_1 + \lambda_2 \equiv 1 \ mod \ N$.
- **Enc**(m, pk): Given a message $m \in \mathbb{Z}_N$ and the public key $pk = N$, this algorithm chooses a random number $r \in \mathbb{Z}_N^*$ and outputs the ciphertext $[m]_{pk} = g^m \cdot r^N \ mod \ N^2 = (1 + mN) \cdot r^N \ mod \ N^2$.
- **Dec**($[m]_{pk}, \lambda$): Given a ciphertext $[m]_{pk}$ and the private key λ, this algorithm outputs the plaintext $m = L([m]_{pk}^{\lambda} \ mod \ N^2)\lambda^{-1} \ mod \ N$, where $[m]_{pk}^{\lambda} \ mod \ N^2 = (1 + mN\lambda)$.
- **PD1**($[m]_{pk}, \lambda_1$): This algorithm performs the first-step partial decryption with the partial private key λ_1. Given a ciphertext $[m]_{pk}$ and λ_1, it outputs $CT^{(1)} = [m]_{pk}^{\lambda_1} = r^{\lambda_1 N}(1 + mN\lambda_1) \ mod \ N^2$.
- **PD2**($[m]_{pk}, CT^{(1)}, \lambda_2$): This algorithm performs the second-step partial decryption with another partial private key λ_2. Given the partial decrypted ciphertext $CT^{(1)}$, it first executes $CT^{(2)} = [m]_{pk}^{\lambda_2} =$

$r^{\lambda_2 N}(1+mN\lambda_2)\ mod\ N^2$, and then computes $m = L(CT^{(1)} \cdot CT^{(2)}\ mod\ N^2)$.

Additive homomorphic property: Given $m_1, m_2 \in \mathbb{Z}_N$ under the same pk, we have $[m_1]_{pk} \cdot [m_2]_{pk} = (1 + (m_1 + m_2)N) \cdot (r_1 + r_2)^N\ mod\ N^2 = [m_1 + m_2]_{pk}$, and $([m]_{pk})^{N-1} = (1 + (N-1)mN) \cdot r^{(N-1)N}\ mod\ N^2 = [-m]_{pk}$.

The notations used in this paper are listed in Table 2.

TABLE 2
Notation Settings

Notations	Description
M	Number of sensing tasks
K	Number of mobile users
$x_{k,m}$	The sensory data of task τ_m submitted by user u_k
x_m	The estimated truth of task τ_m
ϵ	The predifined threshold
$\mathbf{x}$	A vector value
$d(\mathbf{x}_k, \mathbf{x})$	The square of Euclidean distance between $\mathbf{x}_k$ and $\mathbf{x}$
$\mathbf{x} \cdot \mathbf{y}$	The inner product of vectors $\mathbf{x}$ and $\mathbf{y}$
$x \cdot y$	The multiplication of scalars x and y
λ_1, λ_2	Partial private keys
pk, sk	Key-pair of the system
pk_r, sk_r	Key-pair of the data requester
$[m]_{pk}$	Ciphertext of message m encrypted by pk
$\lceil\ \rceil$	The ceiling function
$\lfloor\ \rfloor$	The floor function
$[x\|y]$	The packed value of x and y

5 Our Proposed Scheme

In this section, we present our schemes for secure TD for mobile crowdsensing. The first scheme BPTD is inspired by the salient features of additive secret sharing. It enables some arithmetic computations (*e.g.*, addition, subtraction, and multiplication) directly to be performed on shared data, but needs extra interactions with the mobile users for division and logarithm computations (reveals partial intermediates). For better practicability and security, we present our second design PETD, incorporating the power of additive secret sharing and PCPD to provide enhanced PPTD on hybrid data. PETD offers stronger privacy protection against user-server collusion attacks in a non-interactive way. Finally, we present an optimized PETD with reduced computation and communication cost at the servers.

5.1 BPTD

5.1.1 Design Principle

We observe that one of the core operations in TD is to compute accumulated values such as accumulated square and weighted sensory data. Additionally, the meta-operations involved in these computations are mostly additions/multiplications, which can be efficiently performed on additive shared data based on the aforementioned addition/multiplication protocols. According to the property of arithmetic sharing, the key idea of BPTD is to let the two servers first compute the shares of all accumulated values included in weight estimation, based on which, two shares of weight are to be derived. Secure truth estimation and convergence determination are then both conducted on shared values. In this case, the privacy of all data involved is well protected as the servers do not collude with each other.

However, a non-trivial challenge is that additive secret sharing does not support other operations such as logarithm and division computations which are also required in the TD procedure. Cai *et al.* [2] integrated GC to tackle division and approximately computed logarithm values. However, constructing such GCs is usually time-consuming and there is a need for an efficient approximation with high accuracy for the logarithm function. Reconstructing the intermediate results and then splitting is a viable solution but it confronts the risks of possible indirect privacy disclosure especially when some parties collude with each other.

To tackle the above challenge, we carefully design BPTD based on the existing arithmetic sharing protocols. Our insight is to employ different parties for data reconstruction so that each party only knows partial intermediate results and can derive nothing about the sensory data, weights, and truths by himself. The coordination of the two servers is, however, not sufficient or secure. For example, in weight estimation, we need to reconstruct two values $\sum_{k=1}^{K} d(\mathbf{x}_k, \mathbf{x})$ and $d(\mathbf{x}_k, \mathbf{x})$ for logarithm computations. One server reconstructing $d(\mathbf{x}_k, \mathbf{x}), k \in [1, K]$ would get the accumulated value $\sum_{k=1}^{K} d(\mathbf{x}_k, \mathbf{x})$ and deduce each w_k. To tackle this issue, we let the users join in the reconstruction process but only allow each user u_k to reconstruct a value $d(\mathbf{x}_k, \mathbf{x})$. In this case, there is only one server-user interaction in each iteration. Moreover, even though the user colludes with the server who holds $\sum_{k=1}^{K} d(\mathbf{x}_k, \mathbf{x})$, an inferred weight is merely known by the user himself. For secure truth estimation, since there is no logarithm operation included, we only need to reconstruct $\sum_{k=1}^{K} w_k$, which can be done by the server not recovering data in weight estimation. Such an idea complies with the requirements of minimizing the user-side interactions and information held by each party.

5.1.2 Detailed Design

BPTD consists of one-time system initialization, multi-time iterations, and final truth recovery at the data requester. Specifically, there are three iterative sub-procedures: secure weight estimation, secure truth estimation, and secure convergence determination. Fig. 1 depicts the workflow and inter-entity interaction process of BPTD. Note that our secret sharing-based arithmetic operations are all performed in the integer ring, and non-integer values can be scaled up to integers by multiplying by a parameter P (a magnitude of 10). Accordingly, the original value can be recovered by dividing by P. In Section 7.2, we will show that the accuracy of our result is not compromised if a proper P is chosen.

1) System Initialization. Instead of encrypting the data locally, each user $u_k \in \mathcal{U}$ splits his data vector $\mathbf{x}_k$ into two shares, denoted as $\mathbf{x}_k^{(1)}$ and $\mathbf{x}_k^{(2)}$, and sends them to C_1 and C_2, respectively. The user-side computation cost is M simple modulo (*i.e.*, *mod* 2^l) operations for M sensing objects. The user-server communication cost is the bit length of $2M$ data shares. Besides, the system sets a threshold ϵ to control the convergence of TD. ϵ is also split into $\epsilon^{(1)}$ and $\epsilon^{(2)}$ and each server holds one share. Similarly, the estimated ground truth $\mathbf{x}$ is first initialized with random shares, represented by $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, between the servers.

To enable multiplication operations on shared vectors, the system sets shared vectors $\mathbf{a}$, $\mathbf{b}$, c and $\mathbf{a}'$, $\mathbf{b}'$, c', which is similar to that on shared data as introduced in Section 4.2. The elements of these vectors are uniformly randomized in

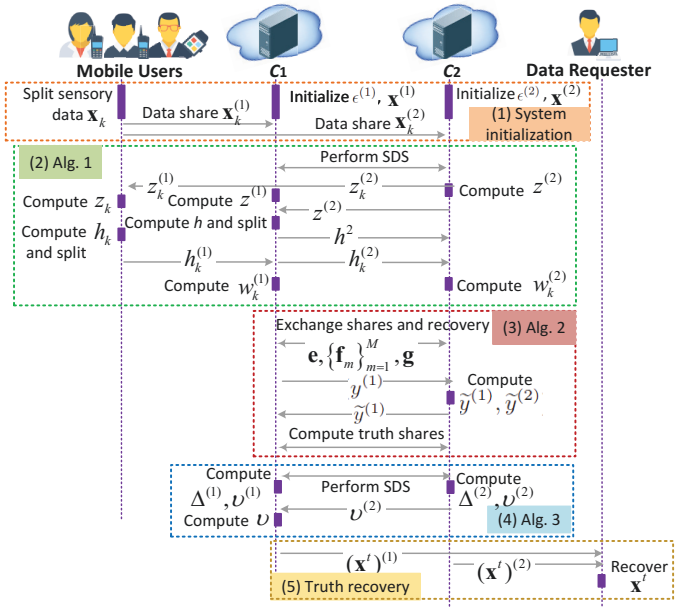


Fig. 1. The workflow of BPTD.

$\mathbb{Z}_{2^l}$, and $c = \mathbf{a} \cdot \mathbf{b}\ mod\ 2^l$, $c' = \mathbf{a}' \cdot \mathbf{b}'\ mod\ 2^l$. Finally, these parameters are split and shared between the servers. For each server, the total storage cost is the bit length of $(KM+3M+2K+3)$ data shares in the initialization phase.

2) Secure Weight Estimation. Since computing $d(\mathbf{x}_k, \mathbf{x})$ is a crucial operation in weight estimation, we first introduce a secure distance square computation method on shared data (SDS), which serves as a main component of our secure weight estimation later. Specifically, given the aforementioned shared data, SDS works as follows. For each mobile user u_k, server C_i ($i = \{1, 2\}$) first locally computes $\mathbf{u}_k^{(i)} = \mathbf{x}_k^{(i)} - \mathbf{x}^{(i)}$, $\mathbf{e}_k^{(i)} = \mathbf{u}_k^{(i)} - \mathbf{a}^{(i)}$, and $\mathbf{f}_k^{(i)} = \mathbf{u}_k^{(i)} - \mathbf{b}^{(i)}$, in which $\mathbf{e}_k^{(i)}$ and $\mathbf{f}_k^{(i)}$ are sent to another server for reconstruction. After recovering $\mathbf{e}_k$ and $\mathbf{f}_k$, C_i computes $z_k^{(i)} = (i-1)\mathbf{e}_k \cdot \mathbf{f}_k + \mathbf{a}^{(i)} \cdot \mathbf{f}_k + \mathbf{b}^{(i)} \cdot \mathbf{e}_k + c^{(i)}$, which outputs the shares of product of two vectors. As a result, C_1 obtains $z_k^{(1)} = d(\mathbf{x}_k, \mathbf{x})^{(1)}$ while C_2 obtains $z_k^{(2)} = d(\mathbf{x}_k, \mathbf{x})^{(2)}$.

Based on the shared distance square, we further present secure weight estimation on shared data, which finally outputs the corresponding weight shares. As illustrated in Algorithm 1, for each user $u_k \in \mathcal{U}$, after performing SDS with C_2, the servers send $z_k^{(1)}$ and $z_k^{(2)}$ to u_k for reconstruction. Additionally, both servers compute the sum of its held distance square shares among all users (Line 6), after which, C_2 obtains $z^{(2)}$ and sends it to C_1 for reconstruction. On the user side, u_k recoveries z_k and computes $h_k = \log z_k$. Since C_1 and u_k only know the partial information, there is no direct privacy disclosure of weights. After deducing the logarithm values, h and h_k are then split into data shares $h^{(i)}$ and $h_k^{(i)}$, $i \in \{1, 2\}$. u_k sends the corresponding shares to the servers for weight share computation.

In Algorithm 1, $5KM+2K+2$ and $5KM+2K$ modulo operations are needed at C_1 and C_2, respectively. Each mobile user performs 2 module operations for data reconstruction and split. For user-server communication cost, 4 data shares are transmitted between each user and the servers. There are $(4KM + 2)$ data shares transmitted between the two servers. For storage cost in each algorithm, besides the algorithm input, we assume that each party only stores the final results. As the storage of inputs of Algorithm 1 has been considered in the initialization phase, C_1 and C_2 only need to store $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$. The corresponding storage cost is the bit length of K data shares, respectively.

Algorithm 1: Secure Weight Estimation on Shared Data

Input: C_1 has $\{\mathbf{x}_k^{(1)}\}_{k=1}^K, \mathbf{x}^{(1)}, \mathbf{a}^{(1)}, \mathbf{b}^{(1)}, c^{(1)}$; C_2 has $\{\mathbf{x}_k^{(2)}\}_{k=1}^K, \mathbf{x}^{(2)}, \mathbf{a}^{(2)}, \mathbf{b}^{(2)}, c^{(2)}$.
Output: C_1 obtains $\mathbf{w}^{(1)}$, C_2 obtains $\mathbf{w}^{(2)}$.

1 // C_1 and C_2;
2 **for** $k = 1$ *to* K **do**
3 Perform SDS;
4 $z_k^{(1)} \leftarrow d(\mathbf{x}_k, \mathbf{x})^{(1)}$, $z_k^{(2)} \leftarrow d(\mathbf{x}_k, \mathbf{x})^{(2)}$;
5 C_1 and C_2 send $z_k^{(1)}$ and $z_k^{(2)}$ to u_k, respecively;
6 C_i computes $z^{(i)} = \sum_{k=1}^K z_k^{(i)}$ for $i \in \{1, 2\}$;
7 C_2 sends $z^{(2)}$ to C_1;
8 // C_1;
9 $z \leftarrow \mathsf{Rec}(z^{(1)}, z^{(2)})$;
10 Compute $h = \log z$, $(h^{(1)}, h^{(2)}) \leftarrow \mathsf{Shr}(h)$;
11 Send $h^{(2)}$ to C_2;
12 // Mobile user;
13 **for** $k = 1$ *to* K **do**
14 u_k computes $z_k \leftarrow \mathsf{Rec}(z_k^{(1)}, z_k^{(2)})$;
15 Compute $h_k = \log z_k$, $(h_k^{(1)}, h_k^{(2)}) \leftarrow \mathsf{Shr}(h_k)$;
16 Send $h_k^{(1)}$ and $h_k^{(2)}$ to C_1 and C_2, respectively;
17 // C_1 and C_2;
18 **for** $k = 1$ *to* K **do**
19 Compute $w_k^{(i)} = h^{(i)} - h_k^{(i)}$ for $i \in \{1, 2\}$;

3) Secure Truth Estimation. After deriving the weight shares $\mathbf{w}^{(1)}$ and $\mathbf{w}^{(2)}$, we can conduct secure truth estimation on shared data, as shown in Algorithm 2. First, to derive $\mathbf{w} \cdot \mathbf{x}_{*,m}$ based on the shared data, C_1 computes $\mathbf{e}^{(1)}$ (relevant to $\mathbf{w}^{(1)}$) and $\mathbf{f}_m^{(1)}$ (relevant to $\mathbf{x}_{*,m}^{(1)}$) while C_2 computes $\mathbf{e}^{(2)}$ and $\mathbf{f}_m^{(2)}$. Similarly, to derive $\mathbf{w} \cdot \mathbf{s}$, C_1 and C_2 also compute $\mathbf{g}^{(1)}$ and $\mathbf{g}^{(2)}$, respectively. After recovering the above values, the servers then compute the shares of the denominator of Eq. (2) (Line 15). To deal with the division computation on shared data, our basic idea is to convert division to multiplication which can be performed efficiently based on the secure multiplication protocol in Section 4.2. Specifically, we let C_2 reconstruct y whose inverse is then rounded by a factor P'. After getting $\tilde{y}$, C_2 further splits it into two shares and sends $\tilde{y}^{(1)}$ to C_1. Finally, both servers obtain $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ by performing secure multiplication protocol on data shares of y_m and $\tilde{y}$.

In Algorithm 2, $2KM + 4K + 3M + 4$ and $2KM + 4K + 3M + 6$ modulo operations are incurred at C_1 and C_2, respectively. Meanwhile, $2KM + 4K + 2M + 4$ data shares are transmitted between the two servers. For storage cost, C_1 stores $\mathbf{s}^{(1)}$ as input (can be also included in the initialization phase, then it can be neglected in this phase) and $\mathbf{x}^{(1)}$ as output. Similarly, C_2 stores the other two shares. Accordingly, the storage cost at both servers is the bit length of $K + M$ data shares.

Algorithm 2: Secure Truth Estimation on Shared Data

Input: C_1 has $\mathbf{w}^{(1)}, \{\mathbf{x}^{(1)}_{*,m}\}_{m=1}^{M}, \mathbf{s}^{(1)}, \mathbf{a}'^{(1)}, \mathbf{b}'^{(1)}, c'^{(1)}$; C_2 has $\mathbf{w}^{(2)}, \{\mathbf{x}^{(2)}_{*,m}\}_{m=1}^{M}, \mathbf{s}^{(2)}, \mathbf{a}'^{(2)}, \mathbf{b}'^{(2)}, c'^{(2)}$.
Output: C_1 obtains $\mathbf{x}^{(1)}$, C_2 obtains $\mathbf{x}^{(2)}$.

// C_1;
$\mathbf{e}^{(1)} \leftarrow \mathbf{w}^{(1)} - \mathbf{a}'^{(1)}$;
for $m = 1$ *to* M **do**
  $\mathbf{f}^{(1)}_m \leftarrow \mathbf{x}^{(1)}_{*,m} - \mathbf{b}'^{(1)}$;
$\mathbf{g}^{(1)} \leftarrow \mathbf{s}^{(1)} - \mathbf{b}'^{(1)}$;
Send $\mathbf{e}^{(1)}, \{\mathbf{f}^{(1)}_m\}_{m=1}^{M}, \mathbf{g}^{(1)}$ to C_2;
// C_2;
$\mathbf{e}^{(2)} \leftarrow \mathbf{w}^{(2)} - \mathbf{a}'^{(2)}$;
for $m = 1$ *to* M **do**
  $\mathbf{f}^{(2)}_m \leftarrow \mathbf{x}^{(2)}_{*,m} - \mathbf{b}'^{(2)}$;
$\mathbf{g}^{(2)} \leftarrow \mathbf{s}^{(2)} - \mathbf{b}'^{(2)}$;
Send $\mathbf{e}^{(2)}, \{\mathbf{f}^{(2)}_m\}_{m=1}^{M}, \mathbf{g}^{(2)}$ to C_1;
// C_1 and C_2;
Reconstruct $\mathbf{e}$, $\{\mathbf{f}_m\}_{m=1}^{M}$ and $\mathbf{g}$;
C_i computes $y^{(i)} = (i-1)\mathbf{e}\cdot\mathbf{g} + \mathbf{g}\cdot\mathbf{a}'^{(i)} + \mathbf{b}'^{(i)}\cdot\mathbf{e} + c'^{(i)}$;
C_1 sends $y^{(1)}$ to C_2;
// C_2;
$y \leftarrow \mathsf{Rec}(y^{(1)}, y^{(2)})$;
$\widetilde{y} \leftarrow \lfloor P'/y \rfloor$, $(\widetilde{y}^{(1)}, \widetilde{y}^{(2)}) \leftarrow \mathsf{Shr}(\widetilde{y})$;
Send $\widetilde{y}^{(1)}$ to C_1;
// C_1 and C_2;
for $m = 1$ *to* M **do**
  $y^{(i)}_m \leftarrow (i-1)\mathbf{e}\cdot\mathbf{f}_m + \mathbf{f}_m\cdot\mathbf{a}'^{(i)} + \mathbf{b}'^{(i)}\cdot\mathbf{e} + c'^{(i)}$ for $i \in \{1, 2\}$;
  Compute $x^{(i)}_m$ based on secure multiplication protocol on two shared data y_m and $\widetilde{y}$;

4) Secure Convergence Determination. After generating a new estimated ground truth, we need to evaluate whether the convergence criterion is satisfied. Meanwhile, no other information should be exposed to both servers except for the evaluation results. Algorithm 3 presents a secure convergence determination protocol on shared data, which takes the shares of two estimated truth vectors $\mathbf{x}^t, \mathbf{x}^{t-1}$ and the predefined threshold ϵ as inputs and outputs a convergence label ν. Finally, although C_1 learns ν, it cannot derive any information about x_m, x'_m, Δ, and ϵ without other shares.

Algorithm 3: Secure Convergence Determination on Shared Data

Input: C_1 has $(\mathbf{x}^t)^{(1)}, (\mathbf{x}^{t-1})^{(1)}, \epsilon^{(1)}$; C_2 has $(\mathbf{x}^t)^{(2)}, (\mathbf{x}^{t-1})^{(2)}, \epsilon^{(2)}$.
Output: Convergence label ν.

// C_1 and C_2;
Perform SDS protocol;
$\Delta^{(1)} \leftarrow d(\mathbf{x}^t, \mathbf{x}^{t-1})^{(1)}$, $\Delta^{(2)} \leftarrow d(\mathbf{x}^t, \mathbf{x}^{t-1})^{(2)}$;
// C_1;
$\nu^{(1)} \leftarrow \Delta^{(1)} - \epsilon^{(1)}$;
// C_2;
$\nu^{(2)} \leftarrow \Delta^{(2)} - \epsilon^{(2)}$, send $\nu^{(2)}$ to C_1;
// C_1;
$\nu \leftarrow \mathsf{Rec}(\nu^{(1)}, \nu^{(2)})$;

In Algorithm 3, C_1 and C_2 both require $5M + 3$ modulo operations. $4M + 1$ data shares are transmitted between the two servers. Finally, C_1 stores the convergence label ν.

5) Truth Recovery. Once convergence is detected in the t-th iteration, C_1 and C_2 send $(\mathbf{x}^t)^{(1)}$ and $(\mathbf{x}^t)^{(2)}$ to the data requester, respectively, in which $2M$ data shares are transmitted between the requester and the servers. Next, the data requester is able to reconstruct the final estimated truth vector $\mathbf{x}^t$ with Rec(), which needs M modulo operations. The associated storage cost is the bit length of M data.

Cost analysis of BPTD. Let *mod* denote the modulo operation and l_1 be the bit length of a data share (or a data), the cost of BPTD with one iteration is summarized as follows. For computation overhead, $(M + 2)$*mod* cost is incurred on each user side. As discussed in the above algorithms, the total computation cost of C_1 and C_2 is both $(7KM + 6K + 8M + 9)$*mod*. It takes M*mod* cost for requester-side data reconstruction. On the other hand, the user-server communication cost is $(2M + 4)l_1$. In contrast, $(6KM+4K+6M+7)l_1$ communication is needed between the two servers while $2Ml_1$ communication is needed between the servers and the requester. For data storage, the corresponding cost at C_1 and C_2 is $(KM+4M+4K+4)l_1$ and $(KM+4M+4K+3)l_1$, respectively. The requester-side storage cost is only Ml_1.

5.2 PETD

Despite the efficiency advantages, we observe that BPTD requires C_1 and each user u_k to reconstruct $\sum_{k=1}^{K} d(\mathbf{x}_k, \mathbf{x})$ and $d(\mathbf{x}_k, \mathbf{x})$ in secure weight estimation. Additionally, C_2 is responsible for recovering $\sum_{k=1}^{K} w_k$ in each secure truth update. In practice, this is not desirable due to possible indirect privacy disclosure and user-side online requirement.

For the above concerns, we continue our second attempt PETD by resorting to the state-of-the-art partial decryption-based Paillier Cryptosystem PCPD, and combining it with additive secret sharing to design protocols for secure TD on hybrid data. Different from exiting homomorphic encryption-based [9] secure computation protocols and PPTD scheme [47], PETD splits both user data and the system private key into two shares between the servers, which reduces the risk of data and key leakage. Moreover, the whole TD is securely conducted by the servers without any user's participation and leakage of the intermediates.

5.2.1 Construction Details

As in BPTD, we require each user to share one part of his sensory data with each server. Instead of using data shares, the estimated truths and threshold are initiated with encrypted random values in PETD. The two servers then collaboratively compute the encrypted distance square between each user's sensory data and the estimated truths via PCPD. Based on this, three secure protocols are all conducted on hybrid data without revealing any information about the user's data, weights, intermediates, and the final results. Finally, only the requester can obtain the final truths. Note that, the rounding factor P is also used to deal with the floating-point number, which is omitted in the following algorithms for ease of presentation. The detailed procedure of PETD is shown in Fig. 2.

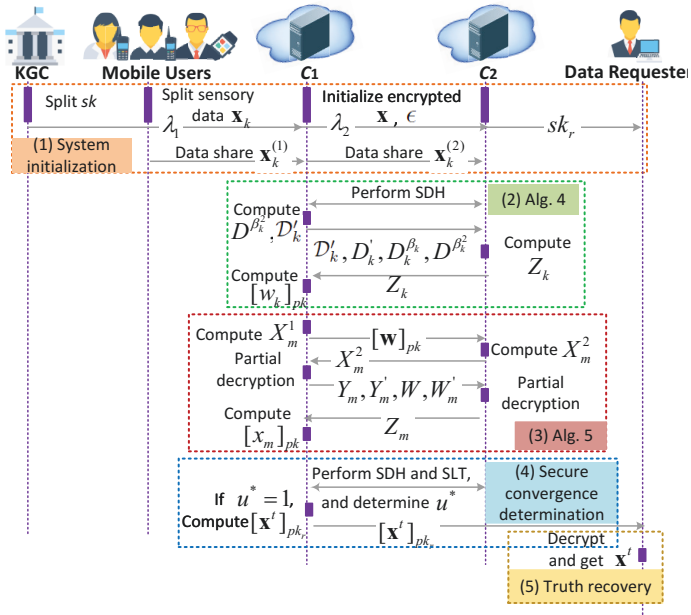


Fig. 2. The workflow of PETD.

1) System Initialization. Given a security parameter κ, two key pairs (pk, sk) and (pk_r, sk_r) are first generated by the KGC[2] via **KeyGen**(). Specifically, pk and pk_r are used to encrypt the sensory data and the estimated truth, respectively. After key generation, the KGC splits sk into λ_1 and λ_2 via **KeyS**(). sk_r is sent to the data requester while λ_1 and λ_2 are sent to C_1 and C_2, respectively. In PETD, $\mathbf{x}$ is randomly initialized and then encrypted with pk by C_1 (each encryption requires one modular multiplication and two modular exponentiation operations). Meanwhile, ϵ is also encrypted with pk. Then, each user u_k uploads $x_k^{(1)}$ and $x_k^{(2)}$ to C_1 and C_2, respectively. As in BPTD, the user-side computation cost is $Mmod$ and the user-server communication cost is $2Ml_1$ in the initialization phase. The difference lies in that C_1 needs $M+1$ modular multiplication and $2(M+1)$ modular exponentiation operations to encrypt $\mathbf{x}$ and ϵ. For storage cost, C_1 needs to store $\{\mathbf{x}_k^{(1)}\}_{k=1}^K$, $[\mathbf{x}]_{pk}$, $[\epsilon]_{pk}$, λ_1, *i.e.*, KM data shares, $M+1$ ciphertexts, and a partial decryption key λ_1. In contrast, C_2 only stores KM data shares $\{\mathbf{x}_k^{(2)}\}_{k=1}^K$ and a partial decryption key λ_2.

2) Secure Weight Estimation. Before illustration, we first present its core sub-protocol, named **S**ecure **D**istance square computation on **H**ybrid data SDH (see Fig. 3). Given C_1 holding $\{\mathbf{x}_k^{(1)}\}_{k=1}^K$, $[\mathbf{x}]_{pk}$, λ_1 and C_2 holding $\{\mathbf{x}_k^{(2)}\}_{k=1}^K, \lambda_2$, SDH enables C_1 to securely get $[d(\mathbf{x}_k, \mathbf{x})]_{pk}$. Let mul and exp denote the modular multiplication and modular exponentiation, respectively. As the underlying sub-protocol, SDH requires C_1 to perform $3M$ encryptions (each costs 1*mul*+2*exp*) and M first-step decryptions (each costs 1*exp*) for each user[3]. For C_2, only $2M$ encryptions and M second-step decryptions (each costs 1*mul*+1*exp*) are needed.

Algorithm 4: Secure Weight Estimation on Hybrid Data

Input: C_1 has $\{\mathbf{x}_k^{(1)}\}_{k=1}^K, [\mathbf{x}]_{pk}, \lambda_1$; C_2 has $\{\mathbf{x}_k^{(2)}\}_{k=1}^K, \lambda_2$.
Output: C_1 obtains $[\mathbf{w}]_{pk}$.

1 // C_1 and C_2;
2 **for** $k = 1$ *to* K **do**
3 Compute $D_k \leftarrow [d(\mathbf{x}_k, \mathbf{x})]_{pk}$;
4 Choose a random number β_k, and compute $D_k^{\beta_k}$, $D'_k \leftarrow PD1(D_k^{\beta_k}, \lambda_1)$;
5 $D \leftarrow \Pi_{k=1}^K D_k$;
6 **for** $k = 1$ *to* K **do**
7 Compute $D^{\beta_k^2}, \mathcal{D}'_k \leftarrow PD1(D^{\beta_k^2}, \lambda_1)$;
8 Send $\mathcal{D}'_k, D'_k, D_k^{\beta_k}, D^{\beta_k^2}$ to C_2;
9 // C_2;
10 **for** $k = 1$ *to* K **do**
11 $D''_k \leftarrow PD2(D_k^{\beta_k}, D'_k, \lambda_2)$, $\mathcal{D}''_k \leftarrow PD2(D^{\beta_k^2}, \mathcal{D}'_k, \lambda_2)$;
12 Compute $\log(\mathcal{D}''_k/D''_k)$, $Z_k \leftarrow [\log(\mathcal{D}''_k/D''_k)]_{pk}$;
13 Send Z_k to C_1;
14 // C_1;
15 **for** $k = 1$ *to* K **do**
16 $[w_k]_{pk} \leftarrow Z_k \cdot [\log \beta_k]_{pk}^{N-1}$;

After that, both servers continue secure weight estimation on hybrid data. As shown in Algorithm 4, for each user u_k, C_1 first derives D_k via SDH under the assistance of C_2. To compute the weight without revealing any information about D_k, C_1 then chooses a random number $\beta_k \in \mathbb{Z}_N^*$ ($\beta_k^2 \in \mathbb{Z}_N^*$ is also satisfied) and computes $D_k^{\beta_k}$, $D^{\beta_k^2}$, where D is the encrypted sum of distance square of all users (Line 5). After partial decryption (*i.e.*, $2Kexp$ computation cost), $D_k^{\beta_k}$ and $D^{\beta_k^2}$ are sent to C_2 with D'_k and $\mathcal{D}'_k$. Next, C_2 performs the partial decryption ($2K(exp+mul)$ computation cost) and derives D''_k and $\mathcal{D}''_k$. It then can obtain the masked weight by computing $\log(\mathcal{D}''_k/D''_k)$. The corresponding ciphertext Z_k (K encryptions with $K(2exp+mul)$ computation cost) is sent to C_1. Since only C_1 knows β_k, it can get the encrypted weight of u_k by multiplying by $[\log \beta_k]_{pk}^{N-1}$ (K encryptions with $K(2exp + mul)$ computation cost). The correctness of weight estimation is shown in Eq. (4), in which any floating point number x is rounded as $\lceil x \cdot P \rceil$. In Eq. (4), it is possible that $\lceil P \cdot \log \beta_k + P \cdot \log(\Sigma_{k=1}^K d(\mathbf{x}_k, \mathbf{x})/d(\mathbf{x}_k, \mathbf{x}))\rceil = \lceil P \cdot \log \beta_k \rceil + \lceil P \cdot \log(\Sigma_{k=1}^K d(\mathbf{x}_k, \mathbf{x})/d(\mathbf{x}_k, \mathbf{x}))\rceil - 1$. In this case, the result is $[\lceil P \cdot w_k \rceil - 1]_{pk}$, a large P will not compromise the accuracy of the recovered weight.

In Algorithm 4, the computation cost of C_1 and C_2 is $(3KM + K)mul + (7KM + 4K)exp$ and $(3KM + 3K)mul + (5KM + 4K)exp$. Let l_2 be the bit length of a ciphertext, the bit length of λ_1 and λ_2 is also l_2. The communication cost between C_1 and C_2 is $(4KM + 5K)l_2$, in which $(4KM)l_2$ data is transmitted between the two servers in SDH. For data storage, C_1 needs Kl_2 cost to store the final weight ciphertexts $[\mathbf{w}]_{pk}$.

2. It is possible to remove the KGC and let C_2 and the requester generate (pk, sk) and (pk_r, sk_r), respectively. C_2 then splits sk into λ_1 and λ_2. λ_1 is sent to C_1 while λ_2 is held by C_2. Such a solution needs entity-side extra computation and communication cost, and loses the advantage of key split in key leakage mitigation as we mentioned in Section 4.3. Hence, we still introduce the KGC for initialization.

3. As in [18], we ignore other operations and focus on the encryption and (partial) decryption cost in computational analysis.

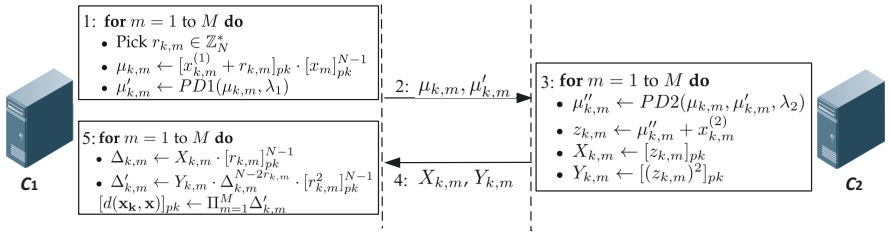


Fig. 3. The illustration of SDH.

$$Z_k \cdot [\log \beta_k]_{pk}^{N-1} = [\log(\beta_k \frac{\Sigma_{k=1}^K d(\mathbf{x}_k, \mathbf{x})}{d(\mathbf{x}_k, \mathbf{x})})]_{pk} \cdot [\log \beta_k]_{pk}^{N-1}$$
$$\overset{Rounded}{\Longrightarrow} [\lceil P \cdot \log(\beta_k \frac{\Sigma_{k=1}^K d(\mathbf{x}_k, \mathbf{x})}{d(\mathbf{x}_k, \mathbf{x})})\rceil]_{pk} \cdot [\lceil P \cdot \log \beta_k \rceil]_{pk}^{N-1}$$
$$= [\lceil P \cdot \log \beta_k + P \cdot \log \frac{\Sigma_{k=1}^K d(\mathbf{x}_k, \mathbf{x})}{d(\mathbf{x}_k, \mathbf{x})}\rceil]_{pk} \cdot [\lceil P \cdot \log \beta_k \rceil]_{pk}^{N-1}$$
$$= [\lceil P \cdot \log \beta_k \rceil + \lceil P \cdot \log \frac{\Sigma_{k=1}^K d(\mathbf{x}_k, \mathbf{x})}{d(\mathbf{x}_k, \mathbf{x})}\rceil]_{pk} \cdot [\lceil P \cdot \log \beta_k \rceil]_{pk}^{N-1}$$
$$= [\lceil P \cdot \log \beta_k \rceil + \lceil P \cdot \log \frac{\Sigma_{k=1}^K d(\mathbf{x}_k, \mathbf{x})}{d(\mathbf{x}_k, \mathbf{x})}\rceil - \lceil P \cdot \log \beta_k \rceil]_{pk}$$
$$= [\lceil P \cdot \log \frac{\Sigma_{k=1}^K d(\mathbf{x}_k, \mathbf{x})}{d(\mathbf{x}_k, \mathbf{x})}\rceil]_{pk} = [\lceil P \cdot w_k \rceil]_{pk}. \quad (4)$$

3) Secure Truth Estimation. With $[\mathbf{w}]_{pk}$, secure truth estimation is performed via Algorithm 5. For each task τ_m, C_1 and C_2 compute X_m^1 and X_m^2, respectively. Next, C_2 sends X_m^2 to C_1 who can obtain the encrypted numerator of Eq. (2) by computing $X_m^1 \cdot X_m^2$. Meanwhile, the encrypted denominator of Eq. (2) is derived by multiplication of all the encrypted weights. Next, a random number r_m is chosen to mask the numerator, denominator, and quotient so that C_2 cannot infer any information from Y''_m, W''_m, and Y''_m/W''_m. The correctness of truth estimation is shown in Eq. (5) (P and the modular operation are included).

$$Z_m^{r_m^{-1}} \overset{Rounded}{\Longrightarrow} ([\lceil P \cdot r_m \frac{\Sigma_{k=1}^K x_{k,m} w_k}{\Sigma_{k=1}^K w_k}\rceil]_{pk})^{r_m^{-1} \ mod \ N}$$
$$= [r_m^{-1} \cdot r_m \cdot \lceil P \cdot \frac{\Sigma_{k=1}^K x_{k,m} w_k}{\Sigma_{k=1}^K w_k}\rceil]_{pk}$$
$$= [\lceil P \cdot \frac{\Sigma_{k=1}^K x_{k,m} w_k}{\Sigma_{k=1}^K w_k}\rceil]_{pk} = [\lceil P \cdot x_m \rceil]_{pk}. \quad (5)$$

In Algorithm 5, C_1 performs $2M$ first-step partial decryptions with $2Mexp$ cost. In contrast, C_2 requires $3Mmul + 4Mexp$ cost for $2M$ second-step partial decryptions and M encrytpions. The server-server communication cost is $(K + 6M)l_2$. Finally, Ml_2 storage cost is incurred at C_1 to store the truths.

4) Secure Convergence Determination. W.l.o.g., given two encrypted estimated truths $[\mathbf{x}^t]_{pk}$, $[\mathbf{x}^{t-1}]_{pk}$ in two consecutive iterations, we can evaluate if Eq. (3) is satisfied via a secure less than (SLT) protocol [18] (In SLT, C_1 performs two encryption and one first-step decryption operations while C_2 performs one encryption and one second-step decryption operations). The objective of SLT is to get the encrypted data $[u^*]_{pk}$ which indicates the relationship between $d(\mathbf{x}^t, \mathbf{x}^{t-1})$ and ϵ_m. If $u^* = 0$, it implies that $d(\mathbf{x}^t, \mathbf{x}^{t-1}) \geq \epsilon_m$. If $u^* = 1$, $d(\mathbf{x}^t, \mathbf{x}^{t-1}) < \epsilon_m$.

Algorithm 5: Secure Truth Estimation on Hybrid Data

Input: C_1 has $[\mathbf{w}]_{pk}, \mathbf{x}_{*,m}^{(1)}, \lambda_1$; C_2 has $\mathbf{x}_{*,m}^{(2)}, \lambda_2$.
Output: C_1 obtains $[\mathbf{x}]_{pk}$.

// C_1;
for $m = 1$ *to* M **do**
  $X_m^1 \leftarrow \Pi_{k=1}^K [w_k]_{pk}^{x_{k,m}^{(1)}}$;
Send $[\mathbf{w}]_{pk}$ to C_2;
// C_2;
for $m = 1$ *to* M **do**
  $X_m^2 \leftarrow \Pi_{k=1}^K [w_k]_{pk}^{x_{k,m}^{(2)}}$;
Send X_m^2 to C_1;
// C_1;
$W \leftarrow \Pi_{k=1}^K [w_k]_{pk}$;
for $m = 1$ *to* M **do**
  $X_m \leftarrow X_m^1 \cdot X_m^2$;
  Choose a random number r_m and compute $Y_m \leftarrow X_m^{r_m^2}, W_m \leftarrow W^{r_m}$;
  $Y'_m \leftarrow PD1(Y_m, \lambda_1), W'_m \leftarrow PD1(W_m, \lambda_1)$;
  Send Y_m, Y'_m, W, W'_m to C_2;
// C_2;
for $m = 1$ *to* M **do**
  $Y''_m \leftarrow PD2(Y_m, Y'_m, \lambda_2), W''_m \leftarrow PD2(W_m, W'_m, \lambda_2)$;
  Compute Y''_m/W''_m, $Z_m \leftarrow [Y''_m/W''_m]_{pk}$;
  Send Z_m to C_1;
//C_1;
for $m = 1$ *to* M **do**
  $[x_m]_{pk} \leftarrow Z_m^{r_m^{-1}}$;

Specifically, C_1 first takes $[\mathbf{x}^t]_{pk}$, $[\mathbf{x}^{t-1}]_{pk}$ as inputs and jointly computes $[d(\mathbf{x}^t, \mathbf{x}^{t-1})]_{pk}$ with c_2 via a secure distance computation protocol similar to SDH (Different from SDH, here C_1 only needs $2M$ encryptions to encrypt a random number r_m, r_m^2 and performs M first-step partial decryptions. C_2 performs M second-step partial decryptions and M encryptions). Both servers then jointly perform SLT which outputs $[u^*]_{pk}$. Through two partial decryptions, we can obtain u^* and further evaluate the convergence. If $u^* = 1$, the convergence is considered satisfied and C_1 takes $[\mathbf{x}^t]_{pk}$ as the final result.

In secure convergence determination, C_1 performs $(2M + 2)mul + (5M + 5)exp$ operations while C_2 needs $(2M + 2)mul + (3M + 3)exp$ operations. The communication cost between the servers is $(4KM + 2)l_2$. Moreover, C_1 needs to store u^*, leading to l_1 storage cost.

5) Truth Recovery. C_1 first transforms $[\mathbf{x}^t]_{pk}$ into $[\mathbf{x}^t]_{pk_r}$, such that only the data requester can decrypt it with his private key sk_r. Specifically, C_1 chooses a random number $r_m \in \mathbb{Z}_N^*$ and computes $[r_m]_{pk}$ and $[r_m]_{pk_r}$, respectively. C_1 derives $[x_m^t + r]_{pk} = [x_m]_{pk} \cdot [r_m]_{pk}, m \in [1, M]$ and partially decrypts it with λ_1. C_2 will re-encrypt $x_m^t + r_m$ with pk_r after further decryption. Holding $[x_m^t + r_m]_{pk_r}$, C_1 is able to derive $[\mathbf{x}^t]_{pk_r}$ by multiplying by $[r_m]_{pk_r}^{N-1}$. Finally, $[\mathbf{x}^t]_{pk_r}$ is sent to the data requester for decryption.

The above truth recovery takes C_1 and C_2 $2Mmul + 5Mexp$ and $2Mmul + 3Mexp$ computation cost, respectively. The server-server and server-requester communication cost is $2Ml_2$ and Ml_2, respectively. Finally, the requester needs $M(mul + 2exp)$ cost to decrypt the encrypted truths.

Cost analysis of PETD. Based on the cost analyzed in the above algorithms, we can summarize the computation, communication, and storage cost of PETD with one iteration. Theoretically, each user needs $Mmod$ cost for data split and the requester needs $M(mul + 2exp)$ cost for the final decryption. The total computation cost of C_1 and C_2 is $(3KM + K + 5M + 3)mul + (7KM + 4K + 14M + 7)exp$ and $(3KM + 3K + 7M + 2)mul + (5KM + 4K + 10M + 3)exp$, respectively. On the other hand, the server-server communication cost is $(8KM + 6K + 8M + 2)l_2$ in total. In contrast, the server-user and server-requester communication cost is $2Ml_1$ and Ml_2, respectively. For data storage, C_1 and C_2 need $(KM + 1)l_1 + (2M + K + 2)l_2$ and $KMl_1 + l_2$ cost, respectively. The storage cost of the requester is Ml_1.

5.2.2 Optimization

We observe that random numbers are necessarily required to protect intermediates from C_2 in the aforementioned subprotocols. It, however, is time-consuming to encrypt and decrypt each masked data (*e.g.*, encrypt $x_{k,m}^{(1)} + r_{k,m}$), especially for large-scale users and tasks. To tackle this deficiency, we seamlessly integrate the data packing technology [17] into our proposed protocol. Instead of encrypting each plaintext, the basic idea of data packing is that one party (*e.g.*, C_1) first packs a certain number[4] (say η) of plaintexts into a value, and then encrypts it and sends the ciphertext to another party (*e.g.*, C_2) who can decrypt it and recover each plaintext after unpacking. Suppose that there are KM data to be packed, the server-side encryption and decryption cost will be reduced to $\lceil KM/\eta \rceil$ after data packing.

W.l.o.g., we assume that there are KM σ-bit integers $v_{k,m}$ to be packed into one value, where $k \in [1, K]$ and $m \in [1, M]$. The packed value can be computed as follows.

$$\begin{aligned} v_{pack} &= [v_{1,1}|v_{1,2}|\cdots|v_{1,M}|\cdots|v_{K,1}|v_{K,2}|\cdots|v_{K,M}] \\ &= \sum_{k=1}^{K}\sum_{m=1}^{M} v_{k,m} 2^{\varrho}, \end{aligned} \tag{6}$$

where $\varrho = \sigma((K-k+1)M - m)$, $v_{1,1}$ and $v_{K,M}$ are the highest and lowest significant bits of v_{pack}.

In SDH, we assume that $\eta = KM$. $x_{k,m}^{(1)} + r_{k,m}$ is chosen to be packed into $x_{pack}^{(1)} = \sum_{k=1}^{K}\sum_{m=1}^{M}(x_{k,m}^{(1)} + r_{k,m})2^{\varrho}$. Here σ included in ϱ is the bit-length of $x_{k,m}^{(1)} + r_{k,m}$, where $r_{k,m}$ is a $(l + \delta)$-bit integer, l is the bit-length of $x_{k,m}^{(1)}$ and δ is a statistical security parameter [17]. C_1 computes $\mu = [x_{pack}^{(1)}]_{pk} \prod_{k=1}^{K}\prod_{m=1}^{M}[x_m]_{pk}^{(N-1)2^{\varrho}}$. Note that, it is not difficult to find that μ is the ciphertext $[x_{1,1}^{(1)} - x_1 + r_{1,1}|\cdots|x_{K,M}^{(1)} - x_M + r_{K,M}]$. After partial decryption at the servers, $x_{k,m}^{(1)} - x_m + r_{k,m}$ is revealed by C_2 via unpacking. Subsequently, both servers perform the same operations as shown in Fig. 3. Obviously, only one encryption is required after data packing at C_1 in step 1, which saves much time compared with KM encryptions. Similarly, the corresponding number of decryptions is reduced to one for the servers.

Since SDH is the underlying protocol of secure weight estimation and a similar protocol applies to convergence determination, the computation cost at the servers can be reduced correspondingly via data packing. In Section 7, we will show the detailed cost savings in our optimized PETD (denoted as PETD-OPT). Theoretically, the total computation cost of C_1 and C_2 can be reduced to $(KM + K + 4M + 5)mul + (2KM + 4K + 11M + 13)exp$ and $(KM + 3K + 6M + 4)mul + (2KM + 4K + 9M + 5)exp$, respectively. In addition, PETD-OPT can save $(4KM - 4)l_2$ communication cost between the two servers (*i.e.*, the server-server communication is $(4KM + 6K + 8M + 6)l_2$).

6 Security Analysis

In this section, we provide a theoretical security analysis of our schemes. Follow the security analysis in [18], we construct simulators $\mathsf{Sim} = (\mathsf{Sim}_U, \mathsf{Sim}_{C_1}, \mathsf{Sim}_{C_2}, \mathsf{Sim}_R)$ to resist four kinds of adversaries $\mathcal{A} = (\mathcal{A}_U, \mathcal{A}_{C_1}, \mathcal{A}_{C_2}, \mathcal{A}_R)$ that corrupt the mobile user U, C_1, C_2, and the task requester R, respectively. Let $\mathcal{E}$ be an environment machine by which the inputs of uncorrupted parties are chosen. Security is defined by comparing a real-world execution and an ideal-world execution. A protocol π is considered secure against non-colluding semi-honest adversaries $\mathcal{A}$ if $\mathrm{REAL}[\mathcal{E}, \mathcal{A}, \pi, \lambda] \approx \mathrm{IDEAL}[\mathcal{E}, \mathcal{S}, f, \lambda]$, in which $\mathrm{REAL}[\mathcal{E}, \mathcal{A}, \pi, \lambda]$ denotes the final output (*i.e.*, view) of the environment $\mathcal{E}$ when interacting with $\mathcal{A}$ and honest parties who execute protocol π on the security parameter λ while $\mathrm{IDEAL}[\mathcal{E}, \mathcal{S}, f, \lambda]$ denotes the output of the environment $\mathcal{E}$ when interacting with Sim and honest parties who run the dummy protocol in presence of an ideal functionality f on λ. In other words, the views in the two worlds are indistinguishable.

Theorem 1. *In BPTD, if all parties are honest-but-curious and there is no collusion between the servers, the sensory data, weights, and the final estimated truths will not be disclosed to any other party (i.e., no direct privacy disclosure). Some intermediates are inevitably exposed but no private data can be inferred from the intermediates as long as the users do not collude with the server (i.e., no indirect privacy disclosure).*

Proof. In secure weight estimation, Sim_U receives each sensory data x as input and splits x into two shares in $\mathbb{Z}_{2^l}$. Sim_U sends one share to $\mathcal{A}_U$. Sim_{C_1} simulates $\mathcal{A}_{C_1}$ as follows: it randomly chooses data in $\mathbb{Z}_{2^l}$ and computes one share of $d(\mathbf{x}_k, \mathbf{x})$. Then, Sim_{C_1} computes the sum of distance partial shares and sends it to $\mathcal{A}_{C_1}$. Sim_{C_2} simulates $\mathcal{A}_{C_2}$ in a similar manner. By now, the views of $\mathcal{A}_U$, $\mathcal{A}_{C_1}$, and $\mathcal{A}_{C_2}$ are all indistinguishable in the real and ideal world as all data

4. The number of data in each pack is determined by the message space of Paillier Cryptosystem and the bit length of data to be packed.

are generated using uniformly random shares. This ascribes the security of additive secret sharing. In other words, the confidentiality of data is protected from direct disclosure as long as the data is shared (no reconstruction) between the two servers. For indirect privacy disclosure, it is a fact that some intermediate results are revealed to the parties due to data reconstruction. Next, we analyze that such reconstructions will not compromise the privacy of users and the requester in our non-colluding setting.

Specifically, in secure weight estimation, $\sum_{k=1}^{K} d(\mathbf{x}_k, \mathbf{x})$ is disclosed to C_1 while $d(\mathbf{x}_k, \mathbf{x})$ is revealed to each user u_k. The collusion between u_k and C_1 only allows u_k to learn his own weight w_k, still keeping w_k private to other parties. The users may collude with each other, a user colludes with a subset of users will not reveal any weight information while collusion among all users does not make sense as it also reveals the user's own weight. In secure truth estimation, we resort to C_2 to reconstruct $\sum_{k=1}^{K} w_k$. For the t-th iteration, the exposed intermediate results include $\sum_{k=1}^{K} d(\mathbf{x}_k, \mathbf{x}^t)$, $d(\mathbf{x}_k, \mathbf{x}^t)$, $\sum_{k=1}^{K} w_k^t$, and $d(\mathbf{x}^t, \mathbf{x}^{t-1})$ (*i.e.*, ν in Algorithm 3). C_1 and the users may continuously collect this data in each iteration and try to infer the sensory data and estimated truths. However, the estimated truth in each iteration is shared by the two servers, it is hence hidden from C_1 in the server-side non-collusion assumption. Although $(x^t - x^{t-1})^2$ is revealed to C_1, it is infeasible to determine x^t and x^{t-1}, respectively. Similarly, C_1 cannot infer the estimated truths from two unknown data x_1 and x_2. On the other hand, C_2 only learns $\sum_{k=1}^{K} w_k^t$, $\sum_{k=1}^{K} w_k^{t-1}$, and a share of weights and estimated truths, no sensory data, weights, and truths can be inferred from these intermediates, *i.e.*, no indirect privacy disclosure. Tang *et al.* [28] showed that the sensory data would be deduced by building K formulas after K iterations when the intermediate estimated truths and the above aggregated data in each iteration are exposed. In BPTD, the estimated truth $\mathbf{x}^t$ is shared between the servers, hence inferring the sensory data is infeasible in our non-collusion assumptions.

As analyzed, as long as the users do not collude with C_1 and the two servers do not collude with each other, the privacy of mobile users and the requester is well protected without direct and indirect privacy disclosure. □

Theorem 2. *In PETD, if all parties are honest-but-curious and there is no collusion between the servers, the sensory data, user weights, intermediate results, and the final estimated truths are oblivious to the adversaries, even if users collude with one server. In other words, there is no direct or indirect privacy disclosure.*

Proof. In PETD, we only provide a proof to show how to construct four independent simulators for Algorithm 4. The security of Algorithm 5 and secure convergence determination can be proved accordingly.

In secure weight estimation, Sim_U receives x as input and splits x into two shares in $\mathbb{Z}_{2^l}$. Sim_U sends one share to $\mathcal{A}_U$. We omit the simulation of SDH as its underlying secure multiplication protocol is proved secure in [18]. On the server side, Sim_{C_1} simulates $\mathcal{A}_{C_1}$ by generating two fictitious encryptions $D_k^{\beta_k}$ and $D^{\beta_k^2}$ with **Enc**() and randomly chosen data. Sim_{C_1} computes D'_k and $\mathcal{D}'_k$ with **PD1**() and sends $D_k^{\beta_k}$, $D^{\beta_k^2}$, D'_k and $\mathcal{D}'_k$ to $\mathcal{A}_{C_1}$. If $\mathcal{A}_{C_1}$ replies with $\perp$, then Sim_{C_1} returns $\perp$. Sim_{C_2} simulates $\mathcal{A}_{C_2}$ by randomly choosing a number z and computing $[z]_{pk}$ with **Enc**(). Sim_{C_2} sends $[z]_{pk}$ to $\mathcal{A}_{C_2}$. If $\mathcal{A}_{C_2}$ replies with $\perp$, then Sim_{C_2} returns $\perp$. Sim_R simulates $\mathcal{A}_R$ by generating encryption $[x_m]_{pk_r}$ of a randomly chosen data x_m, which is then sent to $\mathcal{A}_R$ and outputs $\mathcal{A}_R$'s view. For the above simulators, the view of $\mathcal{A}_U$ in the real and ideal worlds is indistinguishable due to the security of additive secret sharing. Hence the confidentiality of sensory data is guaranteed. The views of $\mathcal{A}_{C_1}$, $\mathcal{A}_{C_2}$, and $\mathcal{A}_R$ consist of the encrypted data they create, which are indistinguishable in the real and ideal executions due to the semantic security of PCPD. The intermediate and final results are well protected.

Regarding collusion attacks, first, collusion among different users does not make sense as it only reveals their own sensory data, rather than information of the non-colluding users. Additionally, these users are not able to infer the estimated truths without the decryption key of the data requester. Second, similar to the intra-user collusion case, users colluding with C_1 or C_2 will disclose nothing about other users' sensory data and weights, as well as the estimated truths. Third, for collusion between the data requester and any server, it is infeasible to derive the sensory data based on the partial data shares and the estimated truths. Moreover, C_1 cannot decrypt $[\mathbf{w}]_{pk}$ with the data requester's private key sk_r. Therefore, the weight privacy of each user is still preserved. Due to the same reason, intermediate results such as $\mathbf{w} \cdot \mathbf{x}_{*,m}$, $\mathbf{w} \cdot \mathbf{s}$ are also concealed. □

7 Experiments

7.1 Simulation Setup

As in the previous work, we used the well-known CRH [13] as the underlying TD method. As to computation and communication cost, we compare our schemes with three most relevant researches [48] (we use the second scheme PPTD-II for comparison due to its better overhead performance), [47] (we call the scheme GHTD), and [2] (or its journal version [3], we denote the scheme in [2] as PPTD-S). For a fair comparison, the above schemes are all based on CRH with the same dataset.

As in prior designs [28], [48], we used a synthetic dataset and all data was generated from a normal distribution with a random mean μ (represents the real ground truth) and a variance σ^2 (reflects the quality of observed data). Specifically, we generated a dataset with 100 mobile users and 50 tasks, and each user observed sensory data whose mean value $\mu \in [20, 40]$ and variance $\sigma^2 \in [1, 2]$. The size of each sensory data l_1 was set to be 64 bits, while the statistical parameter δ was set as 20 bits to achieve 2^{20} statistic security. In addition, N was set to be 1024 bits to achieve 80-bit security levels. We set P' in Algorithm 2 as 1000 to ensure $\widetilde{y} \neq 0$ in our scenario with 100 users. P and ϵ were set in the range from 10^0 to 10^5 and from 0.00001 to 1, respectively. For "log" computation in GHTD, we used Taylor Expansion with order 100 which is sufficient to ensure the accuracy.

7.2 Simulation Results

Accuracy. We first evaluate the accuracy of the final estimated truths of our solutions, with CRH as the baseline. Fig. 4 depicts the impact of P on the estimation accuracy

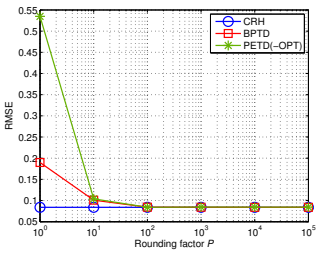


Fig. 4. Accuracy evaluation

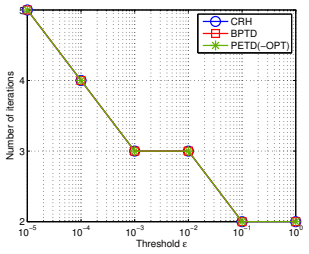
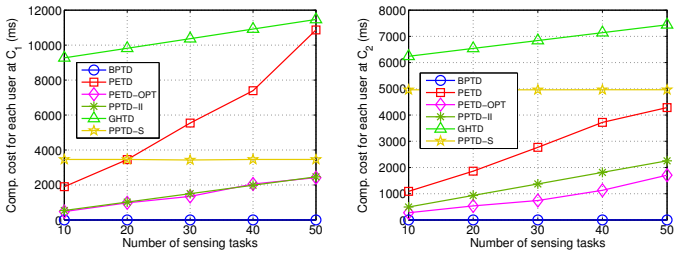
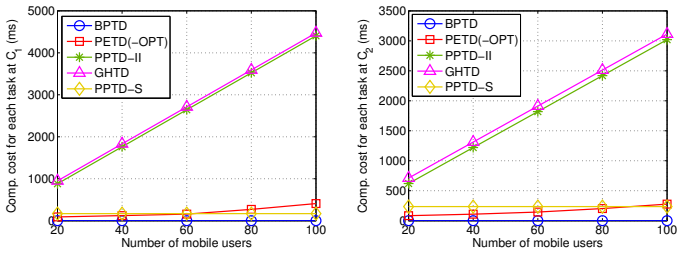


Fig. 5. Convergence determination



(a) (b)

Fig. 6. Computation cost at the servers in secure weight update. (a) On C_1 side. (b) On C_2 side.



(a) (b)

Fig. 7. Computation cost at the servers in secure truth update. (a) On C_1 side. (b) On C_2 side.

(the standard root of mean square error RMSE as the accuracy metric, $\epsilon = 0.01$). When P is 10^0, our two schemes have greater RMSE than CRH as the fractional parts of all sensory data and intermediate results are removed for computations on integers. As P increases, our estimation error decreases and is nearly the same as CRH when P is greater than 10^2. Hence, as long as P is large enough, the accuracy of our schemes will not be compromised. In the following experiments, unless otherwise stated, we set P as 10^5.

Convergence. We show in Fig. 5 the impact of ϵ on the number of iterations for three schemes. As expected, our schemes present the same number of iterations as CRH under varying ϵ. Additionally, we observe that the number of iterations does not always decrease as ϵ decreases. It is reasonable since less ϵ indicates a higher precision which usually needs more iterations (or the same sometimes).

Computation Overhead. Table 3 reports the user-side computation time *w.r.t* the number of tasks taken by each user. It is shown that the computation cost in all solutions grows as more tasks are executed. Among them, PETD and PPTD-S show the same least user-side computation. BPTD is expected to have negligible more time than PETD, which coincides with our theoretical analysis. In GHTD, since users undertake expensive data encryptions for sensing tasks, far more computations are needed than other solutions.

TABLE 3
User's Computation Cost (ms)

M	PPTD-II	GHTD	PPTD-S	Our schemes	
				BPTD	PETD
10	0.25 (13.8×)	40 (2222×)	0.018(1×)	0.019	0.018
20	0.37 (14.2×)	79 (3038×)	0.026 (1×)	0.027	0.026
30	0.52 (13.7×)	110 (2894×)	0.038 (1×)	0.041	0.038
40	0.63 (12.6×)	156 (3120×)	0.05 (1×)	0.053	0.05
50	0.79 (12.9×)	191 (3131×)	0.061 (1×)	0.064	0.061

Next, we fix the number of users to 100. Fig. 6 reports the impact of task quantity on the server-side computation cost for each user in each secure weight update. As observed, the server-side cost of most schemes increases as more tasks are performed, with BPTD exhibiting negligible cost due to the efficient modulo operations on shared data. In contrast, PETD takes both servers much more running time to estimate user's weight, which is more than that in PPTD-II. This is because computing $d(\mathbf{x}_k, \mathbf{x})$ in PETD is conducted on hybrid data, in which encryptions and partial decryptions require more computations than that in the plaintext domain in PPTD-II. PETD essentially provides stronger privacy guarantee at the cost of heavier computation overhead. Fortunately, this large cost is greatly reduced via our PETD-OPT. Moreover, PETD-OPT and PPTD-II have similar C_1-side cost and PETD-OPT takes C_2 less time than PPTD-II. Notably, despite a stable growth, GHTD still bears the most server-side computations as expensive cost is incurred for Taylor Expansion computation based on the secure multiplication protocol. For PPTD-S, it presents good scalability because the server-side cost is dominated by the GC-based computations (two logarithm and one division operations rely on GC, regardless of M). However, it still yeilds more time cost than PETD(-OPT). In real-world applications where each user generally participates in a few tens of tasks, our schemes exhibit more advantages.

Fig. 7 plots the server-side computation cost *w.r.t* the user quantity for each task in secure truth update. Similarly, BPTD shows the least and negligible cost. Different from Fig. 6, PETD presents far less computation cost at the servers than PPTD-II. This is because secure truth update is fully performed on encrypted data in PPTD-II, which requires to call SMP k times for k users. In contrast, PETD takes shared sensory data and encrypted weights as inputs, and only needs some modular exponentiation computations to derive the encrypted weighted data. Since optimization does not work when updating truth, PETD and PETD-OPT share the same cost. For GHTD, it presents comparable cost with PPTD-II but still yields the most computation cost. This is reasonable since GHTD and PPTD-II both use Paillier-based secure multiplication protocol to derive the encrypted sum of weighted truths and weights. Compared with Fig. 6, only one division is performed by the GC, and hence PPTD-S shows a lower cost at both servers per truth estimation, which is comparable with PETD(-OPT).

Last, Figs. 8(a) and 8(b) compare the server-side com-

putations in secure convergence determination. We omit GHTD and PPTD-S as they use the maximum number of iterations as convergence criteria. As reported, PETD incurs much more computation cost than BPTD at both servers, which, however, is greatly reduced in our PETD-OPT with up to 4.1× and 4× savings for C_1 and C_2, respectively. Both BPTD and PETD-OPT exhibit a better performance than PPTD-II. Hence, our schemes provide a more secure method to decide the convergence with higher efficiency.

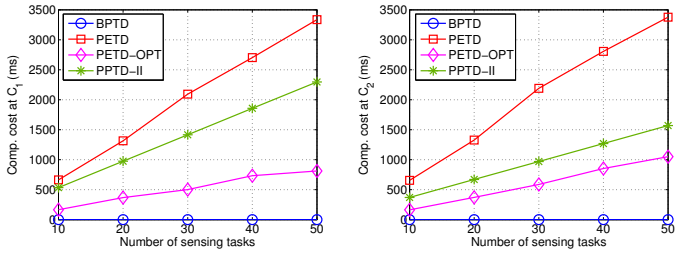


Fig. 8. Computation cost at the servers per convergence determination. (a) On C_1 side. (b) On C_2 side.

Communication overhead. Table 4 presents the user-server communication overhead in each scheme, in which our PETD and PPTD-S yield the fewest bytes. The reason is that they both only need to send two data shares to the servers for each task. BPTD bears more bytes as extra four data shares are transmitted between a user and the servers in each weight estimation (three iterations are involved under the setting $\epsilon = 0.01$). In contrast, the communication cost in PPTD-II is over two times as much as PETD. The additional cost also attributes to the frequent user-server interactions in weight estimation. For GHTD, each user needs to send the 2048-bit ciphertexts to C_1. Hence, it results in 16× bytes as that in our schemes.

TABLE 4
Communication Cost Between a User and the Servers (bytes)

M	PPTD-II	GHTD	PPTD-S	Our schemes	
				BPTD	PETD
10	448 (2.8×)	2560 (16×)	160 (1×)	256	160
20	848 (2.65×)	5120 (16×)	320 (1×)	416	320
30	1248 (2.6×)	7680 (16×)	480 (1×)	576	480
40	1648 (2.57×)	10240 (16×)	640 (1×)	736	640
50	2048 (2.56×)	12800 (16×)	800 (1×)	896	800

Storage overhead. Last, Table 5 compares the two server-side total storage cost in different schemes. Note that for each scheme, we only store the input and output data in the initialization and the three iterative procedures (if the output is an input of another algorithm, it is not counted repeatedly. If secure convergence determination is not involved in the scheme, then no associated output is stored). As shown in the table, BPTD needs the least storage cost for the servers and PETD bears a medium storage cost (less than PPTD-II and GHTD, and more than BPTD and PPTD-S). This is in accord with our complexity analysis in Section 5.2.1 and is reasonable for the sake of achieving non-interaction and enhanced privacy. As PPTD-II, GHTD and PETD all introduce homomorphic encryption for PPTD, the servers require to store the weights and truth ciphertexts. The higher storage overhead of PPTD-II and GHTD is due to the extra storage of sensory data ciphertexts (PETD only stores the sensory data shares). PPTD-S and BPTD both adopt additive secret sharing and hence present comparable storage cost. The reason for the slightly more cost in PPTD-S is that the servers need to store the shares of accumulated distance for each user in secure weight update.

TABLE 5
Server-side Total Storage Cost (bytes)

M	PPTD-II	GHTD	PPTD-S	Our schemes	
				BPTD	PETD
10	341320	343296	31232	26632	103832
20	605560	609536	48032	43592	130072
30	869800	875776	64832	60552	156312
40	1134040	1142016	81632	77512	182552
50	1398280	1408256	98432	94472	208792

8 Conclusion

In this paper, we investigate how to securely and efficiently discover truthful information from conflicting mobile crowdsensing data. We first propose a basic scheme BPTD enabling PPTD directly on shared data, which is very efficient under the help of two non-colluding servers and the mobile users, and ensures the most privacy requirements. To provide stronger privacy protection of intermediates with no user-side interactions, we further propose PETD with integration of a novel homomorphic encryption cryptosystem. BPTD and PETD show a different trade-off between security and efficiency, which caters for different applications according to the practical requirements. The server-side efficiency is further improved in PETD-OPT with data packing. Security analysis and experimental implementations validate the security and better performance of our schemes.

References

[1] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private collaborative forecasting and benchmarking. In *Proc. WPES*, pages 103–114, 2004.

[2] C. Cai, Y. Zheng, and C. Wang. Leveraging crowdsensed data streams to discover and sell knowledge: A secure and efficient realization. In *Proc. ICDCS*, pages 589–599. IEEE, 2018.

[3] C. Cai, Y. Zheng, A. Zhou, and C. Wang. Building a secure knowledge marketplace over crowdsensed data streams. *IEEE Transactions on Dependable and Secure Computing*, 18(6):2601–2616, 2021.

[4] D. Demmler, T. Schneider, and M. Zohner. ABYC: A framework for efficient mixed-protocol secure two-party computation. In *Proc. NDSS*, pages 1–9, 2015.

[5] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on information theory*, 29(2):198–208, 1983.

[6] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, and A. Moorsel. Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In *Proc. CCS*, pages 211–227, 2017.

[7] H. Duan, Y. Zheng, Y. Du, A. Zhou, C. Wang, and M. Au. Aggregating crowd wisdom via blockchain: A private, correct, and robust realization. In *Proc. PerCom*, pages 1–10. IEEE, 2019.

[8] L. Duan, T. Kubo, K. Sugiyama, J. Huang, T. Hasegawa, and J. Walrand. Incentive mechanisms for smartphone collaboration in data acquisition and distributed computing. In *Proc. INFOCOM*, pages 1701–1709, 2012.

[9] Y. Elmehdwi, B. K. Samanthula, and Wei Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *Proc. ICDE*, pages 664–675, 2014.

[10] J. Fan, Q. Li, and G. Cao. Privacy-aware and trustworthy data aggregation in mobile sensing. In *Proc. CNS*, pages 31–39, 2015.

[11] N. Gilboa. Two party rsa key generation. *Lecture Notes in Computer Science*, 1666:116–129, 1999.

[12] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *Proc. SIGCOMM*, pages 350–361, 2011.

[13] Q. Li, Y. Li, J. Gao, W. Fan, and J. Han. Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation. In *Proc. SIGMOD*, pages 1187–1198, 2014.

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

[15] Y. Li, C. Miao, L. Su, J. Gao, Q. Li, B. Ding, Z. Qin, and K. Ren. An efficient two-layer mechanism for privacy-preserving truth discovery. In *Proc. SIGKDD*, pages 1705–1714. ACM, 2018.

[16] Y. Li, H. Xiao, Z. Qin, C. Miao, L. Su, J. Gao, K. Ren, and B. Ding. Towards differentially private truth discovery for crowd sensing systems. *arXiv preprint arXiv:1810.04760*, 2018.

[17] A. Liu, Z. Kai, L. Lu, and G. Liu. Efficient secure similarity computation on encrypted trajectory data. In *Proc. ICDE*, pages 66–77, 2015.

[18] X. Liu, R. Deng, W. Ding, R. Lu, and B. Qin. Privacy-preserving outsourced calculation on floating point numbers. *IEEE Trans. Inform. Forens. Secur.*, 11(11):2513–2527, 2017.

[19] Y. Liu, S. Tang, Hao. Wu, and X. Zhang. Rtpt: A framework for real-time privacy-preserving truth discovery on crowdsensed data streams. *Computer Networks*, 148:349–360, 2019.

[20] Z. Liu, S. Jiang, P. Zhou, and M. Li. A participatory urban traffic monitoring system: The power of bus riders. *IEEE Trans. Intell. Transp. Syst.*, 18(10):2851–2864, 2017.

[21] F. Ma, Y. Li, Q. Li, M. Qiu, J. Gao, S. Zhi, L. Su, B. Zhao, H. Ji, and J. Han. Faitcrowd: Fine grained truth discovery for crowdsourced data aggregation. In *Proc. SIGKDD*, pages 745–754, 2016.

[22] C. Miao, W. Jiang, L. Su, Y. Li, S. Guo, Z. Qin, H. Xiao, J. Gao, and K. Ren. Cloud-enabled privacy-preserving truth discovery in crowd sensing systems. In *Proc. Sensys*, pages 183–196, 2015.

[23] C. Miao, W. Jiang, L. Su, Y. Li, S. Guo, Z. Qin, H. Xiao, Ji. Gao, and K. Ren. Privacy-preserving truth discovery in crowd sensing systems. *ACM Trans. Sensor Networks (TOSN)*, 15(9):1–32, 2019.

[24] C. Miao, L. Su, W. Jiang, Y. Li, and M. Tian. A lightweight privacy-preserving truth discovery framework for mobile crowd sensing systems. In *Proc. INFOCOM*, pages 1–9, 2017.

[25] P. Mohassel and Y. Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *Proc. S & P*, pages 19–38, 2017.

[26] X. Sheng, J. Tang, X. Xiao, and G. Xue. Sensing as a service: Challenges, solutions and future directions. *IEEE Sensors Journal*, 13(10):3733–3741, 2013.

[27] J. Tang, S. Fu, X. Liu, Y. Luo, and M. Xu. Achieving privacy-preserving and lightweight truth discovery in mobile crowdsensing. *IEEE Trans. Knowl. Data Eng.*, 2021, doi:10.1109/TKDE.2021.3054409.

[28] X. Tang, C. Wang, X. Yuan, and Q. Wang. Non-interactive privacy-preserving truth discovery in crowd sensing applications. In *Proc. INFOCOM*, pages 1–9, 2018.

[29] J. Vaidya and C. Clifton. Privacy-preserving data mining: why, how, and when. *Proc. S& P*, 2(6):19–27, 2004.

[30] D. Wang, J. Ren, Z. Wang, X. Pang, Y. Zhang, and X. Shen. Privacy-preserving streaming truth discovery in crowdsourcing with differential privacy. *IEEE Trans. Mob. Comput.*, 2021, doi: 10.1109/TMC.2021.3062775.

[31] Z. Wang, J. Hu, R. Lv, J. Wei, Q. Wang, D. Yang, and H. Qi. Personalized privacy-preserving task allocation for mobile crowdsensing. *IEEE Trans. Mob. Comput.*, 18(6):1330–1341, 2018.

[32] H. Wu, L. Wang, and G. Xue. Privacy-aware task allocation and data aggregation in fog-assisted spatial crowdsourcing. *IEEE Trans. Netw. Sci. Eng.*, 7(1):589–602, 2020.

[33] H. Wu, L. Wang, G. Xue, J. Tang, and D. Yang. Enabling data trustworthiness and user privacy in mobile crowdsensing. *IEEE/ACM Trans. Netw.*, 27(6):2294–2307, 2019.

[34] M. Xiao, G. Gao, J. Wu, S. Zhang, and L. Huang. Privacy-preserving user recruitment protocol for mobile crowdsensing. *IEEE/ACM Trans. Netw.*, 28(2):519–532, 2020.

[35] G. Xu, H. Li, D. Liu, H. Ren, Y. Dai, and X. Liang. Towards efficient privacy-preserving truth discovery in crowd sensing systems. In *Proc. Globecom*, pages 1–6, 2016.

[36] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu. Efficient and privacy-preserving truth discovery in mobile crowd sensing systems. *IEEE Trans. Vehicular Technology*, 68(4):3854–3865, 2019.

[37] G. Xu, H. Li, C. Tan, D. Liu, Y. Dai, and K. Yang. Achieving efficient and privacy-preserving truth discovery in crowd sensing systems. *Computers & Security*, 69:114–126, 2017.

[38] G. Xu, H. Li, S. Xu, H. Ren, Y. Zhang, J. Sun, and R.H. Deng. Catch you if you deceive me: Verifiable and privacy-aware truth discovery in crowdsensing systems. In *Proc. ASIA CCS*, pages 178–192, 2020.

[39] D. Yang, G. Xue, X. Fang, and J. Tang. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *Proc. MobiCom*, pages 173–184, 2012.

[40] D. Yang, G. Xue, X. Fang, and J. Tang. Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones. *IEEE/ACM trans. Netw.*, 24(3):1732–1744, 2015.

[41] Y. Yang, X. Liu, R.H. Deng, and Y. Li. Lightweight sharable and traceable secure mobile health system. *IEEE Trans. Depend. Secur. Comput.*, 17(1):78–91, 2020.

[42] C. Zhang, C. Xu, L. Zhu, Y. Li, C. Zhang, and H. Wu. An efficient and privacy-preserving truth discovery scheme in crowdsensing applications. *Computers & Security*, page 101848, 2020.

[43] C. Zhang, L. Zhu, C. Xu, X. Liu, and K. Sharif. Reliable and privacy-preserving truth discovery for mobile crowdsensing systems. *IEEE Transactions on Dependable and Secure Computing*, 18(3):1245–1260, 2019.

[44] C. Zhang, L. Zhu, C. Xu, K. Sharif, X. Du, and M. Guizani. LPTD: Achieving lightweight and privacy-preserving truth discovery in ciot. *Future Generation Computer Systems*, 90:175–184, 2019.

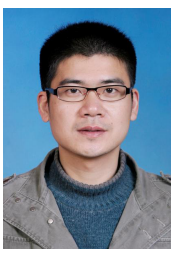
[45] C. Zhang, L. Zhu, C. Xu, K. Sharif, and X. Liu. Pptds: A privacy-preserving truth discovery scheme in crowd sensing systems. *Information Sciences*, 484:183–196, 2019.

[46] H. Zhao, M. Xiao, J. Wu, Y. Xu, H. Huang, and S. Zhang. Differentially private unknown worker recruitment for mobile crowdsensing using multi-armed bandits. *IEEE Transactions on Mobile Computing*, 20(9):2779–2794, 2020.

[47] Y. Zheng, H. Duan, and C. Wang. Learning the truth privately and confidently: Encrypted confidence-aware truth discovery in mobile crowdsensing. *IEEE Trans. Inform. Forens. Secur.*, 13(10):2475–2489, 2018.

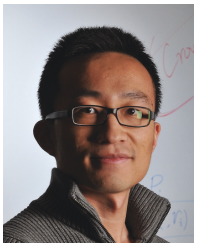
[48] Y. Zheng, H. Duan, X. Yuan, and C. Wang. Privacy-aware and efficient mobile crowdsensing with truth discovery. *IEEE Trans. Depend. Secur. Comput.*, 17(1):121–133, 2020.

Haiqin Wu received her B.S. degree in Computer Science and Ph.D degree in Computer Application Technology from Jiangsu University, Zhenjiang, China, in June 2014 and September 2019, respectively. She is currently a postdoctoral researcher in the Department of Computer Science at the University of Copenhagen. She was a visiting student in the School of Computing, Informatics, and Decision Systems Engineering at Arizona State University. Her research interests include data security and privacy protection, mobile crowdsensing, and blockchain-based applications.

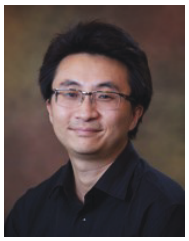


Liangmin Wang received his Ph.D degree in Cryptology from Xidian University, Xi'an, China. He is a full professor in the School of Cyber Science and Engineering, Southeast University, Nanjing, China. He has been honored as a "Wan-Jiang Scholar" of Anhui Province since Nov. 2013. His research interests include data security & privacy. He has published over 60 technical papers in premium journals and conferences, like IEEE/ACM TON, INFOCOM, IoT Journal, and TITS. He has served as a TPC member of many IEEE conferences, such as ICC, HPCC, TrustCom. He is an associate editor of Security and Communication Networks, a member of IEEE, ACM, and a senior member of CCF.

Ke Cheng received the B.Eng degree and the M.Eng degree from Anhui University, Hefei, China, in 2015 and in 2018, respectively. He is currently working toward the Ph.D degree in computer science at the School of Computer Science and Technology, Xidian University, Xi'an, China. He was a visting student at Boise State University. His research interests include security and privacy in cloud computing and big data.



Dejun Yang (M'13-SM'19) received the B.S. degree in computer science from Peking University, Beijing, China, in 2007 and the Ph.D degree in computer science from Arizona State University, Tempe, AZ, USA, in 2013. He is an Associate Professor of computer science with Colorado School of Mines, Golden, CO, USA. His research interests include IoT, networking, mobile sensing and computing, and privacy and security, with a focus on the application of game theory, optimization, algorithm design, and machine learning to resource allocation, security and privacy problems. He received the IEEE Communications Society William R. Bennett Prize in 2019, (best paper award for IEEE/ACM TON and IEEE TNSM in the previous three years), and Best Paper Awards at the IEEE GLOBECOM (2015), the IEEE MASS (2011), and the IEEE ICC (2011 and 2012), as well as a Best Paper Award Runner-up at the IEEE ICNP (2010). He is the TPC vice chair for information systems for the IEEE INFOCOM 2020, a student travel grant co-chair for INFOCOM 2018-2019, and was a symposium co-chair for the International Conference on Computing, Networking and Communications (ICNC) 2016. He currently serves as an Associate Editor for the IEEE Internet of Things Journal.



Jian Tang is a Professor in the Department of Electrical Engineering and Computer Science at Syracuse University, an IEEE Fellow and an ACM Distinguished Member. He received the Ph.D degree in Computer Science from Arizona State University in 2006. His research interests lie in the areas of AI, IoT, Wireless Networking, Mobile Computing and Big Data Systems. Dr. Tang has published over 150 papers in premier journals and conferences. He received an NSF CAREER award in 2009. He also received several best paper awards, including the 2019 William R. Bennett Prize of IEEE ComSoc and the 2019 TCBD (Technical Committee on Big Data) Best Journal Paper Award from IEEE Communications Society (ComSoc), the 2016 Best Vehicular Electronics Paper Award from IEEE Vehicular Technology Society (VTS), and Best Paper Awards from the 2014 IEEE ICC and the 2015 IEEE Globecom, respectively. He has served as an editor for several IEEE journals, including IEEE TBD, IEEE TMC, etc. He also served as a TPC co-chair for some international conferences, including the IEEE/ACM IWQoS'2019, MobiQuitous'2018, IEEE iThings'2015. etc.; as the TPC vice chair for the INFOCOM'2019; and as an area TPC chair for INFOCOM 2017-2018. He is also an IEEE VTS Distinguished Lecturer, and the Chair of the Communications Switching and Routing Committee of IEEE ComSoc.

Guoliang Xue is a professor of Computer Science and Engineering at Arizona State University. He received the Ph.D degree in Computer Science from the University of Minnesota in 1991. His research interests span the areas of Quality of Service provisioning, network security and privacy, crowdsourcing and network economics, RFID systems and Internet of Things. He has published in top conferences such as INFOCOM, MOBICOM, NDSS, and top journals such as IEEE/ACM Transactions on Networking, IEEE Journal on Selected Areas in Communications, and IEEE Transactions on Dependable and Secure Computing. He has received the IEEE Communications Society William R. Bennett Prize in 2019 and Best Paper Runner-up at ICNP'2010 (best paper award for IEEE/ACM TON and IEEE TNSM in the previous three years). He was a keynote speaker at IEEE LCN'2011 and ICNC'2014. He was a TPC Co-Chair of IEEE INFOCOM'2010 and a General Co-Chair of IEEE CNS'2014. He has served on the TPC of conferences including ACM CCS, ACM MOBIHOC, IEEE ICNP, and IEEE INFOCOM. He served on the editorial board of IEEE/ACM Transactions on Networking. He served as the Area Editor of IEEE Transactions on Wireless Communications, overseeing 13 editors in the Wireless Networking area. He is the Steering Committee Chair of IEEE INFOCOM. He is an IEEE Fellow.